Maximum Likelihood Estimation

Antonio Jurlina

2/26/2021

Problem 1. We derived the maximum likelihood (ML) estimator for the population parameters β and σ^2 in the classical linear regression model($Y_i = X_i\beta + \varepsilon_i$). Alternatively, the ML estimator can be obtained by first "concentrating" the log-likelihood function with respect to one of the parameters. For example, concentrating the function with respect to σ^2 – this is a convenient approach for problems that contain variance terms. This means differentiating the log-likelihood function with respect to σ^2, solving the resulting first-order condition for σ^2 as a function of the data and the remaining parameters, and then substituting the result back into the original log-likelihood function. This yields the concentrated log-likelihood function. The ML estimator for β can then be found my maximizing the concentrated log-likelihood function with respect to β. Here, we will demonstrate this process.

a) Deriving the ML estimator for σ^2 as a function of the data and the remaining parameters.

The probability density function[1] for the classical linear regression model, assuming $Y_i \sim N(X_i\beta, \sigma^2)$, is:

$$f(Y_i|X_i, \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(Y_i - X_i\beta)^2}{\sigma^2}}$$

From there we proceed to set up the (log) likelihood equation:

$$L(\theta|Y_N, X) = \prod_{i=1}^{N} f(Y_i|X_i, \theta)$$

$$ln[L(\theta|Y_N, X)] = \sum_{i=1}^{N} ln\left[f(Y_i|X_i, \theta)\right]$$

$$LL(\theta|Y_N, X) = \sum_{i=1}^{N} ln\left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(Y_i - X_i\beta)^2}{\sigma^2}}\right]$$

$$LL(\theta|Y_N, X) = \sum_{i=1}^{N} ln\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \sum_{i=1}^{N} ln\left(e^{-\frac{1}{2}\frac{(Y_i - X_i\beta)^2}{\sigma^2}}\right)$$

$$LL(\theta|Y_N, X) = Nln\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{1}{2}\frac{\sum_{i=1}^{N}(Y_i - X_i\beta)^2}{\sigma^2}$$

$$LL(\theta|Y_N, X) = Nln(1) - Nln\left(\sqrt{2\pi\sigma^2}\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(Y_i - X_i\beta)^2$$

[1] the general form:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

$$LL(\theta|Y_N, X) = -\frac{N}{2}ln(2\pi) - \frac{N}{2}ln\left(\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(Y_i - X_i\beta)^2$$

Now, we take a derivative of the log-likelihood function with respect to σ^2:

$$\frac{\partial LL}{\partial \sigma^2} = -\frac{N}{2}\frac{1}{\sigma^2} + \frac{1}{2\sigma^2\sigma^2}\sum_{i=1}^{N}(Y_i - X_i\beta)^2$$

Finally, setting the derivative to zero and solving for σ^2:

$$-\frac{N}{2}\frac{1}{\sigma^2} + \frac{1}{2\sigma^2\sigma^2}\sum_{i=1}^{N}(Y_i - X_i\beta)^2 = 0$$

$$\frac{-\sigma^2 N + \sum_{i=1}^{N}(Y_i - X_i\beta)^2}{2\sigma^2\sigma^2} = 0$$

$$-\sigma^2 N = -\sum_{i=1}^{N}(Y_i - X_i\beta)^2$$

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(Y_i - X_i\beta)^2$$

b) Using the answer from part a), we will derive the concentrated log-likelihood function.

$$LL(\theta|Y_N, X) = -\frac{N}{2}ln(2\pi) - \frac{N}{2}ln\left(\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(Y_i - X_i\beta)^2$$

Substituting in the answer for σ^2:

$$LL(\theta|Y_N, X) = -\frac{N}{2}ln(2\pi) - \frac{N}{2}ln\left(\frac{1}{N}\sum_{i=1}^{N}(Y_i - X_i\beta)^2\right) - \frac{1}{\frac{2}{N}\sum_{i=1}^{N}(Y_i - X_i\beta)^2}\sum_{i=1}^{N}(Y_i - X_i\beta)^2$$

$$LL(\theta|Y_N, X) = -\frac{N}{2}ln(2\pi) - \frac{N}{2}ln\left(\frac{1}{N}\sum_{i=1}^{N}(Y_i - X_i\beta)^2\right) - \frac{N}{2}$$

Switching to matrix form:

$$LL(\theta|Y_N, X) = -\frac{N}{2}ln(2\pi) - \frac{N}{2}ln\left[\frac{1}{N}(Y_N - X\beta)^T(Y_N - X\beta)\right] - \frac{N}{2}$$

$$LL(\theta|Y_N, X) = -\frac{N}{2}ln(2\pi) - \frac{N}{2}ln\left[\frac{1}{N}\left(Y_N^T Y_N - 2\beta^T X^T Y_N + \beta^T X^T X\beta\right)\right] - \frac{N}{2}$$

c) Using the concentrated log-likelihood (part b)) we derive the ML estimator for β.

First, we take a derivative of the log-likelihood function with respect to β:

$$\frac{\partial LL}{\partial \beta} = -\frac{N}{2}\frac{\frac{1}{N}\left(2X^T Y_N + 2X^T X\beta\right)}{\frac{1}{N}\left(Y_N^T Y_N - 2\beta^T X^T Y_N + \beta^T X^T X\beta\right)}$$

$$\frac{\partial LL}{\partial \beta} = \frac{-N\left(X^T Y_N + X^T X\beta\right)}{Y_N^T Y_N - 2\beta^T X^T Y_N + \beta^T X^T X\beta}$$

Then, we set the derivative to zero and solve for β.

$$\frac{-N\left(X^T Y_N + X^T X\beta\right)}{Y_N^T Y_N - 2\beta^T X^T Y_N + \beta^T X^T X\beta} = 0$$

$$\begin{aligned} X^T Y_N + X^T X\beta &= 0 \\ X^T X\beta &= X^T Y_N \\ \widehat{\beta}_{MLE} &= (X^T X)^{-1} X^T Y_N \end{aligned}$$

d) Using $\widehat{\beta}$ to adjust the ML estimator for σ^2 (from part a)).

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(Y_i - X_i\beta)^2$$

$$\sigma^2 = \frac{1}{N}\left(Y_N - X\beta\right)^T\left(Y_N - X\beta\right)$$

$$\sigma^2 = \frac{1}{N}\left(Y_N^T Y_N - 2\beta^T X^T Y_N + \beta^T X^T X\beta\right)$$

Substituting in the answer for β:

$$\sigma^2 = \frac{1}{N}\left[Y_N^T Y_N - 2\left((X^T X)^{-1} X^T Y_N\right)^T X^T Y_N + \left((X^T X)^{-1} X^T Y_N\right)^T X^T X\left((X^T X)^{-1} X^T Y_N\right)\right]$$

$$\sigma^2 = \frac{1}{N}\left[Y_N^T Y_N - 2Y_N^T X\left((X^T X)^{-1}\right)^T X^T Y_N + Y_N^T X\left((X^T X)^{-1}\right)^T X^T Y_N\right]$$

$$\sigma^2 = \frac{1}{N}\left[Y_N^T Y_N - Y_N^T X\left((X^T X)^{-1}\right)^T X^T Y_N\right]$$

Considering that $\left(A^{-1}\right)^T = \left(A^T\right)^{-1}$,

$$\widehat{\sigma^2}_{MLE} = \frac{1}{N}\left[Y_N^T Y_N - Y_N^T X(X^T X)^{-1} X^T Y_N\right]$$

e) Now, we demonstrate that the order did not matter. That is, we show that the ML estimators of σ^2 and β can be obtained by first concentrating with respect to β and then maximizing the concentrated log-likelihood thereby obtained, with respect to σ^2.

$$\widehat{\beta}_{MLE} = (X^T X)^{-1} X^T Y_N$$

$$LL(\theta|Y_N, X) = -\frac{N}{2}ln(2\pi) - \frac{N}{2}ln\left(\sigma^2\right) - \frac{1}{2\sigma^2}\left(Y_N - X\beta\right)^T\left(Y_N - X\beta\right)$$

$$LL(\theta|Y_N, X) = -\frac{N}{2}ln(2\pi) - \frac{N}{2}ln\left(\sigma^2\right) - \frac{1}{2\sigma^2}\left[Y_N^T Y_N - Y_N^T X(X^T X)^{-1} X^T Y_N\right]$$

$$\frac{\partial LL}{\partial \sigma^2} = -\frac{N}{2\sigma^2} + \frac{Y_N^T Y_N - Y_N^T X(X^T X)^{-1} X^T Y_N}{2\sigma^2\sigma^2}$$

$$-\frac{N}{2\sigma^2} + \frac{Y_N^T Y_N - Y_N^T X(X^T X)^{-1} X^T Y_N}{2\sigma^2\sigma^2} = 0$$

$$\frac{Y_N^T Y_N - Y_N^T X (X^T X)^{-1} X^T Y_N - \sigma^2 N}{2\sigma^2 \sigma^2} = 0$$

$$Y_N^T Y_N - Y_N^T X (X^T X)^{-1} X^T Y_N = \sigma^2 N$$

$$\widehat{\sigma^2}_{MLE} = \frac{1}{N} \left[Y_N^T Y_N - Y_N^T X (X^T X)^{-1} X^T Y_N \right]$$

Problem 2. Consider the nonlinear regression model (in which $\varepsilon_i \sim N(0, \sigma^2)$),

$$y_i = \beta_1 + \beta_2 e^{\beta_3 X_i} + \varepsilon_i.$$

In this problem, we will be writing MATLAB code to perform a Monte Carlo experiment to evaluate the small sample properties of two numerical maximization approaches (a grid search (*fminsearch*) and a gradient-based approach (*fminunc*)) for estimating the population parameters (which we will call θ) from equation above, using maximum likelihood. Our code will consist of two *m-files*: **1)** a base *m-file* that generates pseudo-data, calls the optimization routine, and stores estimates over the Monte Carlo loops, and **2)** a function *m-file* that calculates the log-likelihood.

To facilitate the pseudo-data experiment, suppose that X is uniformly distributed in the population ($X \sim U(-3, 6)$) and that the population parameters θ take the values $\beta_1 = 5$, $\beta_2 = 1.5$, $\beta_3 = -0.5$, and $\sigma^2 = 2$.

Part 1. *Create and evaluate the pseudo-data experiment.*

a) Generating $N = 2500$ observations of pseudo-data consistent with this population.

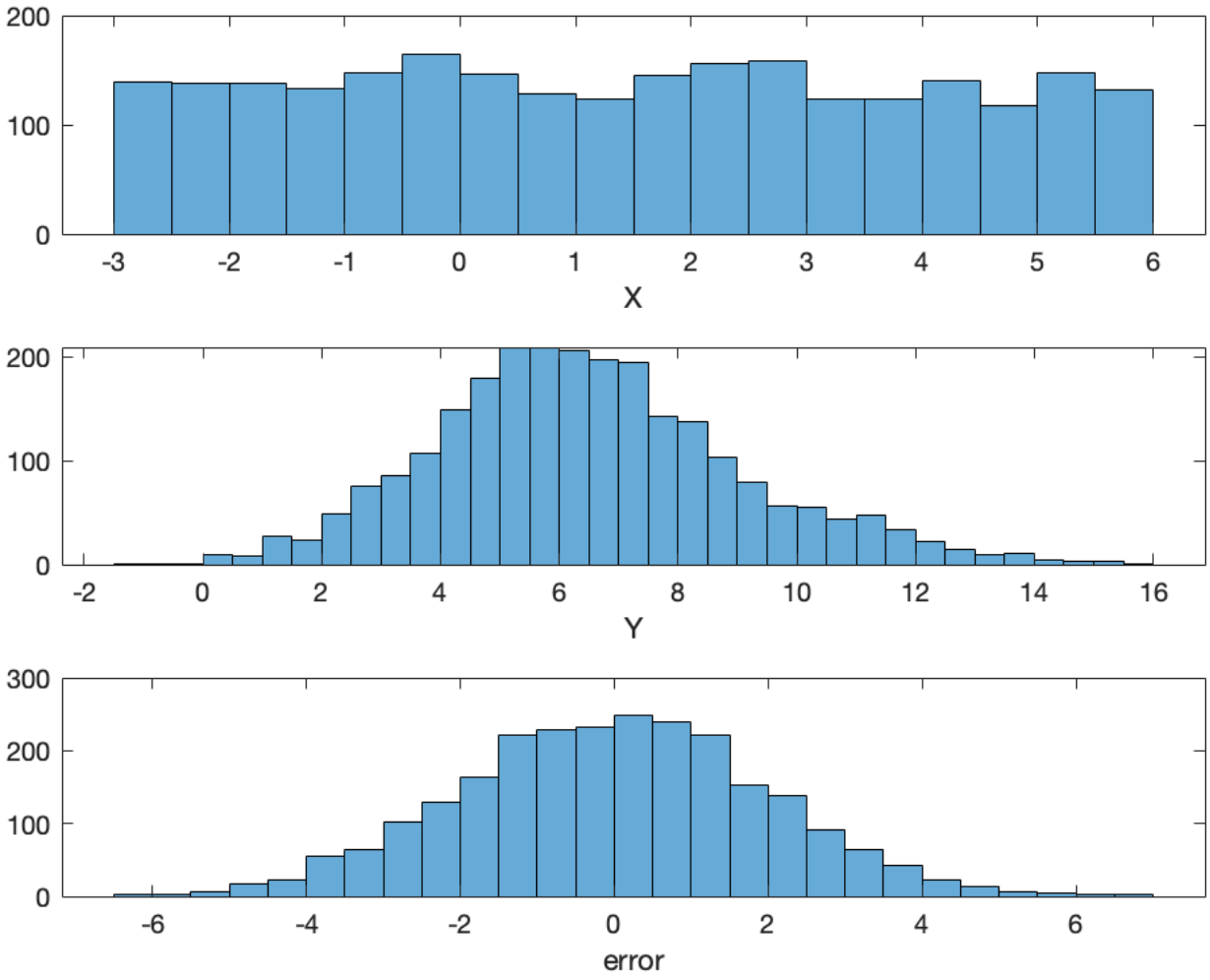

Figure 1: Pseudo-data histograms

```
n_obs = 2500;
theta = [5 1.5 -0.5 2]';
```

```
X_i = unifrnd(-3, 6, [n_obs, 1]);
error = normrnd(0, theta(4,1), [n_obs, 1]);

Y = theta(1,1) + theta(2,1) * exp(theta(3,1)*X_i) + error;

data = [X_i, Y];
```

b) Generating a table of summary statistics (average, median, standard deviation, minimum, and maximum) and histograms for X and y.

```
summary_stats = table([min(X_i), min(Y)]', ...
                      [mean(X_i), mean(Y)]', ...
                      [median(X_i), median(Y)]', ...
                      [max(X_i), max(Y)]', ...
                      [std(X_i), std(Y)]');
summary_stats.Properties.VariableNames = {'min' 'mean' 'median' 'max' 'std'};
summary_stats.Properties.RowNames = {'X' 'Y'};

figure % open new figure
subplot(3,1,1)
histogram(X_i)
xlabel('X')

subplot(3,1,2)
histogram(Y)
xlabel('Y')

subplot(3,1,3)
histogram(error)
xlabel('error')
```

Table 1: Pseudo-data summary statistics

	min	mean	median	max	std
X	-3.00	1.45	1.46	6.00	2.58
Y	-1.18	6.48	6.28	15.64	2.58

c) MATLAB has several minimization commands, but no real maximization commands. This is not a problem for us because maximizing $f(x)$ is equivalent to minimizing $-f(x)$. We wrote a function that calculates the negative of the log-likelihood for a given value of $\theta = \{\beta, \sigma^2\}$. This function is based on a log-likelihood function derived from the initial model statement.

$$y_i = \beta_1 + \beta_2 e^{\beta_3 X_i} + \varepsilon_i$$

$$LL(\theta|Y_N, X) = \sum_{i=1}^{N} ln\left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(Y_i-\beta_1-\beta_2 e^{\beta_3 X_i})^2}{\sigma^2}}\right]$$

$$LL(\theta|Y_N, X) = -\frac{N}{2} ln(2\pi) - \frac{N}{2} ln\left(\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(Y_i - \beta_1 - \beta_2 e^{\beta_3 X_i})^2$$

```
function [neg_LL] = objfun(theta, data)

  sigmasq = exp(theta(4,1));
```

```
  error = data(:,2) - theta(1,1) - theta(2,1)*exp(theta(3,1)*data(:,1));

  [n, k] = size(data);

  LL = -n/2*log(2*pi)-n/2*log(sigmasq)-(1/(2*sigmasq))*error'*error;

  neg_LL = -1 * LL

end
```

d) In our base *m-file*, we use the commands *fminsearch* and *fminunc* to numerically solve the log-likelihood for estimates of θ. Starting values will be random draws from a standard uniform distribution ($U \sim (0, 1)$).

```
options = optimset('Display','iter');
ivalues = rand(4,1);
theta_grid = fminsearch(@(theta) objfun(theta,data), ivalues, options);
theta_grad = fminunc(@(theta) objfun(theta,data), ivalues, options);

theta_grid(4) = theta_grid(4)^2;
theta_grad(4) = theta_grad(4)^2;

table(theta_grid,theta_grad, theta) % comparing the outputs and actual values
```

Table 2: Preliminary MLE outputs

	Grid estimate	Gradient estimate	Real value
Beta 1	5.07	5.07	5.0
Beta 2	1.36	1.37	1.5
Beta 3	-0.53	-0.53	-0.5
sigma_sq	1.87	1.87	2.0

Part 2. *The bootstrapping method.*

e) Running models on data with unknown parameters would introduce much more uncertainty than we have at the moment. In such a situation, when choosing starting values, we are not quite sure what the optimal ones would be and run the risk of picking some that end up getting stuck on outlier estimate solutions (which the grid search method is especially vulnerable to, as shown in Figure 2). Since we intend to use bootstrapping to estimate some standard errors on our estimates, we need to run the models many times over. However, we need to ensure that starting values are optimal (without actually knowing what the best ones would be), and then perform B bootstrap repetitions. To cut down on computing time, we first run the grid search method (the more sensitive one) with a 100 different starting value combinations and then select the combination which produces estimates closest to the median of the resulting distribution. This is not a perfect approach, but it does help us create a set of values we are more confident in reusing across our model runs.

```
initial_values = zeros(100,4);
answer = zeros(100,4);

parfor o = 1:100;
    ivalues = rand(4,1);
    answer(o,:) = fminsearch(@(theta) objfun(theta,data),...
                             ivalues, options);
    initial_values(o,:) = ivalues'
```

```
end

answer(:,4) = answer(:,4).^2

row_polish = round(sum((answer - median(answer)).^2, 2),2);
index = row_polish == min(row_polish);
choose_one = randi(sum(index), 1, 1);
initial_values_clean = initial_values(index,:);
ivalues = initial_values_clean(choose_one,:)';
```

f) With the bootstrap method, we calculate the standard errors for our θ estimates (for both theta_grid and theta_grad) using 10, 100, 200, and 1000 bootstrap samples. Our grid based search initially performs seemingly well in estimating the parameters of the model. However, increasing the sample size slowly, from 10 to 100, 200, and finally a 1000 resamplings, makes the standard error jump up drastically and then settle around a slightly lower (albeit still higher) error. This is most likely due to the fact that initially our search performed well without settling on any local maxima. However, with more repetitions, it became more likely an outlier would occur and affect our average values. Eventually, with a large enough number of resamplings, outlier effect was somewhat negated, and the errors went down. Our gradient based search was more robust to this effect and ended up providing reliably small standard errors that remained the same or decreased.

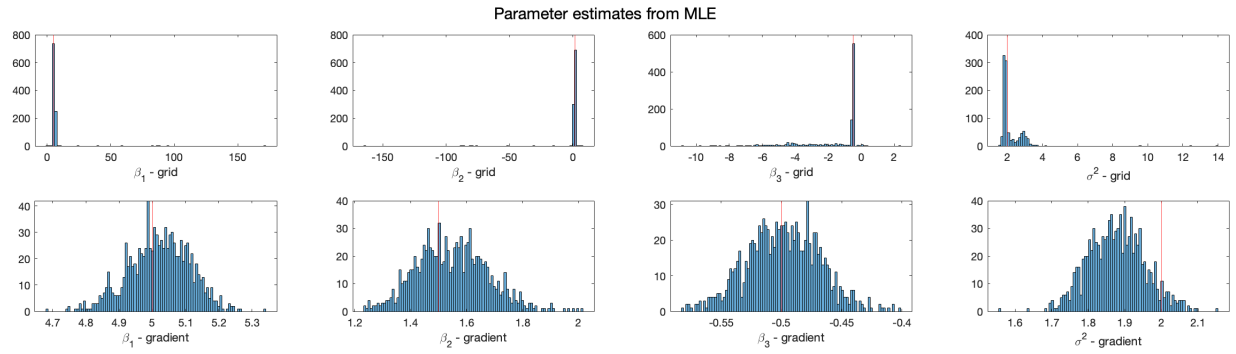

Figure 2: Bootstrapped parameter estimates (after 1000 runs)

```
rng('shuffle')      % Randomize draws for bootstrap

%B = 10;
%B = 100;
%B = 200;
%B = 1000;
theta_grid = zeros(B,4);
theta_grad = zeros(B,4);
parfor b = 1:B;
    bsindex = randsample((1:n_obs),n_obs,true);
    sample = [data(bsindex, 1), data(bsindex, 2)];
    grid = fminsearch(@(theta) objfun(theta,sample), ...
        ivalues, options);
    grad = fminunc(@(theta) objfun(theta,sample), ...
        ivalues, options);

    theta_grid(b,:) = grid;
    theta_grad(b,:) = grad;
```

```
end;

theta_grid(:,4) = theta_grid_1000(:,4).^2;
theta_grad(:,4) = theta_grad_1000(:,4).^2;

figure % open new figure
sgtitle('Parameter estimates from MLE');

subplot(4,4,1)
histogram(theta_grid_1000(:,1), 100)
xline(theta(1,1),'r');
xlabel('\beta_1 - grid');

subplot(4,4,2)
histogram(theta_grid_1000(:,2), 100)
xline(theta(2,1),'r');
xlabel('\beta_2 - grid');

subplot(4,4,3)
histogram(theta_grid_1000(:,3), 100)
xline(theta(3,1),'r');
xlabel('\beta_3 - grid');

subplot(4,4,4)
histogram(theta_grid_1000(:,4), 100)
xline(theta(4,1),'r');
xlabel('\sigma^2 - grid');

subplot(4,4,5)
histogram(theta_grad_1000(:,1), 100)
xline(theta(1,1),'r');
xlabel('\beta_1 - gradient');

subplot(4,4,6)
histogram(theta_grad_1000(:,2), 100)
xline(theta(2,1),'r');
xlabel('\beta_2 - gradient');

subplot(4,4,7)
histogram(theta_grad_1000(:,3), 100)
xline(theta(3,1),'r');
xlabel('\beta_3 - gradient');

subplot(4,4,8)
histogram(theta_grad_1000(:,4), 100)
xline(theta(4,1),'r');
xlabel('\sigma^2 - gradient');

% switch out the value in the denominator for 10, 100, 200, or 1000

var_cov_grid = (1/(1000-1))*(theta_grid-mean(theta_grid))'...
                                *(theta_grid-mean(theta_grid));
var_cov_grad = (1/(1000-1))*(theta_grad-mean(theta_grad))'...
```

```
*(theta_grad-mean(theta_grad));
```

Table 3: Bootstrapped standard errors

	grid (10)	gradient (10)	grid (100)	gradient (100)	grid (200)	gradient (200)	grid (1000)	gradient (1000)
Beta 1	0.52	0.11	0.54	0.08	12.18	0.09	8.21	0.09
Beta 2	0.68	0.20	0.69	0.13	12.10	0.13	8.14	0.13
Beta 3	0.65	0.05	1.79	0.03	1.99	0.03	1.75	0.03
sigma_sq	0.32	0.08	0.46	0.07	0.63	0.08	0.71	0.08

Part 3. *Monte Carlo experiment.*

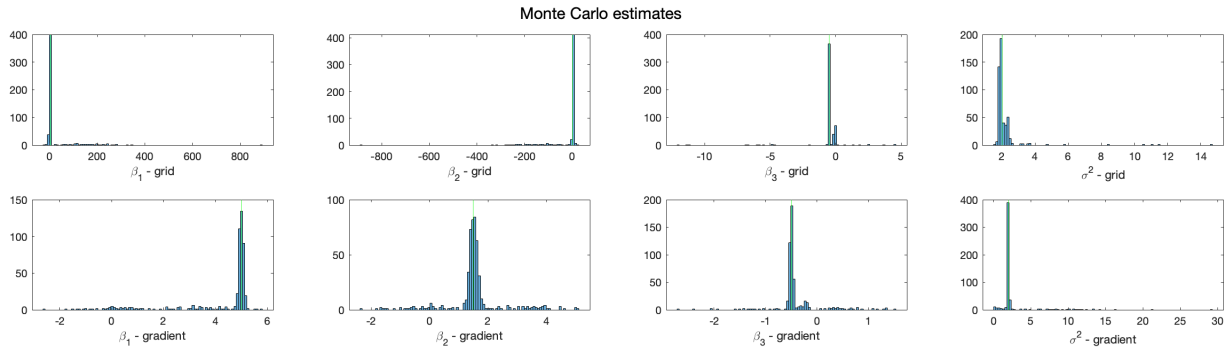

Figure 3: Monte Carlo estimates histograms

g) Now, we repeat the data generation and estimation (using random normal draws for starting values instead), a total of $R = 500$ times, collecting information about the estimates for β_1, β_2, β_3, and σ^2 for each numerical optimization approach (*fminsearch* and *fminunc*). Specifically, we provide:

- Summary statistics for the MLE estimates, including average, median, standard deviations, minumums and maximums.
- A histogram of the estimates of θ (using MATLAB functions *hist*, *figure*, and *subplot*).

Table 4: Monte Carlo estimates with grid search (500 repetitions)

	min	mean	median	max	std
Beta 1	-20.39	24.03	5.00	891.09	67.82
Beta 2	-883.75	-17.40	1.49	27.63	67.63
Beta 3	-12.01	-0.50	-0.49	4.60	1.19
sigma_sq	1.60	2.14	1.95	14.68	1.00

```
%% Monte Carlo
%R = 250;
R = 500;
%R = 25000;
theta_grid_MC = zeros(R,4);
theta_grad_MC = zeros(R,4);
parfor r=1:R;
    X_i = unifrnd(-3, 6, [n_obs, 1]);
```

```
    error = normrnd(0, theta(4,1), [n_obs, 1]);
    Y = theta(1,1) + theta(2,1) * exp(theta(3,1)*X_i) + error;
    data = [X_i, Y];

    starting_values = randn(4,1);

    grid = fminsearch(@(theta) objfun(theta,data), ...
        starting_values, options);
    grad = fminunc(@(theta) objfun(theta,data), ...
        starting_values, options);

    theta_grid_MC(r,:) = grid;
    theta_grad_MC(r,:) = grad;
end

theta_grid_MC(:,4) = theta_grid_MC(:,4).^2;
theta_grad_MC(:,4) = theta_grad_MC(:,4).^2;

%% Monte Carlo summary statistics
MC_grid = table(min(theta_grid_MC)', mean(theta_grid_MC)', ...
                median(theta_grid_MC)', max(theta_grid_MC)', ...
                std(theta_grid_MC)');

MC_grid.Properties.VariableNames = ...
    {'min' 'mean' 'median' 'max' 'std'};
MC_grid.Properties.RowNames=...
    {'beta_1' 'beta_2' 'beta_3' 'sigma^2'};

MC_grad = table(min(theta_grad_MC)', mean(theta_grad_MC)', ...
                median(theta_grad_MC)', max(theta_grad_MC)', ...
                std(theta_grad_MC)');

MC_grad.Properties.VariableNames = ...
    {'min' 'mean' 'median' 'max' 'std'};
MC_grad.Properties.RowNames=...
    {'beta_1' 'beta_2' 'beta_3' 'sigma^2'};

MC_grid
MC_grad

%% Monte Carlo plots
figure % open new figure
sgtitle('Monte Carlo estimates');

subplot(4,4,1)
histogram(theta_grid_MC(:,1), 100)
xline(theta(1,1),'g');
xlabel('\beta_1 - grid');

subplot(4,4,2)
histogram(theta_grid_MC(:,2), 100)
xline(theta(2,1),'g');
xlabel('\beta_2 - grid');
```

```
subplot(4,4,3)
histogram(theta_grid_MC(:,3), 100)
xline(theta(3,1),'g');
xlabel('\beta_3 - grid');

subplot(4,4,4)
histogram(theta_grid_MC(:,4), 100)
xline(theta(4,1),'g');
xlabel('\sigma^2 - grid');

subplot(4,4,5)
histogram(theta_grad_MC(:,1), 100)
xline(theta(1,1),'g');
xlabel('\beta_1 - gradient');

subplot(4,4,6)
histogram(theta_grad_MC(:,2), 100)
xline(theta(2,1),'g');
xlabel('\beta_2 - gradient');

subplot(4,4,7)
histogram(theta_grad_MC(:,3), 100)
xline(theta(3,1),'g');
xlabel('\beta_3 - gradient');

subplot(4,4,8)
histogram(theta_grad_MC(:,4), 100)
xline(theta(4,1),'g');
xlabel('\sigma^2 - gradient');
```

Table 5: Monte Carlo estimates with gradient search (500 repetitions)

	min	mean	median	max	std
Beta 1	-2.60	4.25	4.96	5.79	1.62
Beta 2	-2.36	1.53	1.51	5.11	0.91
Beta 3	-2.71	-0.42	-0.49	1.52	0.40
sigma_sq	0.00	2.39	1.93	29.21	2.43

h) According to the standard error estimates from Figure 5, the gradient search approach seems to be better at estimating the parameters of the model. This result is consistent across all bootstrapping sample sizes.

Table 6: Monte Carlo estimates with grid search (250 repetitions)

	min	mean	median	max	std
Beta 1	-15.09	20.33	5.01	299.06	51.38
Beta 2	-291.72	-13.71	1.48	22.42	51.17
Beta 3	-10.02	-0.40	-0.49	7.79	1.15
sigma_sq	0.00	2.09	1.96	9.26	0.67

Table 7: Monte Carlo estimates with gradient search (250 repetitions)

	min	mean	median	max	std
Beta 1	-1.04	4.41	4.98	7.74	1.48
Beta 2	-2.00	1.54	1.49	4.89	0.78
Beta 3	-2.05	-0.40	-0.49	2.66	0.46
sigma_sq	0.03	2.38	1.93	13.64	2.12

i) According to estimates from both grid and gradient search (Tables 4:9) we can see that both of our approaches are sensitive to sample size. Small samples are unlikely to show enough observations that would include outlier estimates, therebey potentially misleading our expected values. The gradient-based approach is less sensitive to the outlier effect, while the grid-based search algorithm settles on local maxima far from the searched-for global often enough that it warrants large sample sizes in order to mitigate these effects.

Table 8: Monte Carlo estimates with grid search (25,000 repetitions)

	min	mean	median	max	std
Beta 1	-104.43	23.96	5.01	3050.47	63.04
Beta 2	-3042.98	-17.32	1.48	111.77	62.83
Beta 3	-14.49	-0.45	-0.49	8.35	1.12
sigma_sq	0.00	2.11	1.95	26.11	0.86

Table 9: Monte Carlo estimates with gradient search (25,000 repetitions)

	min	mean	median	max	std
Beta 1	-9.01	4.29	4.97	71.76	1.72
Beta 2	-64.47	1.53	1.50	13.69	1.00
Beta 3	-5.48	-0.40	-0.49	3.22	0.43
sigma_sq	0.00	2.46	1.93	319.26	3.59

j) The results of the Monte Carlo estimation make sense given the way these problems were set up. At each repetition, starting values are sampled from a normal distribution (assuming no prior knowledge that would aid in this selection process). Grid search, which we expected to be more sensitive to settling on local maxima, indeed does seem to do so whenever particularly "mistaken" initial random draws are selected. The gradient-based approach is less sensitive to this and ends up with a much smaller variance of estimates across all Monte Carlo repetitions. Finally, it is important to note that, when bootstsrapping, we selected values more likely to converge towards true estimates, while during the Monte Carlo experiment, initial values were redrawn constantly. Had the Monte Carlo initial values been more optimized according to the true parameter values, we would have seen much less variance in final estimates.

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] knitr_1.31      forcats_0.5.0   stringr_1.4.0   dplyr_1.0.3
##  [5] purrr_0.3.4     readr_1.4.0     tidyr_1.1.2     tibble_3.0.5
##  [9] ggplot2_3.3.3   tidyverse_1.3.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.6        highr_0.8         cellranger_1.1.0  pillar_1.4.7
##  [5] compiler_4.0.3    dbplyr_2.0.0      tools_4.0.3       digest_0.6.27
##  [9] lubridate_1.7.9.2 jsonlite_1.7.2    evaluate_0.14     lifecycle_1.0.0
## [13] gtable_0.3.0      pkgconfig_2.0.3   rlang_0.4.10      reprex_0.3.0
## [17] cli_2.2.0         rstudioapi_0.13   DBI_1.1.1         yaml_2.2.1
## [21] haven_2.3.1       xfun_0.20         withr_2.4.1       xml2_1.3.2
## [25] httr_1.4.2        fs_1.5.0          hms_1.0.0         generics_0.1.0
## [29] vctrs_0.3.6       grid_4.0.3        tidyselect_1.1.0  glue_1.4.2
## [33] R6_2.5.0          fansi_0.4.2       readxl_1.3.1      rmarkdown_2.6
## [37] modelr_0.1.8      magrittr_2.0.1    backports_1.2.1   scales_1.1.1
## [41] ellipsis_0.3.1    htmltools_0.5.1.1 rvest_0.3.6       assertthat_0.2.1
## [45] colorspace_2.0-0  stringi_1.5.3     munsell_0.5.0     broom_0.7.3
## [49] crayon_1.4.1
```