Homework 3

Kensuke Suzuki

October 8, 2018

Problem 1: MLE with Nelder-Mead

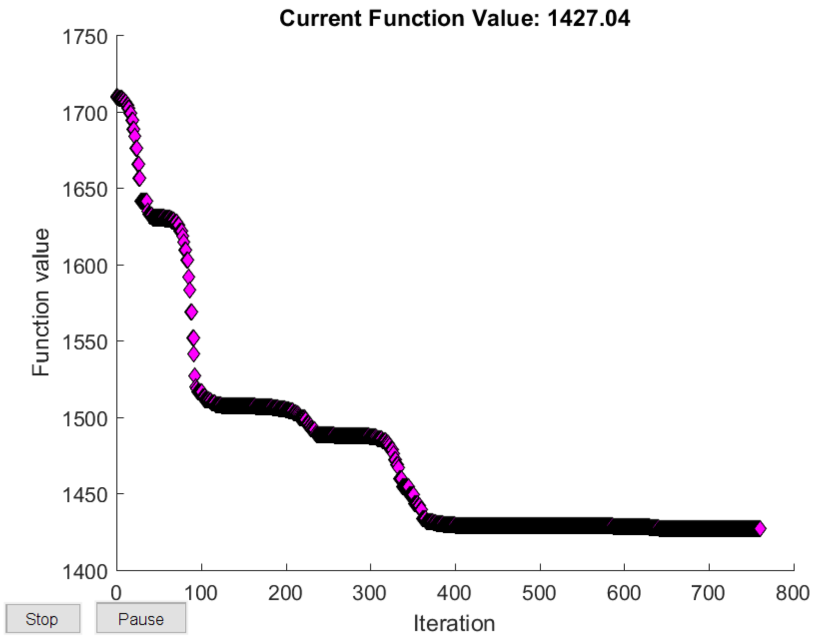
First define the function `TobitLLF.m` which gives the (negative of) log likelihood for given vector of β, $\mathbf{X}$ and $\mathbf{y}$.

```
function LLF = TobitLLF(beta,X,y)
f = -exp(X*beta) + y .* (X*beta) - log(factorial(y)) ;
%f = -exp(X*beta) + y .* (X*beta) ;
LLF = - sum(f);
end
```

Given the vector of $\mathbf{X}$ and $\mathbf{y}$ (data), we re-define anonymous function of β which we will minimize. In the first question, we use the Nelder-Mead method. Following the course material, we use the build-in optimizer `fminsearch`. For the initial value, we use $\beta_0 = \left[\log \frac{\sum_i y_i}{n}, 0, 0, 0, 0, 0\right]^\top$. Optimization algorithm yields the MLE estimates:

$$\beta = \begin{bmatrix} 2.5339 \\ -0.0323 \\ 0.1157 \\ -0.3540 \\ 0.0798 \\ -0.4094 \end{bmatrix}$$

The figure below shows the function values along the iteration.



Problem 2: MLE with Quasi-Newton

In this question, we use the build-in optimizer `fminunc`. We define the function `TobitLLF_grad.m` which returns the function value (negative of log likelihood) and gradient.

```
function [LLF, grad] = TobitLLF_grad(beta,X,y)
f = -exp(X*beta) + y .* (X*beta) - log(factorial(y)) ;
grad = -(-X' * exp(X*beta) + X' * y);
%f = -exp(X*beta) + y .* (X*beta) ;
LLF = - sum(f);
end
```

We specify the option (`option1`) to use this analytical gradient when using the built-in optimizer `fminunc`. The algorithm found the local minimum as the size of the gradient is less than the tolerance. Estimated MLE estimates are:

$$\beta = \begin{bmatrix} 2.5339 \\ -0.0323 \\ 0.1157 \\ -0.3540 \\ 0.0798 \\ -0.4094 \end{bmatrix}$$

The estimates are same as in the question 1.

Problem 3: NLS with lsqnonlin

We now define the function returning the sum of squared residuals, `NlsRSS`.

```
function RSS = NlsRSS(beta,X,y)
res = y - exp(X*beta);
RSS = (res' * res);
end
```

Given vector of $\mathbf{X}$ and $\mathbf{y}$, we define the anonymous function of β which we minimize.
We use the built-in optimizer `lsqnonlin`. We use the same initial value for β. As we discussed in the last question, the estimates are greatly affected by the initial guess. Since the default values for the maximum function evaluation (`MaxFunctionEvaluations`) and the maximum iteration (`MaxIterations`) are not enough for finding the minimum, we specify the option (`options3`) to increase these values. The results are:

$$\beta = \begin{bmatrix} 0.3895 \\ -0.0146 \\ 0.1193 \\ -0.1242 \\ 0.0770 \\ -0.1732 \end{bmatrix}$$

Problem 4: NLS with Nelder-Mead

In this question, we employ the Nelder-Mead method using `fminsearch`. Initial value for β is same as before. Again, as we discussed below, the results are greatly affected by the initial values. The optimization yields:

$$\beta = \begin{bmatrix} 2.5126 \\ -0.0384 \\ 0.1141 \\ -0.2796 \\ 0.0676 \\ -0.3698 \end{bmatrix}$$

Problem 5

For each method, we minimizes the function value with different initial values for β and see how the results are affected by the initial values. For here, we change the initial values for β_0 from 0 to 5 (with step 0.2) and keep $\beta_i = 0$ for $i = 1, ..., 5$

MLE with Nelder-Mead (Figure 1)

We specify the option (`options15`) to increase the maximum function evaluations and iterations. Results are presented in Figure 1. We find that for $\beta_0 \in [0.1, 0.9]$, the estimates seems to be fairly stable (independent of initial values). However, they vary substantially with different initial values for $\beta_0 \geq 1$.

MLE with Quasi-Newton (Figure 2)

Figure 2 is the result with Quasi-Newton method. Unlike the Nelder-Mead method, the Quasi-Newton method yields fairly stable results, i.e., the estimates are not dependent on initial values of β_0. It implies that the Quasi-Newton method are more robust than the Nelder-Mead method and it and yields the results which are independent of initial values.

NLS with `lsqnonlin` (Figure 3)

Figure 3 demonstrates the NLS estimates using `lsqnonlin` optimizer with different initial values. The left y-axis is the coefficients for β_0 and the right y-axis is for the rest of the parameters. The figure reveals that the estimates are affected by the initial values of β_0. Inspection of the figure demonstrates that the estimate for β_0 is almost linear in initial value. For the rest of the estimates, not as clear as in the case of β_0, we find that estimates are increasing or decreasing in the initial values.

NLS with Nelder-Mead (Figure 4)

Finally, the figure 4 demonstrates the results for NLS with Nelder-Mead. The left y-axis is the coefficients for β_0 and the right y-axis is for the rest of the parameters. As in Figure 1, the estimates are stable for the initial value $\beta \in [0.1, 1.5]$, but they vary with different intial values for $\beta \geq 1.6$ Unlike the `lsqnonlin` optimizer, we do not find clear linear relationship between initial guess and estimates.

Discussion:

For MLE, the Quasi-Newton method yields more robust result than the Nelder-Mead result. It would be because the Quasi-Newton method incorporates gradient of the objective function in minimizing the function value. For NLS, initial values matter in both algorithms. Still, Nelder-Mead method seems to have an interval of initial values which yields the stable estimates (this is confirmed in the case of MLE as well).

Matlab Code

```
% ECON512 Homework 3
% Kensuke Suzuki
clear all
delete HW3log.txt
diary('HW3log.txt')
diary on

disp('ECON512 HOMEWORK3: Ken Suzuki')

disp(' ')

data = load('hw3.mat');
X = data.X;
y = data.y;
```

Figure 1: MLE with Nelder-Mead

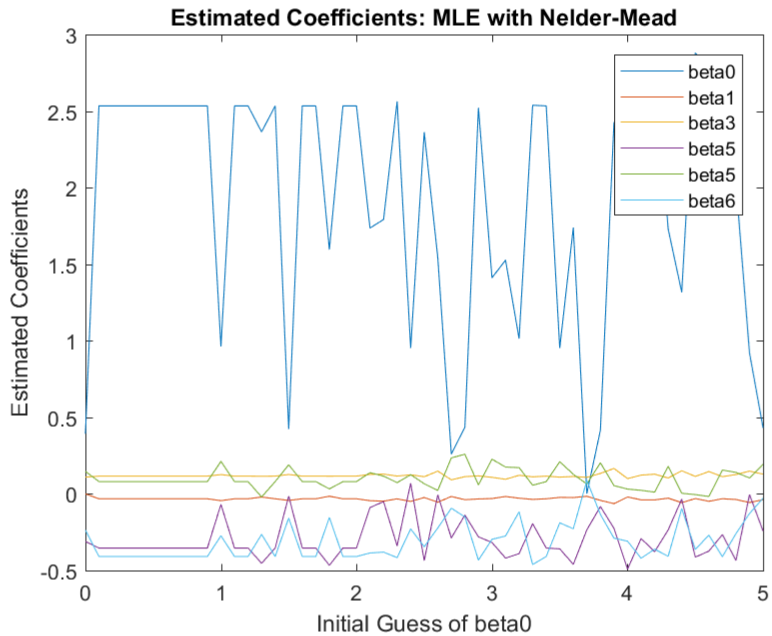


Figure 2: MLE with Quasi-Newton

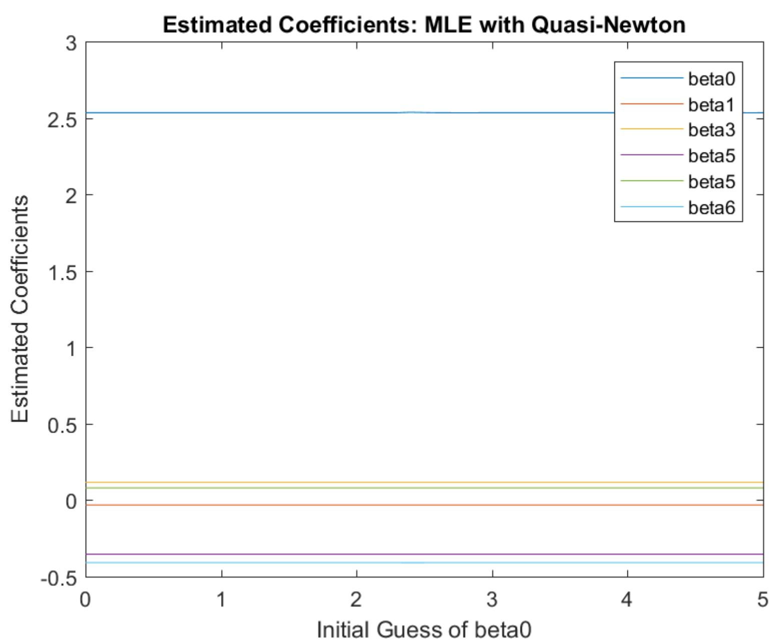


Figure 3: NLS with lsqnonlin

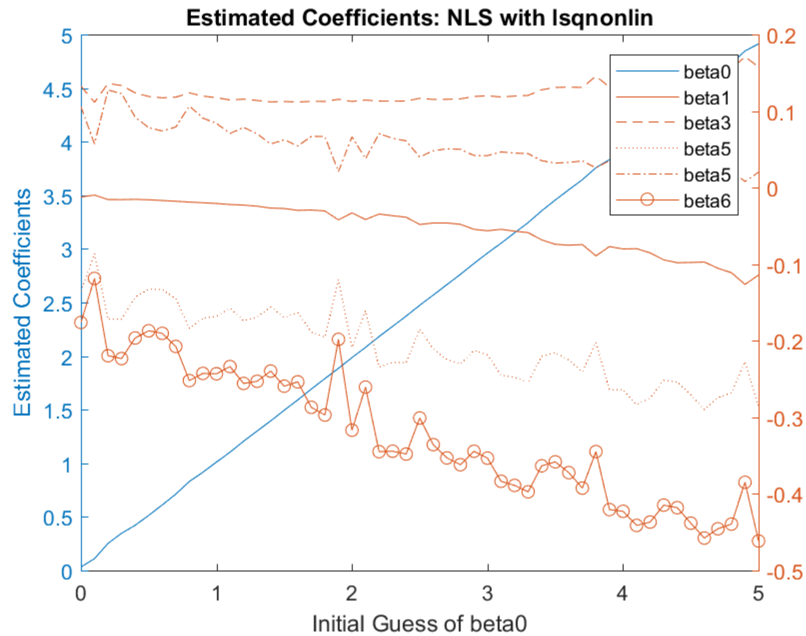
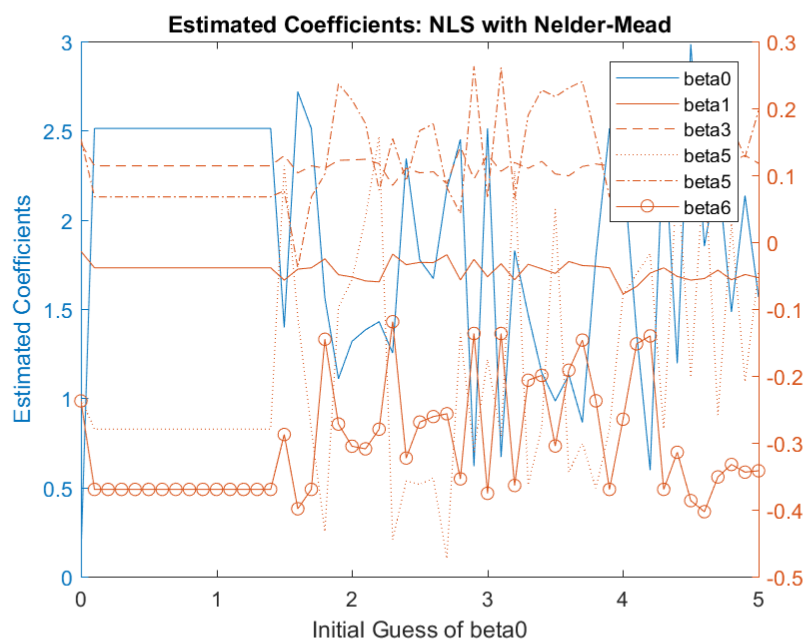


Figure 4: NLS with Nelder-Mead



```
Parameters = {'beta0'; 'beta1'; 'beta2'; 'beta3';'beta4'; 'beta5'};
%bindcell = cellstr(bind);
%% Problem 1: Nelder Mead Simplex Method

% define anonyumous function of beta
TobitLLF_beta =  @(beta) TobitLLF(beta,X,y);
beta0 = [log(mean(y)),zeros(1,5)]';

% options (as we specified in the class)
options1 = optimset('MaxFunEvals', 30000, 'MaxIter', 10000,'PlotFcns',
    @optimplotfval, 'Display','iter');
beta_p1 = fminsearch(TobitLLF_beta, beta0, options1);
saveas(gcf, 'q1.png')

EstimatedCoeff = beta_p1;
disp('———Problem 1———')
disp('Estimated Parameter Vector')
Result_Q1 = table(Parameters, EstimatedCoeff)

%% Problem 2: Quasi-Newton Optimization Method
clear beta

% define anonymouys function of beta: returns function value ans
    gradient
TobitLLF_grad_beta =  @(beta) TobitLLF_grad(beta,X,y);

% option using analytical gradient
options2 = optimoptions('fminunc','Algorithm','quasi-newton',...
            'SpecifyObjectiveGradient',true, 'Display','iter');
[beta_p2, LLF] = fminunc(TobitLLF_grad_beta, beta0, options2);

EstimatedCoeff = beta_p2;

disp('———Problem 2———')
disp('Method: Quasi-Newton Optimization (fminunc)')
disp('Estimated Parameter Vector')
Result_Q2 = table(Parameters, EstimatedCoeff)



%% Problem 3: NLS with lsqnonlin

NlsRSS_beta =  @(beta) NlsRSS(beta,X,y);
options3 = optimoptions(@lsqnonlin, 'MaxFunctionEvaluations', 30000, '
    MaxIterations', 10000)
beta_p3_1 = lsqnonlin(NlsRSS_beta, beta0, -Inf, +inf, options3)
Coeff1 = beta_p3_1;

```

```
disp('--------Problem 3-------')
disp('NLS with lsqnonlin')
disp('Estimated Parameter Vector')
disp('Coeff1: [log(mean(y)),zeros(1,5)] as initial guess')
disp('Coeff2: MLE estimator (question 1) as initial guess')
Result_Q3 = table(Parameters, Coeff1)


%% Problem 4: NLS with Nelder-Mead
%options4 = optimset('MaxFunEvals', 30000, 'MaxIter', 10000);

beta_p4_1 = fminsearch(NlsRSS_beta,beta0)
Coeff1 = beta_p4_1;

disp('--------Problem 4-------')
disp('NLS with Nelder-Mead')
disp('Estimated Parameter Vector')
disp('Coeff1: [log(mean(y)),zeros(1,5)] as initial guess')
disp('Coeff2: MLE estimator (question 1) as initial guess')
Result_Q4 = table(Parameters, Coeff1)

%% Problem 5

intbeta0 = [0:0.1:5];

% define option for Nelder-Mead: not plotting function values
options15 = optimset('MaxFunEvals', 30000, 'MaxIter', 10000);

% MLE with fminsearch
MLE_fmins = zeros(6,size(intbeta0,2));
for i = 1:size(intbeta0,2)
    beta0 = [intbeta0(1,i),zeros(1,5)]';
    beta = fminsearch(TobitLLF_beta, beta0, options15);
    MLE_fmins(:,i)=beta;
end
result_MLE_fmins = [intbeta0; MLE_fmins];

%MLE with quasi newton
MLE_fminunc = zeros(6,size(intbeta0,2));
for i = 1:size(intbeta0,2)
    beta0 = [intbeta0(1,i),zeros(1,5)]';
    [beta, LLF] = fminunc(TobitLLF_grad_beta, beta0, options2);
    %beta = fminunc(TobitLLF_beta, beta0);
    MLE_fminunc(:,i)=beta;
end
result_MLE_fminunc = [intbeta0; MLE_fminunc];

%NLS with lsqnonlin
```

```
NLS_lsqnonlin = zeros(6,size(intbeta0,2));
for i = 1:size(intbeta0,2)
    beta0 = [intbeta0(1,i),zeros(1,5)]';
    beta = lsqnonlin(NlsRSS_beta, beta0, -Inf, +inf, options3);
    NLS_lsqnonlin(:,i)=beta;
end
result_NLS_lsqnonlin = [intbeta0; NLS_lsqnonlin];

%NLS with Nelder Mead
NLS_fmins = zeros(6,size(intbeta0,2));
for i = 1:size(intbeta0,2)
    beta0 = [intbeta0(1,i),zeros(1,5)]';
    beta = fminsearch(NlsRSS_beta,beta0, options15);
    NLS_fmins(:,i)=beta;
end
result_NLS_fmins = [intbeta0; NLS_fmins];

% MLE with fminsearch
figure
plot(intbeta0, result_MLE_fmins(2,:), ...
    intbeta0, result_MLE_fmins(3,:), ...
    intbeta0, result_MLE_fmins(4,:), ...
    intbeta0, result_MLE_fmins(5,:), ...
    intbeta0, result_MLE_fmins(6,:), ...
    intbeta0, result_MLE_fmins(7,:))
title('Estimated Coefficients: MLE with Nelder-Mead')
xlabel('Initial Guess of beta0')
ylabel('Estimated Coefficients')
legend('beta0','beta1', 'beta3', 'beta5', 'beta5', 'beta6')
saveas(gcf, 'MLE_NM.png')

% MLE with Quasi Newton
figure
plot(intbeta0, result_MLE_fminunc(2,:), ...
    intbeta0, result_MLE_fminunc(3,:), ...
    intbeta0, result_MLE_fminunc(4,:), ...
    intbeta0, result_MLE_fminunc(5,:), ...
    intbeta0, result_MLE_fminunc(6,:), ...
    intbeta0, result_MLE_fminunc(7,:))
title('Estimated Coefficients: MLE with Quasi-Newton')
xlabel('Initial Guess of beta0')
ylabel('Estimated Coefficients')
legend('beta0','beta1', 'beta3', 'beta5', 'beta5', 'beta6')
saveas(gcf, 'MLE_QN.png')

% % NLS with lsqnonlin
% figure
% plot(intbeta0, result_NLS_lsqnonlin(2,:), ...
```

```
%     intbeta0, result_NLS_lsqnonlin(3,:), ...
%     intbeta0, result_NLS_lsqnonlin(4,:), ...
%     intbeta0, result_NLS_lsqnonlin(5,:), ...
%     intbeta0, result_NLS_lsqnonlin(6,:), ...
%     intbeta0, result_NLS_lsqnonlin(7,:))
% title('Estimated Coefficients: NLS with lsqnonlin')
% xlabel('Initial Guess of beta0')
% ylabel('Estimated Coefficients')
% legend('beta0','beta1', 'beta3', 'beta5', 'beta5', 'beta6')
% saveas(gcf, 'NLS_lsqnonlin.png')

% NLS with fminsearch
figure
yyaxis left
plot(intbeta0, result_NLS_lsqnonlin(2,:))
xlabel('Initial Guess of beta0')
ylabel('Estimated Coefficients')
title('Estimated Coefficients: NLS with lsqnonlin')
yyaxis right
plot(intbeta0, result_NLS_lsqnonlin(3,:), ...
    intbeta0, result_NLS_lsqnonlin(4,:), ...
    intbeta0, result_NLS_lsqnonlin(5,:), ...
    intbeta0, result_NLS_lsqnonlin(6,:), ...
    intbeta0, result_NLS_lsqnonlin(7,:))
legend('beta0','beta1', 'beta3', 'beta5', 'beta5', 'beta6')
saveas(gcf, 'NLS_lsqnonlin.png')


% NLS with fminsearch
figure
yyaxis left
plot(intbeta0, result_NLS_fmins(2,:))
xlabel('Initial Guess of beta0')
ylabel('Estimated Coefficients')
title('Estimated Coefficients: NLS with Nelder-Mead')
yyaxis right
plot(intbeta0, result_NLS_fmins(3,:), ...
    intbeta0, result_NLS_fmins(4,:), ...
    intbeta0, result_NLS_fmins(5,:), ...
    intbeta0, result_NLS_fmins(6,:), ...
    intbeta0, result_NLS_fmins(7,:))
legend('beta0','beta1', 'beta3', 'beta5', 'beta5', 'beta6')
saveas(gcf, 'NLS_NM.png')

diary off
```