Econ 512 HW 3

Yinshi Gao
yzg115

1. Using MLE Computed via the Nelder-Mead Simplex Method

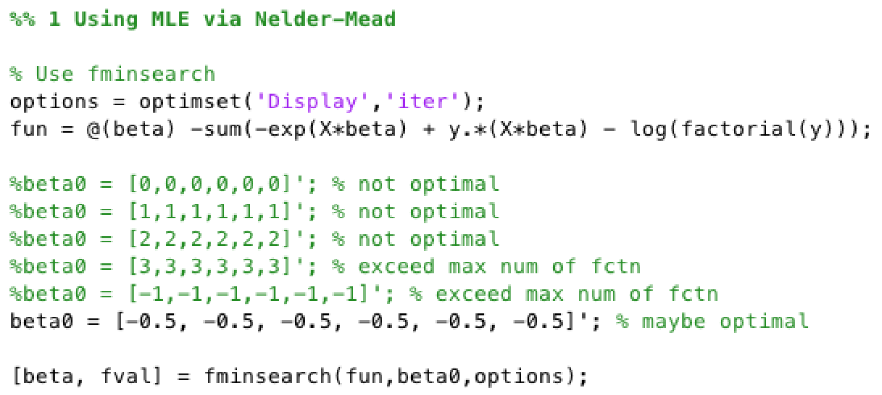
Since the default method matlab function *fminsearch* uses is Nelder-Mead simplex method, I directly apply *fminsearch* on $(-1)\times$ maximum likelihood function. (minimizing $(-1)\times$ maximum likelihood function is equivalent to maximizing maximum likelihood function.)

I test this method with 6 initial values. I have made comments behind each initial value, regarding to the result. There are some explanations for the comment:

- "not optimal": even though the algorithm converges, it is definitely not the global optimum.
- "maybe optimal": it is the minimum value gotten from 6 calculations, though still not sure whether it is the global optimum or not.

By using initial value of $\beta_0 = [-0.5, -0.5, -0.5, -0.5, -0.5, -0.5]'$, the number of iteration needed is 675. (For later comparison.) And the estimated $\beta = [2.5338, -0.0323, 0.1157, -0.3540, 0.0798, -0.4094]'$.

Program



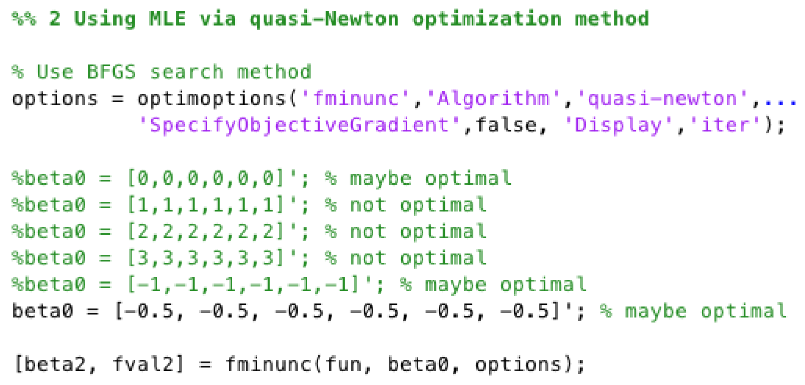
```
%% 1 Using MLE via Nelder-Mead

% Use fminsearch
options = optimset('Display','iter');
fun = @(beta) -sum(-exp(X*beta) + y.*(X*beta) - log(factorial(y)));

%beta0 = [0,0,0,0,0,0]'; % not optimal
%beta0 = [1,1,1,1,1,1]'; % not optimal
%beta0 = [2,2,2,2,2,2]'; % not optimal
%beta0 = [3,3,3,3,3,3]'; % exceed max num of fctn
%beta0 = [-1,-1,-1,-1,-1,-1]'; % exceed max num of fctn
beta0 = [-0.5, -0.5, -0.5, -0.5, -0.5, -0.5]'; % maybe optimal

[beta, fval] = fminsearch(fun,beta0,options);
```

2. Using MLE Computed via a Quasi-Newton Optimization Method

Here, BFGS search method is used in the quasi-newton method. Still, the method is tested with the same 6 initial values as in problem 1. This time, there are 3 initial values giving "maybe optimal" results.

By using intial value of $\beta_0 = [-0.5, -0.5, -0.5, -0.5, -0.5, -0.5]'$, the number of iteration needed is 25. And the estimated β (denoted in Problem 2 as β_2) is $\beta_2 = [2.5339, -0.0323, 0.1157, -0.3540, 0.0798, -0.4094]'$.

Program



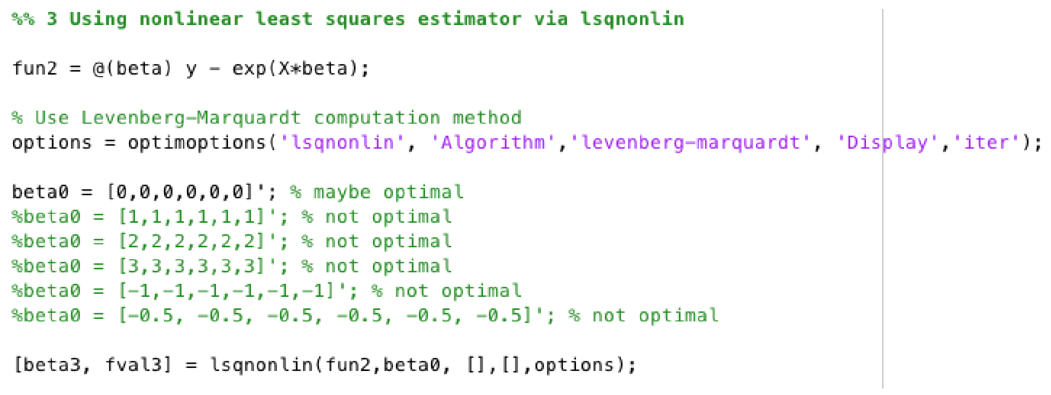
```
%% 2 Using MLE via quasi-Newton optimization method

% Use BFGS search method
options = optimoptions('fminunc','Algorithm','quasi-newton',...
          'SpecifyObjectiveGradient',false, 'Display','iter');

%beta0 = [0,0,0,0,0,0]'; % maybe optimal
%beta0 = [1,1,1,1,1,1]'; % not optimal
%beta0 = [2,2,2,2,2,2]'; % not optimal
%beta0 = [3,3,3,3,3,3]'; % not optimal
%beta0 = [-1,-1,-1,-1,-1,-1]'; % maybe optimal
beta0 = [-0.5, -0.5, -0.5, -0.5, -0.5, -0.5]'; % maybe optimal

[beta2, fval2] = fminunc(fun, beta0, options);
```

3. Using Nonlinear Least Square Estimator Computed with Command *lsqnonlin*

Here, Levenberg-Marquardt computation method is used, and the optimization method is tested with 6 initial values.

When using intial value of $\beta_0 = [0,0,0,0,0,0]'$, the number of iteration needed is 7. And the estimated β (denoted in Problem 3 as β_3) is $\beta_3 = [2.5120, -0.0384, 0.1141, -0.2799, 0.0677, -0.3697]'$.

Program



```
%% 3 Using nonlinear least squares estimator via lsqnonlin

fun2 = @(beta) y - exp(X*beta);

% Use Levenberg-Marquardt computation method
options = optimoptions('lsqnonlin', 'Algorithm','levenberg-marquardt', 'Display','iter');

beta0 = [0,0,0,0,0,0]'; % maybe optimal
%beta0 = [1,1,1,1,1,1]'; % not optimal
%beta0 = [2,2,2,2,2,2]'; % not optimal
%beta0 = [3,3,3,3,3,3]'; % not optimal
%beta0 = [-1,-1,-1,-1,-1,-1]'; % not optimal
%beta0 = [-0.5, -0.5, -0.5, -0.5, -0.5, -0.5]'; % not optimal

[beta3, fval3] = lsqnonlin(fun2,beta0, [],[],options);
```

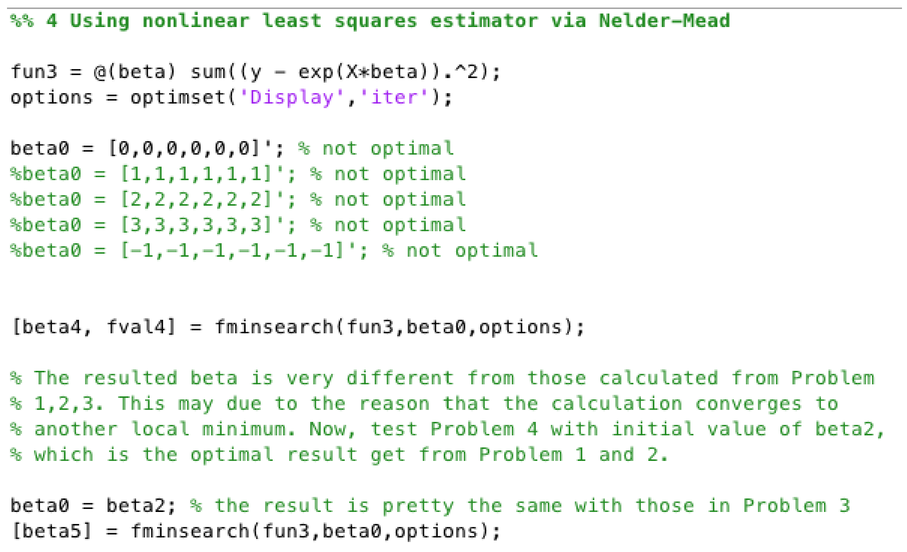
4. Using Nonlinear Least Square Estimator Computed via the Nelder-Mead Simplex Method

The method is tested with the same 6 initial value, however, no optimal value could be gotten, when comparing with the result in Problem 3 which uses the same objective function.

For further testing, I set the initial value to the "maybe optimal" value resulted from calculation in Problem 2 to see what will happen. This time, the result is very close to the value gotten from Problem 3.

When using intial value of $\beta_0 = \beta_2$, the number of iteration needed is 301. And the estimated β (denoted in Problem 4 as β_5) is $\beta_5 = [2.5126, -0.0384, 0.1141, -0.2796, 0.0676, -0.3697]'$.

Program



```
%% 4 Using nonlinear least squares estimator via Nelder-Mead

fun3 = @(beta) sum((y - exp(X*beta)).^2);
options = optimset('Display','iter');

beta0 = [0,0,0,0,0,0]'; % not optimal
%beta0 = [1,1,1,1,1,1]'; % not optimal
%beta0 = [2,2,2,2,2,2]'; % not optimal
%beta0 = [3,3,3,3,3,3]'; % not optimal
%beta0 = [-1,-1,-1,-1,-1,-1]'; % not optimal


[beta4, fval4] = fminsearch(fun3,beta0,options);

% The resulted beta is very different from those calculated from Problem
% 1,2,3. This may due to the reason that the calculation converges to
% another local minimum. Now, test Problem 4 with initial value of beta2,
% which is the optimal result get from Problem 1 and 2.

beta0 = beta2; % the result is pretty the same with those in Problem 3
[beta5] = fminsearch(fun3,beta0,options);
```

5. Comparing the Above 4 Approaches

It is easy to see that, over all, Nelder-Mead simplex method uses more iterations both in MLE and least square estimator models.

Using the same initial values, it seems that MLE with Quasi-Newton method in Problem 2 is more robust, since 3 out of 6 initial values give "maybe optimal" results.

Giving a rank of robustness, (">" means "more robust"):

MLE with Quasi-Newton > MLE with Nelder-Mead > nonlinear least square with *lsqnonlin* > nonlinear least square with Nelder-Mead

Giving a rank of convergence time, ("<" means "using less iterations"):

nonlinear least square with *lsqnonlin* < MLE with Quasi-Newton < nonlinear least square with Nelder-Mead < MLE with Nelder-Mead