HW3 Report

Ece Teoman

October 10, 2018

Here, I include the code and the output but the m.files and the diary can be found in the directory.

1

Estimate of β using the Nelder-Mead simplex method is:

betanm =

2.5339

−0.0323

0.1157

−0.3540

0.0798

−0.4094

2

Estimate of β via a quasi-Newton optimization method, namely the unconstrained minimization method `fminunc` is:

betaqn =

2.5339

−0.0323

0.1157

−0.3540

0.0798

−0.4094

3

Estimate of β using NLS estimator computed using the command `lsqnonlin` :

betanls =

2.5126

−0.0384

0.1141

−0.2796

0.0676

−0.3698

4

Estimate of β using NLS estimator computed using the Nelder-Mead simplex method is:

betanlsnm =

2.5126

−0.0384

0.1141

−0.2796

0.0676

−0.3698

5

I initially used a vector of zeros as my initial 'guess' for β. For this last part, I tested these four methods by using vectors of ones, twos and negative ones. For some reason, for vectors of fours and above, NLS and NLS via Nelder-Mead kept giving me errors and I didn't have enough time to diagnose the problem. I report the estimates by indexing them with the initial values and report the time elapsed right below of each.
For Nelder-Mead method:

```
betanm1 =

2.5339
−0.0323
0.1157
−0.3540
0.0798
−0.4094

Elapsed time is 0.084738 seconds.
```

```
betanm2 =

    2.5339
   -0.0323
    0.1157
   -0.3540
    0.0798
   -0.4094

Elapsed time is 0.084174 seconds.


betanm_1 =

    2.5339
   -0.0323
    0.1157
   -0.3540
    0.0798
   -0.4094

Elapsed time is 0.073461 seconds.
```

For quasi-Newton:

```
betaqn1 =

    0.9825
```

```
0
0.7368
0.9196
0.8973
0.9142

Elapsed time is 0.009441 seconds.



betaqn2 =

1.9825
1.0000
1.7368
1.9155
1.8952
1.9125

Elapsed time is 0.006987 seconds.



betaqn_1 =

2.5339
-0.0323
0.1157
```

−0.3540

0.0798

−0.4094

Elapsed time is 0.040698 seconds.

For NLS:

−386.2101

6.5045

−5.7481

6.6512

−6.2720

19.0902

Elapsed time is 0.697951 seconds.

betanls2 =

−416.4367

6.9547

−6.1263

7.4460

−6.2516

20.3269

Elapsed time is 1.089859 seconds.

```
betanls_1 =


−1
−1
−1
−1
−1
−1

Elapsed time is 0.007049 seconds.
```

For NLS via Nelder-Mead:

```
betanlsnm1 =


3.7840
−0.0714
0.1261
−0.2841
0.0401
−0.4330

Elapsed time is 0.052179 seconds.



betanlsnm2 =
```

```
2.3735
-5.8206
3.0009
3.2772
1.2728
4.2675

Elapsed time is 0.003642 seconds.


betanlsnm_1 =

2.5126
-0.0384
0.1141
-0.2796
0.0676
-0.3698

Elapsed time is 0.057874 seconds.
```

Looking at these results, a rough ranking would be

- Nelder-Mead: pretty accurate and not too slow.
- NLS via Nelder-Mead: fairly quick and not too far from the accurate values.
- quasi-Newton: pretty quick but not very accurate.
- NLS with `lsqnonlin`: inaccurate and slow.

6 MATLAB codes

Here are the functions and the rest of the code I've written to get the answers above:

6.1 Functions

```
function negll = negll( beta, X, y )

negll =-sum(-exp(X*beta) + X*beta.*y - log(factorial(y)));

end

function nls = ssnls( beta, X, y )

nls =sum((y-exp(X*beta)).^2);

end
```

6.2 Main

```
%% Q1
load('hw3.mat')

beta0=zeros(6, 1);

options=optimset('MaxIter', 10000, 'TolFun', 1e-16, 'MaxFunEvals', 10000);
fnm= @(beta)negll(beta, X, y);
[betanm, fvalnm, exitflagnm, outputnm] = fminsearch(fnm, beta0, options)
```

```
%% Q2

[betaqn, fvalqn, exitflagqn, outputqn] = fminunc(@(beta) ...
    negll(beta, X, y), beta0)

%% Q3

fnls= @(beta)ssnls(beta, X, y);
options2= optimoptions(@lsqnonlin, 'MaxIter', 10000, 'TolFun', ...
    1e-16, 'MaxFunEvals', 10000);
[betanls, fvalnls] = lsqnonlin(@(beta) y-exp(X*beta), beta0, [],[], options2)

%% Q4

[betanlsnm, fvalnlsnm, outputnlsnm] = fminsearch(fnls, beta0, options)

%% Q5
%% Ones
%% NM
load('hw3.mat')

beta1=ones(6, 1);
tic
options=optimset('MaxIter', 10000, 'TolFun', 1e-16, 'MaxFunEvals', 10000);
fnm= @(beta)negll(beta, X, y);
betanm1 = fminsearch(fnm, beta1, options)
toc
```

```
%% QN

tic
betaqn1 = fminunc(@(beta) negll(beta, X, y), beta1)
toc

%% NLS
tic
fnls= @(beta)ssnls(beta, X, y);
options2= optimoptions(@lsqnonlin, 'MaxIter', 10000, 'TolFun', ...
    1e-16, 'MaxFunEvals', 10000);
betanls1 = lsqnonlin(@(beta) y-exp(X*beta), beta1, [],[], options2)
toc

%% NLS via NM
tic
betanlsnm1 = fminsearch(fnls, beta1, options)
toc

%% Twos
beta2=2*ones(6,1);

%% NM
tic
options=optimset('MaxIter', 10000, 'TolFun', 1e-16, 'MaxFunEvals', 10000);
fnm= @(beta)negll(beta, X, y);
betanm2 = fminsearch(fnm, beta2, options)
toc
```

```
%% QN

tic
betaqn2 = fminunc(@(beta) negll(beta, X, y), beta2)
toc

%% NLS
tic
fnls= @(beta)ssnls(beta, X, y);
options2= optimoptions(@lsqnonlin, 'MaxIter', 10000, 'TolFun', 1e-16, ...
    'MaxFunEvals', 10000);
betanls2 = lsqnonlin(@(beta) y-exp(X*beta), beta2, [],[], options2)
toc

%% NLS via NM
tic
betanlsnm2 = fminsearch(fnls, beta2, options)
toc

%% Minus Ones
beta_1=-1*ones(6,1);

%% NM
tic
options=optimset('MaxIter', 10000, 'TolFun', 1e-16, 'MaxFunEvals', 10000);
fnm= @(beta)negll(beta, X, y);
betanm_1 = fminsearch(fnm, beta_1, options)
```

```
toc

%% QN
tic
betaqn_1 = fminunc(@(beta) negll(beta, X, y), beta_1)
toc

%% NLS
tic
fnls= @(beta)ssnls(beta, X, y);
options2= optimoptions(@lsqnonlin, 'MaxIter', 10000, 'TolFun', ...
    1e-16, 'MaxFunEvals', 10000);
betanls_1 = lsqnonlin(@(beta) y-exp(X*beta), beta_1, [],[], options2)
toc

%% NLS via NM
tic
betanlsnm_1 = fminsearch(fnls, beta_1, options)
toc
```