Advanced Econometrics 1

Computer Lab Exercise Week 5

In this computer lab exercise, we analyse some details of numerical optimization of the criterion function of extremum estimators in MATLAB. The exercise is based on Section 5.9 in Cameron & Trivedi, where they consider a dataset of 10000 (simulated) observations on a regressor x and a duration (non-negative) dependent variable y. The data are collected in the text file `Chapter5data.txt`. As explained in Section 5.9, x consists of i.i.d. $\mathcal{N}[1,1]$ draws, and $y|x$ follows the exponential distribution with parameter $\lambda(x) = \exp(\beta_1 + \beta_2 x) = \exp(\mathbf{x}'\boldsymbol{\beta})$, where the true value $\boldsymbol{\beta}_0$ is $(2,-1)'$. One may think of the observations y_i as durations (for example, of an unemployment spell of person i), which are influenced by the person's characteristic x_i. We compare maximum likelihood estimation of the exponential regression model to nonlinear least-squares estimation of the mean function

$$\mathrm{E}[y|x] = \frac{1}{\lambda(x)} = \exp(-\mathbf{x}'\boldsymbol{\beta}).$$

In addition to the data-file mentioned earlier, you will need the MATLAB script file `CompExWeek5.m`, and two MATLAB function files `loglikexp.m` and `rbr.m` (row-by-row multiplication procedure). All these files should be in the same folder on your network drive.

1. Open the file `CompExWeek5.m` using MATLAB (do not double-click the file since that will open it in Mathematica).
   The program first loads the data from the text file, and creates an $N \times 1$ dependent vector $\mathbf{y}$ and a $N \times 2$ regressor matrix $\mathbf{X}$. Next, it prints some simple descriptive statistics of $\mathbf{y}$ to the screen, as well as the results of an OLS regression of $\mathbf{y}$ on $\mathbf{X}$ (which is not a consistent estimator of $\boldsymbol{\beta}_0$). Finally, it estimates the parameters using maximum likelihood using MATLAB's built-in optimiser `fmincon` and the separate log-likelihood function file `loglikexp.m`. Note that `fmincon` minimizes and hence should minimize **minus** the log-likelihood function.
   The estimates are printed along with their standard errors, which are calculated by the square roots of the diagonal elements of either $-N^{-1}\widehat{\mathbf{A}}^{-1}$ ("ML") or $N^{-1}\widehat{\mathbf{A}}^{-1}\widehat{\mathbf{B}}\widehat{\mathbf{A}}^{-1}$ ("QML"), see the slides or the textbook.
   Run the program, and interpret the output:

(a) Have a look at the OLS estimates; they appear to be very far from the true value $\boldsymbol{\beta}_0 = (2, -1)'$, and in fact they have the wrong sign.

i. Give an intuitive explanation of the wrong sign.

ii. Sketch how one could prove formally that the OLS estimator is inconsistent.

(b) Next, consider the ML estimates; check that the two types of standard errors lead to very similar results. Give an explanation based on knowledge of ML properties and/or equalities.

2. We now extend the program, by also estimating $\boldsymbol{\beta}$ by nonlinear least-squares. The criterion function to be maximised is

$$Q_N(\boldsymbol{\beta}) = -\frac{1}{2N}\sum_{i=1}^{N}(y_i - \exp(-\mathbf{x}_i'\boldsymbol{\beta}))^2.$$

(a) Sketch how you could prove that this NLS estimator is consistent.

(b) Derive for $Q_N(\boldsymbol{\beta})$ the gradient vector $\mathbf{h}_N(\boldsymbol{\beta}) = \frac{\partial Q_N(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$ and Hessian matrix $\mathbf{H}_N(\boldsymbol{\beta}) = \frac{\partial^2 Q_N(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}\partial \boldsymbol{\beta}'} = \frac{\partial \mathbf{h}_N(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}'}$.

(c) Estimate β using NLS. You can use the the built-in optimiser `fmincon`. You will have to write a separate function file `nlsexp.m`, giving the NLS criterion function, and possibly the gradient (required by the `optimset('GradObj','on')` statement, which tells the program to use analytical instead of numerical gradients). You can use the `loglikexp.m` as a guide.

3. Extend the program to a Monte Carlo study to compare the ML and NLS estimator for this model. Some suggestions:

- After having loaded `y` and `x` in the beginning of the program, you should draw `y` from the exponential distribution with mean `exp(x-2)`, because $\beta_1 = 2$ and $\beta_2 = -1$, and the mean of the exponential distribution is $\lambda(x)^{-1} = \exp(-\beta_1 - \beta_2 x)$. This is done with the command `y = exprnd(exp(x-2));`. Running the program again will give the estimated for the new simulated y, which can be seen as a Monte Carlo with one replication.
- To get more replications, you would need to put the part of the program that simulates `y` and does the calculations, in a for loop, by adding `for i=1:R` at

the beginning, and `end` at the end (no semicolons). Here `R` is the number of replications, that you should give a value first, e.g., `R = 1000;`.

- You should also first create $R \times 2$ matrices for the ML and NLS estimates in each replication, e.g., by `b_ML = zeros(R,2);` and `b_NLS = zeros(R,2);`. After the estimation process for each of the two estimators, you can then add statements such as `b_ML(i,:) = b'` to save the estimate in the matrix for later use.
- Change the `options` definition to `options = optimset('GradObj','on', 'Display', 'off')`, to avoid intermediate results being printed to the screen. Furthermore, the program will be speeded up by setting the starting value `b0` to the true value, so that we are starting the iterations in the right neighbourhood. This looks like cheating (because we don't know the true value in practical applications), but should not matter for the end result, only for the number of iterations needed and hence speed of the program. This also implies that you can leave out OLS estimation from the program, which gives a very bad starting value anyway (setting `b0` to zero would probably be better), and which again saves a bit of time.
- After running the program, you can calculate the Monte Carlo mean and standard deviation of both estimators (using the `mean` and `std` functions). This should help you in assessing the relative inefficiency of NLS, compared to ML. You can also inspect histograms or estimated densities (see Section 7 of the "Getting Started" document).