Empirical exercise – The likelihood function and the linear regression model

This is a function that returns as output the value of a log-likelihood function, and the value of the gradient at the maximising values of the parameters. Let us call the former `Obj`, and the latter `GradObj`. The function accepts the data for `y` and `X`, and the parameter vector `T_true` as input arguments. `T_true` is a vector of parameters where all but the last element represent the `B_true` vector, and the last element represents the scalar `sigma`. `T_true` is a `N_par` by `1` vector because `B_true` is `N_par_coefficients` by `1`, and `sigma` is a scalar that adds on `N_par_coefficients` by 1.

Take a closer look at `likelihood`. The `normpdf(y-X*B_true,mu,sigma)` function accepts as input arguments the values contained in `y-X*B_true`, an assumed mean value of `mu`, and an assumed standard deviation value of `sigma`. It returns as an output argument the value of the normal density function at each value of `y-X*B_true`. This means that `likelihood` is a `N_obs` by `1` vector because `N_obs` is 50, and each element of the vector represents the likelihood contribution of each individual in the data.

`Obj` takes the logarithm of the product of the individual probability density functions, which is equivalent to the sum of the logarithms of the individual density functions. Hence, `Obj` represents our objective function and returns a scalar value at the optimizing `B_true` and `sigma`. The idea is that we choose as estimates those values of `B_true` and `sigma` that are most consistent with the sample data. `Obj` is multiplied by minus one because `fminunc` is using a minimization and not a maximization algorithm. It uses a minimization algorithm because computational properties are better that way.

Consider the construction of the gradient object `GradObj`. It defines a vector of Not-a-Number with a dimension equal to the dimension of `T_true`. It does not matter if it is defined as a row or a column vector; both are accepted by `fminunc`. `GradObj(1:end-1)` replaces all but the last entry of `GradObj` with the derivative of the log likelihood function with respect to the parameter vector `B_true`. Because there are 4 parameters in `B_true`, `GradObj(1:end-1)` is a 4×1 vector. `GradObj(end)` replaces the last entry of `GradObj` with the derivative of the log likelihood function with respect to the scalar parameter `sigma`, and hence `GradObj(end)` is 1×1. Note that both the `GradObj(1:end-1)` and the `GradObj(end)` are multiplied by minus one for the same reason that the log likelihood function is multiplied.

```
function [Obj,GradObj] = exercisemllrmfun(y,X,T_true,N_obs,N_par)
B_true = T_true(1:end-1);
sigma = T_true(end);
mu = 0;
% The likelihood function
likelihood_N_obs = normpdf(y-X*B_true,mu,sigma);
% The loglikelihood function
loglikelihood_N_obs = log(likelihood_N_obs);
% The objective function
Obj = -sum(loglikelihood_N_obs);
% The gradient of the objective function
GradObj = NaN(N_par,1));
GradObj(1:end-1) = -(1/sigma^2)*X'*(y-X*B_true);
GradObj(end) = (N_obs/sigma)-(1/sigma^3)*(y-X*B_true)'*(y-X*B_true);
```

```
end
```