Written Exam Economics Summer 2020

Introduction to Programming and Numerical Analysis

Date 16th – 18th May 2019

This exam question consists of 6 pages in total.

(An electronic version of the notebook with the exam questions is also available).

Answers only in English.

You should hand-in a single zip-file named with your groupname only. The zip-file should contain:

1. A general README.md for your portfolio
2. A Feedback.txt file with a list of the groups each group member have given peer feedback to
3. Your inaugural project (in the folder /inauguralproject)
4. Your data analysis project (in the folder /dataproject)
5. Your model analysis project (in the folder /modelproject)
6. Your exam project (in the folder /examproject)

In [1]:

```
import numpy as np
```

Linear regression

Consider the following **linear equation:**

$$y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \epsilon_i$$

Assume you have access to data of the **independent variables** ($x_{1,i}$, $x_{2,i}$) and the **dependent variable** (y_i) for N individuals, where i indexes individuals. The variable ϵ_i is a mean-zero **stochastic shock**.

Assume the **data generating process** is given by:

In [2]:

```
def DGP(N):

    # a. independent variables
    x1 = np.random.normal(0,1,size=N)
    x2 = np.random.normal(0,1,size=N)

    # b. errors
    eps = np.random.normal(0,1,size=N)

    extreme = np.random.uniform(0,1,size=N)
    eps[extreme < 0.05] += np.random.normal(-5,1,size=N)[extreme < 0.05]
    eps[extreme > 0.95] += np.random.normal(5,1,size=N)[extreme > 0.95]

    # c. dependent variable
    y = 0.1 + 0.3*x1 + 0.5*x2 + eps

    return x1, x2, y
```

The data you have access to is:

In [3]:

```
np.random.seed(2020)
x1,x2,y = DGP(10000)
```

Question 1: Estimate the vector of coefficients $\beta = (\beta_0, \beta_1, \beta_2)$ using **ordinary least squares (OLS)** implemented with **matrix algebra** by

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

where $\mathbf{X}'$ is the transpose of $\mathbf{X}$ and

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 1 & x_{1,1} & x_{2,1} \\ 1 & x_{1,2} & x_{2,2} \\ \vdots & \vdots & \\ 1 & x_{1,N} & x_{2,N} \end{pmatrix}$$

Question 2: Construct a 3D plot, where the data is plotted as scattered points, and the prediction of the model is given by the plane

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{1,i} + \hat{\beta}_2 x_{2,i}$$

Question 3: Esimtate the vector of coefficients $\beta = (\beta_0, \beta_1, \beta_2)$ using a **numerical solver** to solve the ordinary least square problem, shown below, directly. Compare your results with the matrix algebra results.

$$\min_{\beta} \sum_{i=1}^{N} (y_i - (\beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i}))^2$$

Question 4: Estimate the vector of coefficients $\beta = (\beta_0, \beta_1, \beta_2)$ using **least absolute deviations (LAD)** using a numerical solver to solve the following problem directly:

$$\min_{\beta} \sum_{i=1}^{N} |y_i - (\beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i})|$$

where $|z|$ is the absolute value of z.

Question 5: Set $N = 50$. Repeat the estimation using the **OLS** and **LAD** methods $K = 5000$ times, drawing a new random sample from the data generating process each time. Compare the estimates from each method using histograms. Which method do you prefer? Explain your choice.

Durable purchases

Consider a **household** living in two periods.

In the **second period** it gets utility from **non-durable consumption**, c, and **durable consumption**, $d + \chi x$:

$$\begin{aligned} v_2(m_2, d) &= \max_{c} \frac{(c^{\alpha}(d + \chi x)^{1-\alpha})^{1-\rho}}{1-\rho} \\ & \text{s.t.} \\ & x = m_2 - c \\ & c \in [0, m_2] \end{aligned}$$

where

- m_2 is cash-on-hand in the beginning of period 2
- c is non-durable consumption
- d is pre-commited durable consumption
- $x = m_2 - c$ is extra durable consumption
- $\rho > 1$ is the risk aversion coefficient
- $\alpha \in (0, 1)$ is the utility weight on non-durable consumption
- $\chi \in (0, 1)$ implies that extra durable consumption is *less* valuable than pre-comitted durable consumption
- the second constraint ensures the household *cannot* die in debt

The **value function** $v_2(m_2, d)$ measures the household's value of having m_2 at the beginning of period 2 with precomitted durable consumption of d. The optimal choice of non-durable consumption is denoted $c^*(m_2, d)$. The optimal extra durable consumption function is $x^*(m_2, d) = m_2 - c^*(m_2, d)$.

Define the so-called **end-of-period 1 value function** as:

$$w(a,d) \equiv \beta \mathbb{E}_1 \left[v_2(m_2,d) \right]$$

where

$$\begin{aligned} m_2 &= (1+r)a + y \\ y &= \begin{cases} 1-\Delta & \text{with prob. } \frac{1}{3} \\ 1 & \text{with prob. } \frac{1}{3} \\ 1+\Delta & \text{with prob. } \frac{1}{3} \end{cases} \end{aligned}$$

and

- a is assets at the end of period 1
- $\beta > 0$ is the discount factor
- $\mathbb{E}_1$ is the expectation operator conditional on information in period 1
- y is income in period 2
- $\Delta \in (0,1)$ is the level of income risk (mean-preserving)
- r is the return on savings

In the **first period**, the household chooses it's pre-comitted level of durable consumption for the next-period,

$$\begin{aligned} v_1(m_1) &= \max_d w(a,d) \\ &\text{s.t.} \\ a &= m_1 - d \\ d &\in [0, m_1] \end{aligned}$$

where m_1 is cash-on-hand in period 1. The second constraint ensures the household *cannot* borrow. The **value function** $v_1(m_1)$ measures the household's value of having m_1 at the beginning of period 1. The optimal choice of pre-committed durable consumption is denoted $d^*(m_1)$.

The **parameters** and **grids** for m_1, m_2 and d should be:

In [4]:

```
# a. parameters
rho = 2
alpha = 0.8
beta = 0.96
r = 0.04
Delta = 0.25

# b. grids
m1_vec = np.linspace(1e-8,10,100)
m2_vec = np.linspace(1e-8,10,100)
d_vec = np.linspace(1e-8,5,100)
```

Question 1: Find and plot the functions $v_2(m_2,d)$, $c^*(m_2,d)$, and $x^*(m_2,d)$. Comment.

Question 2: Find and plot the functions $v_1(m_1)$ and $d^*(m_1)$. Comment.

Hint: For interpolation of $v_2(m_2,d)$ consider using `interpolate.RegularGridInterpolator([GRID-VECTOR1,GRID-VECTOR2],VALUE-MATRIX,bounds_error=False,fill_value=None)`.

Next, consider an **extension** of the model, where there is also a **period 0**. In this period, the household makes a choice whether to stick with the level of durables it has, $z = 0$, or adjust its stock of durables, $z = 1$. If adjusting, the household loses a part of the value of its durable stock; more specificaly it incurs a proportional loss of $\Lambda \in (0, 1)$.

Mathematically, the **household problem in period 0** is:

$$v_0(m_0, d_0) = \max_{z \in \{0,1\}} \begin{cases} w(m_0, d_0) & \text{if } z = 0 \\ v_1(m_0 + (1 - \Lambda)d_0) & \text{if } z = 1 \end{cases}$$

The **parameters** and **grids** for m_0 and d_0 should be:

In [5]:

```
Lambda = 0.2
m0_vec = np.linspace(1e-8,6,100)
d0_vec = np.linspace(1e-8,3,100)
```

Question 3: For which values of m_0 and d_0 is the optimal choice not to adjust, i.e. $z = 0$? Show this in a plot. Give an interpretion of your results.

Gradient descent

Let $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ be a two-dimensional vector. Consider the following algorithm:

Algorithm: `gradient_descent()`

Goal: Minimize the function $f(\boldsymbol{x})$.

1. Choose a tolerance $\epsilon > 0$, a scale factor $\Theta > 0$, and a small number $\Delta > 0$
2. Guess on $\boldsymbol{x}_0$ and set $n = 1$
3. Compute a numerical approximation of the jacobian for f by

$$\nabla f(\boldsymbol{x}_{n-1}) \approx \frac{1}{\Delta} \begin{bmatrix} f\left(\boldsymbol{x}_{n-1} + \begin{bmatrix} \Delta \\ 0 \end{bmatrix}\right) - f(\boldsymbol{x}_{n-1}) \\ f\left(\boldsymbol{x}_{n-1} + \begin{bmatrix} 0 \\ \Delta \end{bmatrix}\right) - f(\boldsymbol{x}_{n-1}) \end{bmatrix}$$

4. Stop if the maximum element in $|\nabla f(\boldsymbol{x}_{n-1})|$ is less than ϵ
5. Set $\theta = \Theta$
6. Compute $f_n^\theta = f(\boldsymbol{x}_{n-1} - \theta \nabla f(\boldsymbol{x}_{n-1}))$
7. If $f_n^\theta < f(\boldsymbol{x}_{n-1})$ continue to step 9
8. Set $\theta = \frac{\theta}{2}$ and return to step 6
9. Set $x_n = x_{n-1} - \theta \nabla f(\boldsymbol{x}_{n-1})$
10. Set $n = n + 1$ and return to step 3

Question: Implement the algorithm above such that the code below can run.

Optimizer function:

In [6]:

```
def gradient_descent(f,x0,epsilon=1e-6,Theta=0.1,Delta=1e-8,max_iter=10_000):
    pass
```

Test case:

In [7]:

```
def rosen(x):
    return (1.0-x[0])**2+2*(x[1]-x[0]**2)**2

x0 = np.array([1.1,1.1])
try:
    x,it = gradient_descent(rosen,x0)
    print(f'minimum found at ({x[0]:.4f},{x[1]:.4f}) after {it} iterations')
    assert np.allclose(x,[1,1])
except:
    print('not implemented yet')
```

```
not implemented yet
```