MKT 7317 Problem Set 1

Due date: 1/30 (Sun), 23:59 CST
Updated December 28, 2021

Before getting into start working on this problem set, among other things, please get yourselves familiarized with (i) how to use @jit of the numba library, and (ii) how to parallelize your code in Python using `joblib`:

```
from joblib import Parallel, delayed
import time
def f(x):
    time.sleep(2)
    return x**2
results = Parallel(n_jobs=8)(delayed(f)(i) for i in range(10))
```

And note that `joblib` deals with the variables accessed in the parallelized processes slightly differently in the usual loops, so be sure to read the documentation carefully.

1 (Python) Calculating the OLS Estimator for a Multiple Regression

Note. You should submit the python codes that are used to generate the results. Submission without codes will get zero credit. Also, in this question, you are NOT allowed to use any automated commands/packages.

- (a) Load "cps09mar.csv" into memory (use either polars or pandas)
- (b) Generate a column of ones, and generate a variable `log_wage` by taking natural logarithm of the variable `earnings`.
- (c) Extract the data matrices $\mathbf{y}$, $\mathbf{X}$ as numpy array.

Now Consider the population regression function:

$$y_i = \mathbf{x}_i'\boldsymbol{\beta} + u_i,$$

where y_i corresponds to the `log_wage` variable and $\mathbf{x}_i$ corresponds to a constant and `female`, `education`, `hours` variables in the data.

- (d) Find the sample size n of this data and column rank k of the matrix $\mathbf{X}$s
- (e) Calculate the OLS estimator $\hat{\boldsymbol{\beta}}$ for $\boldsymbol{\beta}$ using the usual closed-form formula $\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$
- (f) Calculate the prediction $\hat{y}_i = \mathbf{x}_i'\hat{\boldsymbol{\beta}}$ for each given sample point $\mathbf{x}_i$. Save the prediction $\hat{\mathbf{y}} = (\hat{y}_1, ..., \hat{y}_n)'$ in memory for future use in the following steps.
- (g) Calculate the residual $\hat{u}_i := y_i - \mathbf{x}_i'\hat{\boldsymbol{\beta}}$ for each given sample point $\mathbf{x}_i$. Save the residual vector $\hat{\mathbf{u}} = (\hat{u}_1, ..., \hat{u}_n)'$ in memory for future use in the following steps.
- (h) Calculate the standard error of the regression $\sqrt{s^2}$ using only basic arithmetic operations and basic matrix operations.
- (i) Calculate $Est.Var\left(\hat{\boldsymbol{\beta}} | \{\mathbf{x}_k\}_{k=1}^n\right)$, the estimated covariance matrix of $\hat{\boldsymbol{\beta}}$ given data, using only basic arithmetic operations and basic matrix operations.
- (j) Calculate the Total Sum of Squares (TSS), Explained Sum of Squares (ESS), Residual Sum of Squares (RSS) using only basic arithmetic operations and basic matrix operations.
- (k) Calculate R^2. Verify that the following is true:

$$\frac{ESS}{TSS} = 1 - \frac{RSS}{TSS}.$$

- (l) Numerically calculate $\sum_{i=1}^n \hat{u}_i$ and $\sum_{i=1}^n \hat{u}_i\mathbf{x}_i$. Compare your results with the theoretical results.
- (m) Construct the t-statistic corresponding to $H_0 : \mathbf{c}'\boldsymbol{\beta} = 0$ vs. $H_1 : \mathbf{c}'\boldsymbol{\beta} \neq 0$ for the following $\mathbf{c}$s, respectively, and calculate the corresponding p-values.

$$\mathbf{c} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \mathbf{c} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \mathbf{c} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \mathbf{c} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

- (n) Construct the t-statistic corresponding to $H_0 : \mathbf{c}'\boldsymbol{\beta} = 1$ vs. $H_1 : \mathbf{c}'\boldsymbol{\beta} \neq 1$ for $\mathbf{c} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$ and calculate the corresponding p-value.

- (o) Now consider minimizing the objective function numerically using IPOPT:

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(y_i - \mathbf{x}_i'\boldsymbol{\beta}\right)^2 .$$

  Code up the objective function, its gradient, and hessian as separate functions within a class. Solve the following problem numerically with cyIPOPT by (i) supplying only objective function and gradient to cyIPOPT (`ipoptproblem.add_option('hessian_approximation', 'limited-memory')`) and (ii) supplying objective function, gradient, hessian althogether respectively (`ipoptproblem.add_option('hessian_approximation', 'exact')`). Compare the results with (e). Do they coincide? (hint. vecotrize and matricize the problem will make your lives simpler)

2 (Python) (Weak) Law of Large Numbers

- (a) Consider a continuous random variable $X_i \sim Uniform\left[0, 2\right]$. What is $E\left[X_i\right]$ and $Var\left(X_i\right)$?

- (b) Consider the X_i defined above in (a) for $i = 1, 2, ..., n$, where each $X_i \perp\!\!\!\perp X_j$ whenever $i \neq j$. Consider the sample mean

$$\bar{X}_n := \frac{1}{n}\sum_{i=1}^{n} X_i.$$

  What is $E\left[\bar{X}_n\right]$ and $Var\left(\bar{X}_n\right)$? (The answer might be a function of n).

Now repeat the following (c)-(e) for $n = 1, 2, 3, 5, 10, 50, 100, 1000, 3000$. Parallelize your routine for (c)-(e) using joblib.

- (c) (Python) Generate a size n vector of independent $Uniform\left[0, 2\right]$ random variables and calculate its sample mean $\bar{X}_n$.

- (d) (Python) Take $\left|\bar{X}_n - E\left[X_i\right]\right|$ and report the value.

- (e) (Python) Now consider a continuous transformation $f\left(x\right) = 2x^2 - 5x + 1 + \frac{1}{3x}$. Take

$$\left|f\left(\bar{X}_n\right) - f\left(E\left[X_i\right]\right)\right|$$

and report the value.

- (f) What happens to the reported value in (d) and (e) as n increases? Discuss.

3 (Python) The Central Limit Theorem

- (a) Consider a continuous random variable $X_i \sim Uniform\,[0,2]$. What is $E\,[X_i]$ and $Var\,(X_i)$?

- (b) Consider the X_i defined above in (a) for $i = 1, 2, ..., n$, where each $X_i \perp\!\!\!\perp X_j$ whenever $i \neq j$. Consider the sample mean

$$\bar{X}_n := \frac{1}{n}\sum_{i=1}^{n} X_i.$$

  What is $E\,[\bar{X}_n]$ and $Var\,(\bar{X}_n)$? (The answer would be a function of n).

- (c) Consider the transformation

$$Y_n := \sqrt{n}\left(\bar{X}_n - E\,[X_i]\right).$$

  What is $E\,[Y_n]$ and $Var\,(Y_n)$?

- (d) Consider the transformation

$$Z_n := \sqrt{n}\frac{\left(\bar{X}_n - E\,[X_i]\right)}{\sqrt{Var\,(X_i)}}.$$

  What is $E\,[Z_n]$ and $Var\,(Z_n)$?

Now repeat the following for $n = 1, 2, 3, 5, 10, 50, 100, 1000, 3000$. Parallelize your routine for (e)-(m) using joblib.

- (e) (Python) Generate $t = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, ..., 2500$ $n \times 1$ vectors of independent $Uniform\,[0,2]$ random variables and calculate its sample mean $\bar{X}_n^t$ respectively for each t. Denote this size 2500 vector as

$$\mathbf{v}_n^{2500} \equiv \left(\bar{X}_n^1, \bar{X}_n^2, ..., \bar{X}_n^{2500}\right)$$

  for now.

- (f) (Python) Calculate the mean and variance of $\mathbf{v}_n^{2500}$ and report.

- (g) (Python) Recall that each element of $\mathbf{v}_n^{2500}$ is composed of $\bar{X}_n^t$ for $t = 1, ..., 2500$. Now, for each $t = 1, 2, ..., 2500$, take the transformation

$$Y_n^t := \sqrt{n}\left(\bar{X}_n^t - E\left[X_i\right]\right)$$

  and denote the transformed vector as

$$\mathbf{y}_n^{2500} \equiv \left(Y_n^1, Y_n^2, ..., Y_n^{2500}\right).$$

  That is, subtract the $E\left[X_i\right]$ (that you calculated in (a)) from each element of $\mathbf{v}_n^{2500}$, and then multiply it by $\sqrt{n}$, and then denote it by $\mathbf{y}_n^{2500}$.

- (h) (Python) Calculate the mean and variance of $\mathbf{y}_n^{2500}$ and report.

- (i) (Python) Plot the histogram of $\mathbf{y}_n^{2500}$ and report.

- (j) (Python) Recall that each element of $\mathbf{v}_n^{2500}$ is composed of $\bar{X}_n^t$ for $t = 1, ..., 2500$. Now, take the transformation

$$Z_n^t := \sqrt{n}\frac{\left(\bar{X}_n^t - E\left[X_i\right]\right)}{\sqrt{Var\left(X_i\right)}}$$

  and denote the transformed vector as $\mathbf{z}_n^{2500}$. That is, subtract the $E\left[X_i\right]$ (that you calculated in (a)) from each element of $\mathbf{v}_n^{2500}$, then multiply it by $\sqrt{n}$, and divide it by $\sqrt{Var\left(X_i\right)}$ ($Var\left(X_i\right)$ you calculated in (a)), and then denote it by

$$\mathbf{z}_n^{2500} \equiv \left(Z_n^1, Z_n^2, ..., Z_n^{2500}\right).$$

- (k) (Python) Calculate the mean and variance of $\mathbf{z}_n^{2500}$ and report.

- (l) (Python) Plot the histogram of $\mathbf{z}_n^{2500}$ and report.

- (m) What happens to the reported values in (f), (h), (k) and histograms in (i) and (l) as n increases? Discuss.

4 (Python) WLLN with Simple Regression

In this exercise, you will generate datasets for simple regression yourself, and then try to estimate the model parmaters to examine the properties of simple regression OLS estimators as the sample size n grows.

Repeat the following for $n = 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30, 50, 75, 100, 250, 500, 1000, 2000, 3000$. Parallelize your routine for (a)-(e) using joblib.

- (a) (Python) Generate a $n \times 1$ column vector of $Uniform[0, 12]$ random variable and denote it as x.
- (b) (Python) Generate a $n \times 1$ column vector of $Uniform\,[-4, 4]$ random variable and denote it as u.
- (c) (Python) Generate the y vector using the following formula:

  $$y_i = 3 + 2x_i + u_i$$

  for each $i = 1, 2, 3, ..., n$. That is, i'th row of x and u corresponds to i'th observation.
- (d) (Python) Now you have a Monte-Carlo dataset of size n. Estimate the β in the following model

  $$y_i = \alpha + \beta x_i + u_i$$

  using OLS. (Recall the formula $\frac{\widehat{Cov}(x_i, y_i)}{\widehat{Var}(x_i)}$.) What is the calculated value of $\hat{\beta}_{OLS,n}$? Report.
- (e) What happens to $\left|\hat{\beta}_{OLS,n} - 2\right|$ as n increases? Discuss.

5 (Python) CLT with Simple Regression

In this exercise, you will generate datasets for simple regression yourself, and then try to estimate the model parmaters to examine the properties of simple regression OLS estimators as the sample size n grows.

Repeat the following for $n = 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30, 50, 75, 100, 250, 500, 1000, 2000, 3000$. Parallelize your routine using joblib.

- (a) (Python) Generate a $n \times 1$ column vector of $Uniform\,[0, 12]$ random variable and denote it as x.
- (b) (Python) Generate a $n \times 1$ column vector of $Uniform\,[-4, 4]$ random variable and denote it as u.
- (c) (Python) Generate the y vector using the following formula:

  $$y_i = 3 + 2x_i + u_i$$

  for each $i = 1, 2, 3, ..., n$. That is, i'th row of x and u corresponds to i'th observation.
- (d) (Python) Now you have a Monte-Carlo dataset of size n. Estimate the β in the following model

  $$y_i = \alpha + \beta x_i + u_i$$

using OLS. (Recall the formula $\frac{\widehat{Cov}(x_i,y_i)}{\widehat{Var}(x_i)}$.) Save it in the memory.

- (e) (Python) Repeat (a)-(d) for 2,500 times. You must have 2,500 $\hat{\beta}_{OLS,n}$ estimates in the memory at the end of this sub-question. Denote this size 2,500 vector by $\mathbf{b} = \left(\hat{\beta}^1_{OLS,n}, \hat{\beta}^2_{OLS,n}, ..., \hat{\beta}^{2500}_{OLS,n}\right.$

- (f) (Python) Calculate the variance of **b** and report.

- (g) (Python) Subtract 2 from **b** and multiply $\sqrt{n}$ on each element of **b** and denote this as $\mathbf{c} = (c_1, c_2, ..., c_{2500})$, i.e.,

$$c_i = \sqrt{n}\left(\hat{\beta}^i_{OLS,n} - 2\right).$$

  Draw the histogram of **c** and report the histogram.

- (h) What happens to the reported values in (f) and the histogram in (g) as n grows large? Discuss.