Grant Jackson - Homework 3

September 11, 2024

0.1 Least Squares Estimation using Matrix Formula and Numerical Optimization

- Lecture note 2: p.19

[1]:
```
import os
os.chdir('C:\\Users\gmoor\Documents\Economic Analytics 1\Data')
```

[2]:
```
import numpy as np
import pandas as pd
import math

raw0 = pd.read_csv('College.csv')
raw0['Private']=pd.get_dummies(raw0['Private'],drop_first=True, dtype=float)
raw0.head()
```

[2]:
```
                     Unnamed: 0  Private  Apps  Accept  Enroll  Top10perc  \
0  Abilene Christian University      1.0  1660    1232     721         23
1            Adelphi University      1.0  2186    1924     512         16
2                Adrian College      1.0  1428    1097     336         22
3           Agnes Scott College      1.0   417     349     137         60
4     Alaska Pacific University      1.0   193     146      55         16

   Top25perc  F.Undergrad  P.Undergrad  Outstate  Room.Board  Books  Personal  \
0         52         2885          537      7440        3300    450      2200
1         29         2683         1227     12280        6450    750      1500
2         50         1036           99     11250        3750    400      1165
3         89          510           63     12960        5450    450       875
4         44          249          869      7560        4120    800      1500

   PhD  Terminal  S.F.Ratio  perc.alumni  Expend  Grad.Rate
0   70        78       18.1           12    7041         60
1   29        30       12.2           16   10527         56
2   53        66       12.9           30    8735         54
3   92        97        7.7           37   19016         59
4   76        72       11.9            2   10922         15
```

0.1.1 1) Least Squares Estimation using Matrix Algebra

https://www.fsb.miamioh.edu/lij14/411_note_matrix.pdf

[3]:
```
# convert the dataframe to a numpy array (excluding the first column-college
↪names)
raw00 = raw0.iloc[:,1:].values
```

- We can suppress scientific notation using "np.set_printoptions" (e.g. np.set_printoptions(precision=2, suppress=True))
- Scientific notation: https://en.wikipedia.org/wiki/Scientific_notation
- np.set_printoptions: https://numpy.org/doc/stable/reference/generated/numpy.set_printoptions.html

[4]:
```
raw00
```

[4]:
```
array([[1.0000e+00, 1.6600e+03, 1.2320e+03, …, 1.2000e+01, 7.0410e+03,
         6.0000e+01],
        [1.0000e+00, 2.1860e+03, 1.9240e+03, …, 1.6000e+01, 1.0527e+04,
         5.6000e+01],
        [1.0000e+00, 1.4280e+03, 1.0970e+03, …, 3.0000e+01, 8.7350e+03,
         5.4000e+01],
        …,
        [1.0000e+00, 2.0970e+03, 1.9150e+03, …, 2.0000e+01, 8.3230e+03,
         4.9000e+01],
        [1.0000e+00, 1.0705e+04, 2.4530e+03, …, 4.9000e+01, 4.0386e+04,
         9.9000e+01],
        [1.0000e+00, 2.9890e+03, 1.8550e+03, …, 2.8000e+01, 4.5090e+03,
         9.9000e+01]])
```

[5]:
```
np.set_printoptions(precision=3, suppress=True)
```

[6]:
```
print(raw00)
```

```
[[    1.  1660.  1232. …    12.  7041.    60.]
 [    1.  2186.  1924. …    16. 10527.    56.]
 [    1.  1428.  1097. …    30.  8735.    54.]
 …
 [    1.  2097.  1915. …    20.  8323.    49.]
 [    1. 10705.  2453. …    49. 40386.    99.]
 [    1.  2989.  1855. …    28.  4509.    99.]]
```

[7]:
```
# Construct X matrix
# X=raw00[:,[4,0,8,11,16]] # select predictors (note that the first column was
↪removed)
# nrow = X.shape[0]
# intcpt = np.ones( (nrow,1), ) # create an intercept
# X = np.concatenate((intcpt, X), axis=1) # add the intercept to X (i.e X =
↪[intcpt,X] )
```

[8]:
```
# Construct X matrix
X=raw00[:,[4,0,8,11,16]] # select predictors (note that the first column was␣
↪removed)
```

[9]:
```
X
```

[9]:
```
array([[   23.,      1.,  7440.,  2200.,  7041.],
       [   16.,      1., 12280.,  1500., 10527.],
       [   22.,      1., 11250.,  1165.,  8735.],
       …,
       [   34.,      1.,  6900.,   781.,  8323.],
       [   95.,      1., 19840.,  2115., 40386.],
       [   28.,      1.,  4990.,  1250.,  4509.]])
```

[10]:
```
nrow = X.shape[0] # or can use len(x)
```

[11]:
```
intcpt = np.ones( (nrow,1), ) # create an intercept
```

[12]:
```
X = np.concatenate((intcpt, X), axis=1) # add the intercept to X (i.e X =␣
↪[intcpt,X] )
```

[ ]:
```

```

[13]:
```
# Construct Y vector
Y=raw00[:,[15]]
# raw00[:,15] returns a one-dimensional vector, and raw00[:,[15]] returns a␣
↪two-dimensional "column" vector. Y should be the latter.
```

i) Compute LS estimates $\hat{\beta} = (X'X)^{-1}X'Y$

- inv( ) from numpy.linalg
- transpose function
- matrix multiplication

[14]:
```
from numpy.linalg import inv
OLSres = inv(X.T@X)@(X.T@Y) # X.T means X(prime), @ = matrix multiplication
print(OLSres)
```

```
[[ 7.943]
 [ 0.178]
 [ 4.864]
 [ 0.001]
 [-0.002]
 [-0.   ]]
```

[15]:
```
# Compare the results to the previous result obtained from the statsmodels␣
↪package
import statsmodels.formula.api as smf
```

```python
raw0.rename(columns = {'perc.alumni':'palumni'}, inplace = True) # Changing the␣
↪column name from perc.alumni to palumni

# Fit a regression model
OLSres2 = smf.ols('palumni ~ Top10perc + Private + Outstate + Personal +␣
↪Expend', data=raw0).fit()
print(OLSres2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                palumni   R-squared:                       0.387
Model:                            OLS   Adj. R-squared:                  0.383
Method:                 Least Squares   F-statistic:                     97.42
Date:                Wed, 11 Sep 2024   Prob (F-statistic):           1.44e-79
Time:                        19:27:59   Log-Likelihood:                -2867.5
No. Observations:                 777   AIC:                             5747.
Df Residuals:                     771   BIC:                             5775.
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      7.9430      1.427      5.567      0.000       5.142      10.744
Top10perc      0.1778      0.027      6.512      0.000       0.124       0.231
Private        4.8641      0.978      4.972      0.000       2.944       6.784
Outstate       0.0009      0.000      6.198      0.000       0.001       0.001
Personal      -0.0022      0.001     -4.025      0.000      -0.003      -0.001
Expend     -1.029e-05      0.000     -0.100      0.920      -0.000       0.000
==============================================================================
Omnibus:                       16.092   Durbin-Watson:                   2.028
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               16.514
Skew:                           0.344   Prob(JB):                     0.000259
Kurtosis:                       3.195   Cond. No.                     6.40e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 6.4e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[16]:
```python
# Calculate Residuals
resid = Y-X@OLSres
# Calculate SER (Standard error of the regression)
SER = (resid.T@resid)/(nrow-X.shape[1])
# Calculate SE
```

```
SE = np.sqrt(np.diag(SER*inv(X.T@X))).reshape((X.shape[1],1)) # Compare this to
↪the previous result from the statsmodels package
```

[17]:
```
SE
```

[17]:
```
array([[1.427],
       [0.027],
       [0.978],
       [0.   ],
       [0.001],
       [0.   ]])
```

[18]:
```
# Calculate T statistics
Tstat = OLSres/SE
```

[19]:
```
Tstat
```

[19]:
```
array([[ 5.567],
       [ 6.512],
       [ 4.972],
       [ 6.198],
       [-4.025],
       [-0.1  ]])
```

[20]:
```
from scipy import optimize
```

[21]:
```
# Define loss fn in two ways

# loss function 1
#def loss(inpt,Y,X):
#    nrow=Y.shape[0]
#    inpt = inpt.reshape((-1,1))
#     loss0=0

#     for i in range(0,nrow):

#             resid = Y[i,:]-X[i,:]@inpt
#             loss0 = loss0+resid*resid
            # can be done simply: loss0+=resid*resid (add and assign)

#     return loss0

# loss function 2
def loss2(inpt,Y,X):
    inpt = inpt.reshape((-1,1))

    resid = Y-X@inpt
```

```
    loss0 = (Y-X@inpt).T@(Y-X@inpt)

    return loss0
```

- Optimizer "fmin" : https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fmin.html

[22]:
```
# Optimize: starting values, stopping rules should be specified by the user
inpt0 = np.zeros((X.shape[1],1)) # starting value
OLSres2=optimize.fmin(loss2,
                      inpt0,
                      args=(Y,X),
                      maxfun=40000,
                      maxiter=40000,
                      ftol=1e-10,
                      xtol=1e-10,
                      disp=True
                     )
```

```
Optimization terminated successfully.
         Current function value: 73023.898799
         Iterations: 1486
         Function evaluations: 2368
```

[23]:
```
print(OLSres2)
```

```
[ 7.943  0.178  4.864  0.001 -0.002 -0.   ]
```

0.1.2 HW3

- Pick one of your linear regression specifications in HW2
- Compute least squares estimates, standard errors of the estimates and t-statistics using the matrix formula and optimization algorithm as described above
- Compare them to the results obtained previously from the statsmodels package

[24]:
```
raw0.head()
```

[24]:
```
                     Unnamed: 0  Private  Apps  Accept  Enroll  Top10perc  \
0  Abilene Christian University      1.0  1660    1232     721         23
1            Adelphi University      1.0  2186    1924     512         16
2                Adrian College      1.0  1428    1097     336         22
3           Agnes Scott College      1.0   417     349     137         60
4     Alaska Pacific University      1.0   193     146      55         16

   Top25perc  F.Undergrad  P.Undergrad  Outstate  Room.Board  Books  Personal  \
0         52         2885          537      7440        3300    450      2200
1         29         2683         1227     12280        6450    750      1500
2         50         1036           99     11250        3750    400      1165
3         89          510           63     12960        5450    450       875
4         44          249          869      7560        4120    800      1500
```

```
   PhD  Terminal  S.F.Ratio  palumni  Expend  Grad.Rate
0   70        78       18.1       12    7041         60
1   29        30       12.2       16   10527         56
2   53        66       12.9       30    8735         54
3   92        97        7.7       37   19016         59
4   76        72       11.9        2   10922         15
```

[25]:
```
# Index: PAlumni=15, Top10perc=4, Outstate=8, Private=0, Apps=1

# Construct X matrix
X_hw = raw00[:, [4, 8, 0, 1]] # select predictors
nrow = X_hw.shape[0]
intcpt = np.ones((nrow, 1)) # create an intercept
X_hw = np.concatenate((intcpt, X_hw), axis=1) # add the intercept to x

# Construct Y vector
Y_hw = raw00[:, [15]]
```

[26]:
```
# Least Squares Estimation using Matrix Algebra
from numpy.linalg import inv
OLSres_hw = inv(X_hw.T@X_hw)@(X_hw.T@Y_hw) # X.T means X(prime), @ = matrix␣
 ↪multiplication
OLSres_hw
```

[26]:
```
array([[ 5.233],
       [ 0.209],
       [ 0.001],
       [ 2.775],
       [-0.001]])
```

[27]:
```
# Calculate Residuals
resid_hw = Y_hw-X_hw@OLSres_hw
# Calculate SER (Standard error of the regression)
SER_hw = (resid_hw.T@resid_hw)/(nrow-X_hw.shape[1])
# Calculate SE
SE_hw = np.sqrt(np.diag(SER_hw*inv(X_hw.T@X_hw))).reshape((X_hw.shape[1],1)) #␣
 ↪Compare this to the previous result from the statsmodels package

SE_hw
```

[27]:
```
array([[1.025],
       [0.026],
       [0.   ],
       [1.124],
       [0.   ]])
```

[28]:
```
# Calculate T statistics
Tstat_hw = OLSres_hw/SE_hw

Tstat_hw
```

[28]:
```
array([[ 5.106],
       [ 8.102],
       [ 8.448],
       [ 2.469],
       [-4.65 ]])
```

[29]:
```
# Least Squares Estimation using Numerical Optimization
def loss_hw(inpt_hw, Y_hw, X_hw):
    inpt_hw = inpt_hw.reshape((-1, 1))
    resid_hw = Y_hw - X_hw @ inpt_hw
    loss0_hw = (Y_hw - X_hw @ inpt_hw).T @ (Y_hw - X_hw @ inpt_hw)
    return loss0_hw
```

[30]:
```
# Optimize
inpt0_hw = np.zeros((X_hw.shape[1], 1))  # starting value
OLSres2_hw = optimize.fmin(loss_hw,
                        inpt0_hw,
                        args=(Y_hw, X_hw),
                        maxfun=40000,
                        maxiter=40000,
                        ftol=1e-10,
                        xtol=1e-10,
                        disp=True)
```

```
Optimization terminated successfully.
         Current function value: 72551.471860
         Iterations: 1035
         Function evaluations: 1713
```

[31]:
```
print(OLSres2_hw)
```

```
[ 5.233  0.209  0.001  2.775 -0.001]
```

[32]:
```
# 3) Compare with statsmodels
model = smf.ols('palumni ~ Top10perc + Outstate + Private + Apps', data=raw0).
 ↪fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                palumni   R-squared:                       0.391
Model:                            OLS   Adj. R-squared:                  0.388
Method:                 Least Squares   F-statistic:                     124.0
Date:                Wed, 11 Sep 2024   Prob (F-statistic):           1.01e-81
```

```
Time:                        19:28:00   Log-Likelihood:                -2865.0
No. Observations:                 777   AIC:                             5740.
Df Residuals:                     772   BIC:                             5763.
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      5.2335      1.025      5.106      0.000       3.222       7.245
Top10perc      0.2086      0.026      8.102      0.000       0.158       0.259
Outstate       0.0011      0.000      8.448      0.000       0.001       0.001
Private        2.7749      1.124      2.469      0.014       0.569       4.981
Apps          -0.0005      0.000     -4.650      0.000      -0.001      -0.000
==============================================================================
Omnibus:                       16.226   Durbin-Watson:                   2.028
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               16.657
Skew:                           0.348   Prob(JB):                     0.000242
Kurtosis:                       3.176   Cond. No.                     4.02e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 4.02e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[ ]: