3. Linear Regression, Prediction

November 20, 2020

0.1 3. Linear Regression: Prediction

In this exercise we will apply ordinary least squares regression and the lasso estimator to the Oregon Health data set.

[59]:
```
import csv
import pandas as pd
# You may have to adjust the path here
Oregdata = pd.read_csv('Oregon.csv', sep=',', na_values=".")
Oregdata.head()
```

[59]:
```
   household_id  treatment  english_list  female_list  zip_msa  weight_12m  \
0        100002          1             1            1        1         1.0
1        100005          1             0            1        1         1.0
2        100006          1             1            0        1         1.0
3        100009          0             1            1        1         1.0
4        100013          1             1            0        0         1.0

   docvis  hhinc_pctfpl_12m  race_hisp_12m  race_white_12m  …  \
0       0         60.054909              0               1  …
1       0        129.710540              1               0  …
2       1         34.134354              0               1  …
3       1         47.788094              0               1  …
4      10         11.542013              0               1  …

   ddddraXnum_3_3  ddddraXnum_4_2  ddddraXnum_5_2  ddddraXnum_6_2  \
0               0               0               0               0
1               0               0               0               0
2               0               0               0               0
3               0               0               0               0
4               0               0               0               0

   ddddraXnum_7_2  edu_12m_2  edu_12m_3  edu_12m_4  age2008  chronicdis
0               0          1          0          0       24           0
1               0          1          0          0       39           1
2               0          0          0          1       62           3
3               0          1          0          0       31           0
4               0          1          0          0       45           2
```

```
[5 rows x 32 columns]
```

We are interested in predicting the number of doctor visits. Thus, our regression model is

$$DOCVIS_i = \beta_1 * TREATMENT_i + X_i'\beta + \varepsilon_i,$$

where $DOCVIS_i$ is the number of doctor visits, $TREATMENT_i$ is access to health insurance. The controls X_i contain the variables:

dddraw_sur_2 ddddraw_sur_3 ddddraw_sur_4 ddddraw_sur_5 ddddraw_sur_6 ddddraw_sur_7 dddnumhh_li_2 dddnumhh_li_3 ddddraXnum_2_2 ddddraXnum_2_3 ddddraXnum_3_2 ddddraXnum_3_3 ddddraXnum_4_2 ddddraXnum_5_2 ddddraXnum_6_2 ddddraXnum_7_2

1. Split the data set randomly into a train (70%) and test (30%) sample. When you split the sample, please use `random_state=145`. Explain in your own words: Why do we need sample splitting?
2. Implement a function `MSE` that calculates the mean squared error for a given pair of vectors $\hat{y}$ and y.
3. A colleague is convinced that the best prediction you can make is 1, i.e., $\hat{y}_i = 1$, for $i, \ldots, n$. Calculate the train and test mean squared error based on the sample split in part 1.
4. Use the two different model specification from Exercise 2.2. and 2.4. of Problem Set 3 and calculate the test and train mean squared error using ols regression. Summarize your results.
5. Repeat exercise 4. using lasso regression instead of ols regression (Hint: *linear_model.LassoCV()*). Compare the performance to the results with ols regression. What model specification/ estimation procedure gives the best test mean squared error? Illustrate your results in a table/figure!
6. The function *linear_model.LassoCV()* includes the parameter *cv* to apply k-fold cross-validation. Explain in your own words: Why do we need k-fold cross-validation and how does it work?

[250]:
```python
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

[251]:
```python
#load

import csv
import pandas as pd
Oregdata = pd.read_csv('Oregon.csv', sep=',', na_values=".")
Oregdata.head()
```

[251]:
```
   household_id  treatment  english_list  female_list  zip_msa  weight_12m  \
0        100002          1             1            1        1         1.0
1        100005          1             0            1        1         1.0
2        100006          1             1            0        1         1.0
3        100009          0             1            1        1         1.0
4        100013          1             1            0        0         1.0
```

```
   docvis  hhinc_pctfpl_12m  race_hisp_12m  race_white_12m  …  \
0       0         60.054909              0               1  …
1       0        129.710540              1               0  …
2       1         34.134354              0               1  …
3       1         47.788094              0               1  …
4      10         11.542013              0               1  …

   ddddraXnum_3_3  ddddraXnum_4_2  ddddraXnum_5_2  ddddraXnum_6_2  \
0               0               0               0               0
1               0               0               0               0
2               0               0               0               0
3               0               0               0               0
4               0               0               0               0

   ddddraXnum_7_2  edu_12m_2  edu_12m_3  edu_12m_4  age2008  chronicdis
0               0          1          0          0       24           0
1               0          1          0          0       39           1
2               0          0          0          1       62           3
3               0          1          0          0       31           0
4               0          1          0          0       45           2

[5 rows x 32 columns]
```

[252]:
```
#pre-process

type(Oregdata)
y =Oregdata["docvis"]
print(len(y))
#trtmt =Oregdata["treatment"]
X =Oregdata[["treatment", "ddddraw_sur_2", "ddddraw_sur_3", "ddddraw_sur_4",␣
 ↪"ddddraw_sur_5", "ddddraw_sur_6",
             "ddddraw_sur_7", "dddnumhh_li_2", "dddnumhh_li_3",␣
 ↪"ddddraXnum_2_2", "ddddraXnum_2_3", "ddddraXnum_3_2",
             "ddddraXnum_3_3", "ddddraXnum_4_2", "ddddraXnum_5_2",␣
 ↪"ddddraXnum_6_2", "ddddraXnum_7_2"]]
```

[252]:
```
pandas.core.frame.DataFrame
```

```
15518
```

[253]:
```
#sample split

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.3,␣
 ↪random_state = 145)
print("Model:")
```

```python
print(X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)
```

```
Model:
(10862, 17) (4656, 17) (10862,) (4656,)
```

Explain why we need sample spliting:

0.1.1 Linear Regression

Simple Model

[254]:
```python
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, Y_train)
```

[254]:
```
LinearRegression()
```

[255]:
```python
print("Coefficient of Treatment:", model.coef_[0])
```

```
Coefficient of Treatment: 0.26191564758749875
```

[256]:
```python
import statsmodels.api as sm

Xc = sm.add_constant(X)
ols = sm.OLS(y, Xc)
ols = ols.fit(cov_type = "HC3")
#ols.summary()
```

[257]:
```python
#print confidence interval for treatment

print("Effect of Treatment:", ols.params["treatment"])
print(" ")
print("Confidence Interval of Treatment")
ols.conf_int(alpha=0.05)[1:2]
print("p-value of Treatment:")
ols.pvalues["treatment"]
```

```
Effect of Treatment: 0.2641965998616533

Confidence Interval of Treatment
```

[257]:

	0	1
treatment	0.177349	0.351044

```
p-value of Treatment:
```

[257]:
```
2.4865426754793102e-09
```

[258]:
```python
#Define MSE Function
import numpy as np

def MSE(Y_pred, Y_true):
    MSE = np.mean((Y_pred - Y_true)**2)
    return MSE
```

[259]:
```python
#predict with model

Y_pred_test = model.predict(X_test)
Y_pred_train = model.predict(X_train)

Y_pred_test.shape
Y_pred_train.shape
Y_test.shape
Y_train.shape
```

[259]: (4656,)

[259]: (10862,)

[259]: (4656,)

[259]: (10862,)

[260]:
```python
#Measure MSE

MSE_reg_s = MSE(Y_pred_test, Y_test)
print("MSE Test Set:", MSE_reg_s)
print("MSE Train Set:", MSE(Y_pred_train, Y_train))
```

```
MSE Test Set: 6.758826786664225
MSE Train Set: 7.105773640137756
```

Complex Model

[261]:
```python
#4 including additional variables to model specification

type(Oregdata)
y =Oregdata["docvis"]
print(len(y))
#trtmt =Oregdata["treatment"]
Xc =Oregdata[["treatment", "ddddraw_sur_2", "ddddraw_sur_3", "ddddraw_sur_4",␣
↪"ddddraw_sur_5", "ddddraw_sur_6",
             "ddddraw_sur_7", "dddnumhh_li_2", "dddnumhh_li_3",␣
↪"ddddraXnum_2_2", "ddddraXnum_2_3", "ddddraXnum_3_2",
             "ddddraXnum_3_3", "ddddraXnum_4_2", "ddddraXnum_5_2",␣
↪"ddddraXnum_6_2", "ddddraXnum_7_2", "female_list",
```

```
            "hhinc_pctfpl_12m", "age2008", "edu_12m_2", "edu_12m_3",␣
↪"edu_12m_4", "english_list", "zip_msa",
            "race_white_12m", "race_black_12m", "race_hisp_12m"]]
```

[261]: pandas.core.frame.DataFrame

15518

[262]:
```
#sample split

from sklearn.model_selection import train_test_split

def split(x, y):
    X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size = 0.3,␣
↪random_state = 145)
    return X_train, X_test, Y_train, Y_test

X_train, X_test, Y_train, Y_test = split(Xc, y)
print("Model:")
print(X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)
```

Model:
(10862, 28) (4656, 28) (10862,) (4656,)

[263]:
```
#fit new specified model

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, Y_train)
#print("Coefficient:", model.coef_)
```

[263]: LinearRegression()

[264]:
```
print("Coefficient of Treatment:", model.coef_[0])
```

Coefficient of Treatment: 0.2690040151454135

[265]:
```
#predict with model

Y_pred_test = model.predict(X_test)
Y_pred_train = model.predict(X_train)
```

[266]:
```
#Measure MSE of new specified model

MSE_reg_c = MSE(Y_pred_test, Y_test)
print("MSE Test Set:", MSE_reg_c)
print("MSE Train Set:", MSE(Y_pred_train, Y_train))
```

```
MSE Test Set: 6.634334329279756
MSE Train Set: 6.9826692988567665
```

0.1.2 Lasso regression

Simple Model

[267]:
```python
X_train, X_test, Y_train, Y_test = split(X, y)
```

[268]:
```python
from sklearn.linear_model import LassoCV
model = LassoCV()
model.fit(X_train, Y_train)
#print("Coefficient:", model.coef_)
print(X_train.shape)
```

[268]:
```
LassoCV()
```

```
(10862, 17)
```

[269]:
```python
#predict with model

Y_pred_test = model.predict(X_test)
Y_pred_train = model.predict(X_train)
```

[270]:
```python
#Measure MSE

MSE_lasso_s = MSE(Y_pred_test, Y_test)
print("MSE Test Set:", MSE_lasso_s)
print("MSE Train Set:", MSE(Y_pred_train, Y_train))
```

```
MSE Test Set: 6.756416315771539
MSE Train Set: 7.116682378986752
```

Complex Model

[271]:
```python
X_train, X_test, Y_train, Y_test = split(Xc, y)
```

[272]:
```python
from sklearn.linear_model import LassoCV
model = LassoCV()
model.fit(X_train, Y_train)
#print("Coefficient:", model.coef_)
print(X_train.shape)
```

[272]:
```
LassoCV()
```

```
(10862, 28)
```

[273]:
```python
#predict with model

Y_pred_test = model.predict(X_test)
```

```
Y_pred_train = model.predict(X_train)
```

[274]:
```
#Measure MSE

MSE_lasso_c = MSE(Y_pred_test, Y_test)
print("MSE Test Set:", MSE_lasso_c)
print("MSE Train Set:", MSE(Y_pred_train, Y_train))
```

```
MSE Test Set: 6.6341529674946145
MSE Train Set: 6.9944395561231065
```

Compare Results

[285]:
```
results = [ ('OLS', MSE_reg_s, MSE_reg_c,) ,
             ('Lasso', MSE_lasso_s, MSE_lasso_c) ]

table = pd.DataFrame(results, columns = ["Method", "Simple" , "Complex",])

table
```

[285]:

	Method	Simple	Complex
0	OLS	6.758827	6.634334
1	Lasso	6.756416	6.634153

Lasso is a little better

Explain k-fold cross-validation

[ ]:
```
```