Q1f

May 15, 2024

(f) In the homework folder you have access to the data file SimpleReg.csv. The data contains a feature column x and a response column y. Read the data, then center the x data, and fit a linear model in the form of $y = \beta 0 + \beta 1x$. Now use an R or Python program to calculate the LOOCV CVn, as we did in the class (if you use R, pick the first element of delta). Also write a code that calculates the CVn using equation (3). You should see that the two methods produce identical results. You may also be surprised with how faster your customized code is, compared to the R cv.glm function!

[1]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from ISLP.models import sklearn_sm
from sklearn.model_selection import cross_validate
from sklearn.preprocessing import StandardScaler
```

[2]:
```
Data = pd.read_csv('SimpleReg.csv')
y = Data['y']
X = Data[['x']]
# standardlize training data
X_cent = StandardScaler(with_mean=True,  with_std=False).fit_transform(X)
X = pd.DataFrame({'intercept': np.ones(X.shape[0]), 'x': X_cent.flatten()})
# X
```

1 = = = = = From Sample code = = = = =

[3]:
```
M = sklearn_sm(sm.OLS)
M_CV = cross_validate(M, X, y, cv=Data.shape[0])
cv_error = np.mean(M_CV['test_score'])
print('CVn:', cv_error)
```

```
CVn: 0.3695297270024033
```

2 ===== $CV_n = \frac{1}{n}\sum_{j=1}^{n}(\frac{y_j - \hat{y}}{1-h_j})^2$ =====

[4]:
```
n = len(Data)
MSE = np.zeros(n)

x = X['x']
Sxy = np.sum(Data.x * Data.y)
Sxx = np.sum(Data.x * Data.x)
muY = np.mean(y)

for j in range(n):

    beta_1_hat = (Sxy - (n/n-1) * x[j] * (y[j] - muY)) / (Sxx - (n/n-1) * x[j]␣
 ↪* x[j])
    beta_0_hat = y[j] - (n * (y[j] - muY) / (n-1)) + (beta_1_hat * x[j] / (n-1))

    # predict y_hat
    y_hat = beta_0_hat + (beta_1_hat * x[j])

    # calculate MSE
    h_j = 1/n + ((x[j] * x[j]) / Sxx)
    MSE[j] = ( (y[j] - y_hat) / (1 - h_j) ) ** 2

print('CVn:', np.sum(MSE)/len(Data))
```

```
CVn: 0.3703803767708858
```

The results are pretty close, all around 0.37, but not identical.