Lab2_practice

September 16, 2021

1 Lab 2 - Basic concepts in Machine Learning

1.0.1 Author: Yunting Chiu

1.1 Exercise 1

Write a Python program to reproduce the results of Example 3.1 page 74 and 75. Use different values of the noise variance and report your findings

1.2 Install the Packages

[1]:
```
# Install the related libraries
%matplotlib inline
import matplotlib as plt
import numpy as np
import random
import math
from sklearn import datasets
import pandas as pd
import os
```

[2]:
```
n = 50 # 50 samples
theta = np.array([[0.25, -0.25, 0.25]]) # stated in the question, 2D array
sigma1 = math.sqrt(1) # variance is 1, so the sd is sqrt(var)
sigma2 = math.sqrt(10) # variance is 10
sigma3 = math.sqrt(100) # variance is 100
sigma4 = math.sqrt(1000) # variance is 1000
np.random.seed(10)
```

1.3 Creating the Simulated Data

The regression model will be written as:

$$y = \theta^T X + \eta$$

[3]:
```
X = np.zeros(shape = (3, n), dtype=np.float32)
X[0, :] = 10*np.random.rand(1, n) # uniform distribution over [0, 1), then * 10.
↪ So [0, 10)
```

```
X[1, :] = 10*np.random.rand(1, n)
X[2, :] = np.ones(shape = (1, n))
# print(X) # X is 2D array
y = np.dot(theta, X)
#print(y)
print(y.shape)
```

```
(1, 50)
```

[4]:
```
print(X.shape)
```

```
(3, 50)
```

[5]:
```
# Add noise term to the original points
y1 = y + sigma1*np.random.rand(y.shape[0], y.shape[1]) # return 1x50 array
y2 = y + sigma2*np.random.rand(y.shape[0], y.shape[1]) # return 1x50 array
y3 = y + sigma3*np.random.rand(y.shape[0], y.shape[1])
y4 = y + sigma4*np.random.rand(y.shape[0], y.shape[1])
print(y1)
```

```
[[ 1.77511488  0.92216268 -0.15054627  2.20866053  1.31793054 -0.83005729
  -1.13788463  1.04765981 -1.04187366  0.4056049   0.76970034  0.8887382
  -0.40305731  0.21417091  2.23382557  1.22617669  1.96341933  0.39494318
   2.6034641   0.46722859  1.72523224  0.28419082  0.5029143   1.08870944
   0.95850035  1.82554725 -0.25775016 -0.191221    0.32582345  0.75745252
   0.99772654  0.81285982  1.25380393  0.7739626   0.6462029   1.86360759
   0.34405442  2.24998464 -1.01184463  1.3798495   1.00416372  2.3579796
   1.14829855  0.82241314  0.86405782  1.60784749 -1.30426408  1.78635744
   0.96198717 -0.53233298]]
```

The **LS estimate** is given by

$$\hat{\theta} = (X^T X)^{-1} X^T y$$

[6]:
```
# Apply LS solution on Eq(3.17), we need to make sure the dimension is right
LS1 = np.dot(np.dot(np.linalg.inv(np.dot(X, X.conj().T)), X), y1.conj().T)
LS2 = np.dot(np.dot(np.linalg.inv(np.dot(X, X.conj().T)), X), y2.conj().T)
LS3 = np.dot(np.dot(np.linalg.inv(np.dot(X, X.conj().T)), X), y3.conj().T)
LS4 = np.dot(np.dot(np.linalg.inv(np.dot(X, X.conj().T)), X), y4.conj().T)
print(LS1)
```

```
[[ 0.24001312]
 [-0.25841043]
 [ 0.80558171]]
```

[7]:
```
print("If variance equal to 1, the theta 0 is.{}, theta 1 is.{}, and theta 2 is.
 ↪{}".format(LS1[0], LS1[1], LS1[2]))
```

```
If variance equal to 1, the theta 0 is.[0.24001312], theta 1 is.[-0.25841043],
and theta 2 is.[0.80558171]
```

[8]:
```
print("If variance equal to 10, the theta 0 is.{}, theta 1 is.{}, and theta 2␣
↪is.{}".format(LS2[0], LS2[1], LS2[2]))
```

```
If variance equal to 10, the theta 0 is.[0.30958399], theta 1 is.[-0.26907574],
and theta 2 is.[1.46816628]
```

[9]:
```
print("If variance equal to 100, the theta 0 is.{}, theta 1 is.{}, and theta 2␣
↪is.{}".format(LS3[0], LS3[1], LS3[2]))
```

```
If variance equal to 100, the theta 0 is.[0.1070667], theta 1 is.[-0.26201911],
and theta 2 is.[6.05287797]
```

[10]:
```
print("If variance equal to 1000, the theta 0 is.{}, theta 1 is.{}, and theta 2␣
↪is.{}".format(LS4[0], LS4[1], LS4[2]))
```

```
If variance equal to 1000, the theta 0 is.[0.32073315], theta 1
is.[-0.25493826], and theta 2 is.[16.61228336]
```

[11]:
```
df1 = pd.DataFrame(LS1, columns=['var1'])
df2 = pd.DataFrame(LS2, columns=['var10'])
df3 = pd.DataFrame(LS3, columns=['var100'])
df4 = pd.DataFrame(LS4, columns=['var1000'])
df = pd.concat([df1, df2, df3, df4], axis = 1)
df
```

[11]:

	var1	var10	var100	var1000
0	0.240013	0.309584	0.107067	0.320733
1	-0.258410	-0.269076	-0.262019	-0.254938
2	0.805582	1.468166	6.052878	16.612283

1.4 Findings

The larger the variance, the less accurate the θs. Ideally, we need to reduce the error term.

1.5 Output

[12]:
```
%%capture
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
↪texlive-plain-generic
!jupyter nbconvert --to pdf "/content/drive/MyDrive/American_University/
↪2021_Fall/DATA-642-001_Advanced Machine Learning/GitHub/Labs/02/
↪Lab2_practice.ipynb"
```