Lab4_Yunting

October 6, 2021

1 Lab04 - Sparsity Aware Learning

Author: Yunting Chiu

2 Exercise 1

[1]:
```
# Install the libraries
import random
import numpy as np
import matplotlib.pyplot as plt
from functools import reduce
%matplotlib inline
plt.style.use(['ggplot'])
```

2.1 1A

[2]:
```
X = np.array([[0.5, 2, 1.5], [1.5, 2.3, 3.5]]) # matrix X
theta = np.array([2.5, 0, 0], ndmin = 2).T
y = np.dot(X, theta)
print(theta)
```

```
[[2.5]
 [0. ]
 [0. ]]
```

[3]:
```
print(X)
```

```
[[0.5 2.  1.5]
 [1.5 2.3 3.5]]
```

[4]:
```
print(y)
```

```
[[1.25]
 [3.75]]
```

According to the textbook 9.10, we can know the L2 minimizer accepts the closed from the following solution:

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

where $X^T(XX^T)^{-1}$ (a m x n matrix) is called a pseudo-inverse.

[5]:
```
# theta2 = np.dot(np.dot(X.T, np.linalg.inv(np.dot(X, X.T))), y)
theta2 = np.dot(np.dot(X.T, np.linalg.inv(np.dot(X, X.T))), y)
error_L2 = np.linalg.norm(y - np.dot(X, theta2))
error_theta = np.linalg.norm(theta2 - theta)
print('The L2 norm minimized solution is {}'.format(theta2))
```

```
The L2 norm minimized solution is [[ 0.5641321 ]
 [-0.27265745]
 [ 1.00883257]]
```

[6]:
```
print(" The error achieved with L2 norm minimization is {}, which is closed to␣
 ↪zero".format(error_L2))
```

```
 The error achieved with L2 norm minimization is 9.930136612989092e-16, which is
closed to zero
```

There is an existing function called `numpy.linalg.pinv` of pseudoinverse of X matrix, the outcome is also the same as the previous formula.

[7]:
```
pesu_X_ex1 = np.dot(X.T, np.linalg.inv((np.dot(X, X.T))))
print(pesu_X_ex1)
print("---------------------------")
pesu_X_ex2 = np.linalg.pinv(X)
print(pesu_X_ex2)
```

```
[[-0.36897081  0.2734255 ]
 [ 0.93394777 -0.38402458]
 [-0.45560676  0.42089094]]
---------------------------
[[-0.36897081  0.2734255 ]
 [ 0.93394777 -0.38402458]
 [-0.45560676  0.42089094]]
```

2.2 1B

We need to estimate the smallest number of parameters that can be explained the obtained observations. Consider all possible combinations of zero in θ, removing the respective columns of X and check whether the system of equations is satisifed.

Let's start checking for possible 1-sparse solution.

2.2.1 1-sparse solution [x, 0, 0]

This one has the ideal solution, the θ_0 lead to zero estimation error.

[8]:
```
subX_11 = np.array(X[:, 0], ndmin = 2).T # ndmin = Number of array dimensions
# print(subX_11)
theta_11 = np.zeros((3, 1))
theta_11[0] = np.dot(np.linalg.inv(np.dot(subX_11.T, subX_11)), np.dot(subX_11.
 ↪T, y))
print(theta_11)
error1 = np.linalg.norm(y - np.dot(X, theta_11)) #check that theta_11 is a␣
 ↪solution
error_theta1 = np.linalg.norm(theta_11 - theta)
print('Achieved error: %.20f'% error1)
print('Achieved error in theta %.20f'% error_theta1)
```

```
[[2.5]
 [0. ]
 [0. ]]
Achieved error: 0.00000000000000000000
Achieved error in theta 0.00000000000000000000
```

2.2.2 1-sparse solution [0, x, 0]

[9]:
```
subX_12 = np.array(X[:, 1], ndmin = 2).T # ndmin = Number of array dimensions
# print(subX_22)
theta_12 = np.zeros((3, 1))
theta_12[1] = np.dot(np.linalg.inv(np.dot(subX_12.T, subX_12)), np.dot(subX_12.
 ↪T, y))
print(theta_12)
error2 = np.linalg.norm(y - np.dot(X, theta_12)) #check that theta_22 is a␣
 ↪solution
error_theta2 = np.linalg.norm(theta_12 - theta)
print('Achieved error: %.20f'% error2)
print('Achieved error in theta %.20f'% error_theta2)
```

```
[[0.        ]
 [1.19752422]
 [0.        ]]
Achieved error: 1.51741327826356919850
Achieved error in theta 2.77201447624411478898
```

2.2.3 1-sparse solution [0, 0, x]

[10]:
```
subX_13 = np.array(X[:, 2], ndmin = 2).T # ndmin = Number of array dimensions
# print(subX_22)
theta_13 = np.zeros((3, 1))
theta_13[2] = np.dot(np.linalg.inv(np.dot(subX_13.T, subX_13)), np.dot(subX_13.
 ↪T, y))
print(theta_13)
```

```
error3 = np.linalg.norm(y - np.dot(X, theta_13)) # check that theta2 is a␣
 ↪solution
error_theta3 = np.linalg.norm(theta_13 - theta)
print('Achieved error: %.20f'% error3)
print('Achieved error in theta %.20f'% error_theta3)
```

```
[[0.        ]
 [0.        ]
 [1.03448276]]
Achieved error: 0.32826608214930647067
Achieved error in theta 2.70557841835779600004
```

1-sparse [x, 0, 0] has the ideal solution, so there is no need to evaluate 2-sparse solutions

[11]:
```
theta0 = theta_11 # because this combination has the lowest error
print("With zero estimation error of {}, we can skip for searching 2-sparse␣
 ↪solutions".format(theta0))
```

```
With zero estimation error of [[2.5]
 [0. ]
 [0. ]], we can skip for searching 2-sparse solutions
```

2.3 1C

As the θ_2 is the smaller one in L2 norms, which makes sense.

[12]:
```
print("L0 norm of theta 0 is {}".format(np.linalg.norm(theta0)))
print("L2 norm of theta 2 is {}".format(np.linalg.norm(theta2)))
if np.linalg.norm(theta0) > np.linalg.norm(theta2):
  print("theta2 is the smaller one")
else:
  print("theta0 is the smaller one")
```

```
L0 norm of theta 0 is 2.5
L2 norm of theta 2 is 1.187573265586891
theta2 is the smaller one
```

3 Testing Zone

[13]:
```
"""
print('Check solution [x, 0, 0]')
theta_11 = np.zeros((3, 1))
# print(theta_11)
subX_11 = np.array(X[:, 0], ndmin = 2).T

theta_11[0] = np.dot(np.linalg.inv(np.dot(subX_11.T, subX_11)), np.dot(subX_11.
 ↪T, y))
print(theta_11)
```

```
error1 = np.linalg.norm(y - np.dot(X, theta_11)) #check that theta_11 is a␣
 ↪solution
error_theta1 = np.linalg.norm(theta_11 - theta)

print('Achieved error: %.20f'% error1)
print('Achieved error in theta %.20f'% error_theta1)
print('--------------')
"""
```

[13]:

```
"\nprint('Check solution [x, 0, 0]')\ntheta_11 = np.zeros((3, 1))\n#
print(theta_11)\nsubX_11 = np.array(X[:, 0], ndmin = 2).T\n\ntheta_11[0] =
np.dot(np.linalg.inv(np.dot(subX_11.T, subX_11)), np.dot(subX_11.T,
y))\nprint(theta_11)\nerror1 = np.linalg.norm(y - np.dot(X, theta_11)) #check
that theta_11 is a solution\nerror_theta1 = np.linalg.norm(theta_11 -
theta)\n\nprint('Achieved error: %.20f'% error1)\nprint('Achieved error in theta
%.20f'% error_theta1)\nprint('--------------')\n"
```

4 Output

[ ]:

```
# should access the Google Drive files before running the chunk
%%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/04/submit/
 ↪Lab4_Yunting.ipynb"
```