國立雲林科技大學機械工程系

機器學習課堂報告

Linear Algebra

學生：機械四B B10711150 葉書廷

指導教授：吳英正 教授

日期：民國 111 年　3 月 30 日

目的：

先前實做 Curve Fitting 時我們曾使用 Polyfit 的方式直接解出方程式之係數，而本次將使用線性代數中的矩陣運算解出其係數，並比較兩者差異。

方法：

一般來說，以方程式$ax + b = y$為例，在已知(x, y)的情況下要解出 a、b，需要至少兩個(x, y)才能解出唯一解，同理，方程式$ax^2 + bx + c = y$需要至少三組(x, y)才能解出未知數 a、b、c，而解聯立方程式的方式其中一種就是利用反矩陣：

$$ax + b = y \quad \rightarrow \quad \begin{bmatrix} x1 & x1^0 \\ x2 & x2^0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y1 \\ y2 \end{bmatrix} \text{.....(1)}$$

$$X \cdot p = y$$

$$\rightarrow \quad p = X^{-1} \cdot y \text{.....(2)}$$

透過式子(2)，我們可以利用反矩陣解出各項係數，但是，反矩陣僅在方陣中存在，也就是說，若今天有 40 行聯立方程式，僅要求出二次方程式的三個係數，我們無法從一個 40x3 的矩陣中求出反矩陣以及各項係數的唯一解，此時需要利用一個方法求出近似的反矩陣，擬反矩陣(Pseudoinverse Matrix)，利用此方式也能求出方程式各項係數的最佳解，其定義為：

$$A^{+} = (A^{T}A)^{-1}A^{T}$$

將其代回(2)中取代X^{-1}的位子，即能求出方程式各項係數之最佳解。

結果：

從(圖一)可以看出，使用 Polyfit 與 Pseudoinverse 兩種方式計算出的方程式以及標準差一模一樣，事實上在小數點後 14、15 位的位置開始才有一點差別，但這種微小差距基本上可以忽略。

```
=============
|| Polyfit ||
=============
equation :  + (0.06276)x^2 + (-0.02694)x^1 + (2.85902)x^0
Standard deviation : 0.216290

====================
|| Pseudoinverse ||
====================
equation :  + (0.06276)x^2 + (-0.02694)x^1 + (2.85902)x^0
Standard deviation : 0.216290
```

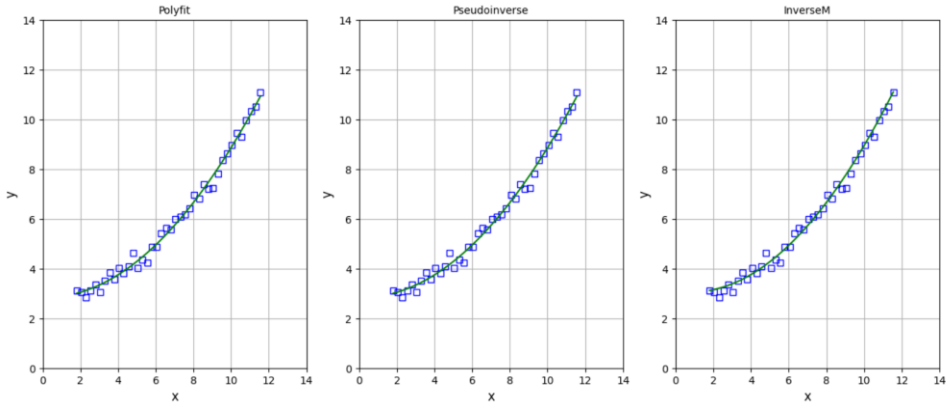
(圖一)、Polyfit 與 Pseudoinverse 兩種輸出結果

若是假設(x, y)資料只有三筆，分別是原資料中的第 1 筆、第 21 筆、以及第 40 筆，利用三筆資料求三個係數項，將 X 塑型成方陣，就能夠使用反矩陣的方式處理，而其結果如(圖二)、(圖三)，可以看出其方程式係數、標準差以及圖形趨勢都與 Pseudoinverse 的結果相像，但因其資料只取最初最後以及中間的三筆(經測試這三筆具有最好的標準差)，而畫出來的曲線是與原資料(40 筆)比對，所以標準差表現較差。

```
====================
|| Pseudoinverse ||
====================
equation :  + (0.06276)x^2 + (-0.02694)x^1 + (2.85902)x^0
Standard deviation : 0.216290

==============
|| InverseM ||
==============
equation :  + (0.06984)x^2 + (-0.11433)x^1 + (3.10912)x^0
Standard deviation : 0.228199
```

(圖二)、Pseudoinverse 與 inverseMatrix 兩種輸出結果



(圖二)、三種方式圖形比較

程式撰寫：

```
import numpy as np
import matplotlib.pyplot as plt

def read_data(datafile_name):
    # read from txt
    data = np.loadtxt('python\\Deep_Learning\\DL_IngJeng\\' +
                      datafile_name+'.csv',delimiter=',') #from vscode root directory
    n = len(data[:, 0])
    x = data[:, 0]
    y = data[:, 1]
    return n, x, y

def Polyfit(n_poly, n, x, y):
    return np.polyfit(x, y, n_poly)

def Pseudoinverse(n_poly, n, x, y):
```

```
    a = np.ones([n_poly + 1, n])
    for i in range(n_poly):
        a[i] = x ** (n_poly-i)
    p = np.zeros([1, n_poly + 1])
    X = np.transpose(a)
    Y = np.transpose(y)
    p = np.transpose(p)
    XT = np.transpose(X)
    XXTinv = np.linalg.inv(np.dot(XT, X))
    p = np.dot(np.dot(XXTinv, XT), Y)
    return p

def InverseM(n_poly, n, x, y):
    a = np.ones([n_poly + 1, n])
    for i in range(n_poly):
        a[i] = x ** (n_poly-i)
    x = [a.T[0], a.T[20], a.T[39]]
    y = [y[0], y[20], y[39]]
    p = np.zeros([1, n_poly + 1])
    p = np.dot(np.linalg.inv(x), y)
    return p

def main():
    Solution = {}
    Solution['Polyfit'], Solution['Pseudoinverse'], Solution['InverseM'] = {}, {}, {}
    n_poly = 2 #determine the n degree polynomial equation
    datafile_name = 'data_of_poly2' #input("Enter the data file name:")
    n, x, y= read_data(datafile_name)
    Solution['Polyfit']['para'] = Polyfit(n_poly, n, x, y)
    Solution['Pseudoinverse']['para'] = Pseudoinverse(n_poly, n, x, y)
    Solution['InverseM']['para'] = InverseM(n_poly, n, x, y)
    i=1
    plt.figure(figsize=(18, 6))
    for key in Solution.keys():
        Solution[key]['y_ex'] = 0
        Solution[key]['y_eq'] = ''
        for j in range(0,n_poly+1):
            Solution[key]['y_ex'] += Solution[key]['para'][j] * (x**(n_poly-j))
            Solution[key]['y_eq'] += '+ (' + str(round(Solution[key]['para'][j], 5)) + ')x^' + str(n_poly
- j) + ' '
        sy2 = np.sum((Solution[key]['y_ex']-y)**2)/(n-(n_poly+1))
        Solution[key]['sy'] = sy2**0.5
        print(','='*(len(key)+6), '\n ||', key, '||\n','='*(len(key)+6))
        print(' equation：', Solution[key]['y_eq'])
        print(' Standard deviation：%f\n'%Solution[key]['sy'])
        plt.subplot(int('13'+str(i)))
        plt.xlabel('x', fontsize=12), plt.ylabel('y', fontsize=12)
        plt.xlim(0, 14), plt.ylim(0, 14)
        plt.title(key, fontsize=10)
        plt.grid(True, which='both')
        plt.plot(x, y, 'bs', markerfacecolor='none')
        plt.plot(x, Solution[key]['y_ex'], 'g')
        i+=1
    plt.show()
main()
```