國立雲林科技大學機械工程系

機器學習課堂報告

Curve Fitting

學生：機械四B B10711150 葉書廷

指導教授：吳英正 教授

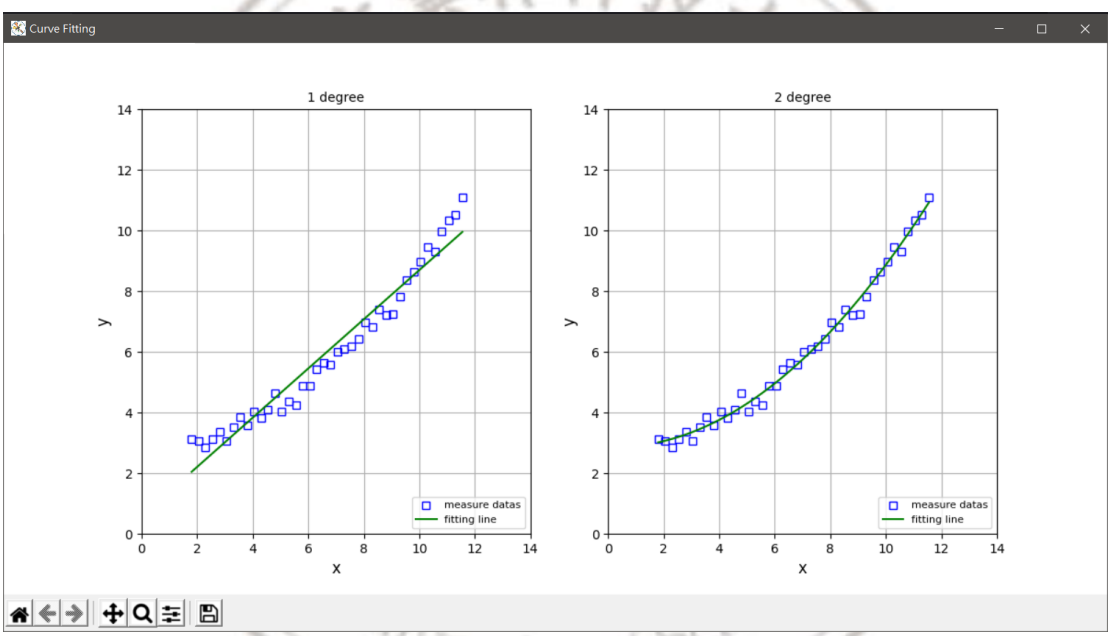
日期：民國 111 年　3 月 09 日

目的：

當我們擁有一些(x, y)對應的數據時，可以利用 Curve Fitting 的方式取得一個函數圖形擬和這些數據，這樣的方式能夠在未來我們再次得到一個資料(x)時，能夠藉由這個函數圖形去推測這個資料對應的數據(y)，並透過計算標準差的方式判斷這些函數圖形擬和情形的好壞，並比較多項式的次數對標準差的影響。

結果：

根據(圖一)可以發現，一次項以及二次項的 Curve Fitting 有著明顯的差異，二次項在線段與數據的吻合程度高過一次項非常多，也從(圖二)中得知其標準差比一次項要小上不少。

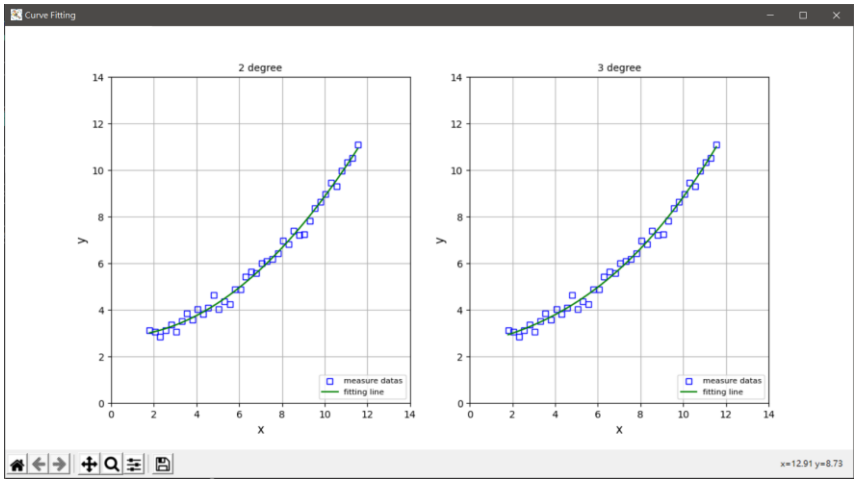


(圖一)、一次與二次 Curve Fitting 圖形

```
1 degree : 0.5245404702645863
2 degree : 0.2134253953148766
```

(圖二)、一次與二次之標準差

若我們繼續提高多項式的次數的話，可以得知其圖形已經沒有顯著的變化(圖三)，但標準差卻有變得更好的趨勢(圖四)，由此可以先簡單推斷提高次數能夠使標準差更好。



（圖三）、二次與三次 Curve Fitting 圖形

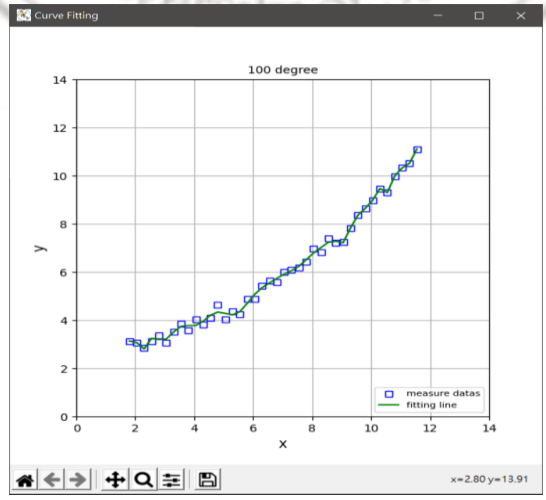
```
2 degree : 0.2134253953148766
3 degree : 0.21047239853107444
```

（圖四）、二次與三次之標準差

繼續提高次數就會發現，並不是只要無限增加多項式的次數，就能夠無限的降低標準差，事實上，在這個範例中，我將次數提高到 17 次時，就出現過擬合(圖五)的情況，也就是說第 17 次的標準差比第 16 次的還要大，且出現 RankWarning 的警告，這是由於計算的數值誤差導致擬和沒有正確定義，除此之外，增加多項式的次數，還會導致線段不夠平滑(圖六)，且過多的係數也容易造成計算上的麻煩或是係數本身位數過多導致計算誤差，所以不建議將次數無限提高。

```
16 degree : 0.16264766543121004
d:/vscode/python/Deep_Learning/DL_IngJeng/HW03_20220309.py:35: RankWarning: Polyfit may be poorly conditioned
  n_poly, y_expected = para_calculation(n, x, y)
17 degree : 0.16277684539376389
```

（圖五）、16 次與 17 次產生的過擬和與錯誤警告



（圖六）、100 次 Curve Fitting 圖形

程式撰寫：

```
#Curve Fitting
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 para_calculation(n, x, y):
    n_poly = 2 # n degree polynomial equation (1~9)
    y_ex = [0 for i in range(n_poly)] #intialize list to storage data
    sy2 = [0 for i in range(n_poly)]
    sy = [0 for i in range(n_poly)]
    y_eq = ['' for i in range(n_poly)]
    print("Standard Deviation：")
    for i in range(1,n_poly+1):
        z = np.polyfit(x, y, i)
        for j in range(0,i+1):
            y_ex[i-1] += z[-(j+1)] * (x**j)
            y_eq[i-1] += '+(' + str(z[-(j+1)]) + ')x^' + str(j)
        sy2[i-1] = np.sum((y_ex[i-1]-y)**2)/(n-2)
        sy[i-1] = sy2[i-1]**0.5
        print(i,"degree :",sy[i-1]) #print standard deviation
    [print('<',i,'equation > :',y_eq[i-1]) for i in range(1,n_poly+1)] #print polynomial
eq
    return n_poly, y_ex

def main():
    datafile_name = 'data_of_poly2' #input("Enter the data file name:")
    n, x, y = read_data(datafile_name)
    n_poly, y_expected = para_calculation(n, x, y)
```

```
    plt.figure('Curve Fitting',figsize=(12, 6))
    for i in range(n_poly):
        plt.subplot(int('1'+str(n_poly)+str(i+1)))
        plt.xlabel('x', fontsize=12), plt.ylabel('y', fontsize=12)
        plt.xlim(0, 14), plt.ylim(0, 14)
        plt.title(str(i + 1) +' degree', fontsize=10)
        plt.grid(True, which='both')
        plt.plot(x, y, 'bs', markerfacecolor='none', label='measure datas')
        plt.plot(x, y_expected[i], 'g', label='fitting line')
        plt.legend(loc='lower right', prop={'size': 8})

main()
plt.show()
```