assignment07

November 15, 2018

1 This is script implement approximate using pseudo inverse matrix

2 Name : Jee-Hye Yang

3 Student ID : 20145708

4 GitHub address : https://github.com/geehyeS2/assignment07

4.0.1 import packages for plottion graphs and manipulating data:

```
In [97]: import numpy as np
         import matplotlib.pyplot as plt
         import numpy.linalg as lin
```

4.0.2 Defined number of point and std

```
In [98]: num      = 1001
         std      = 5
```

4.0.3 Indicate random point 'x' as a function

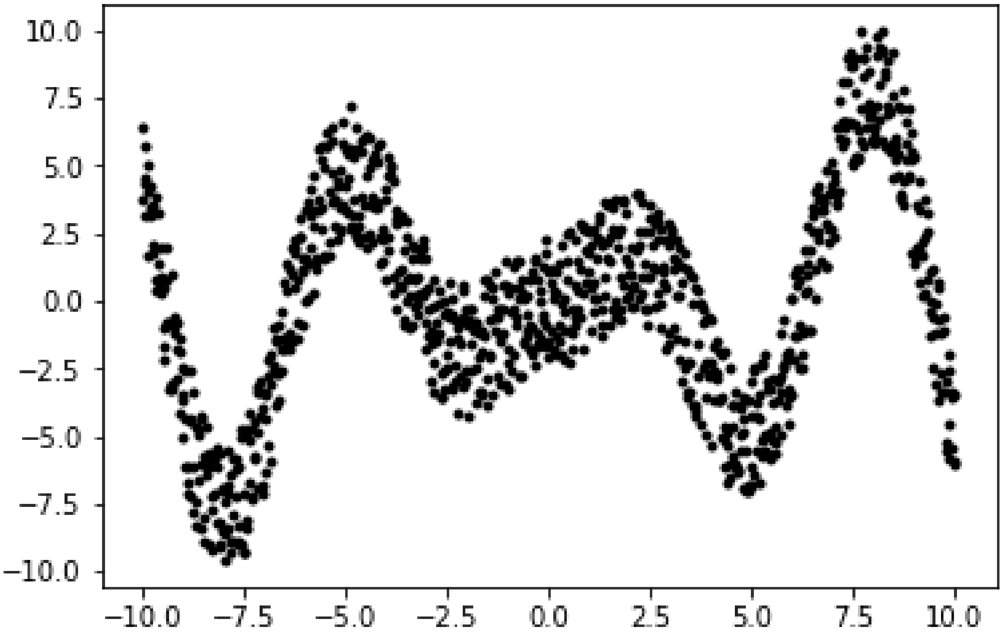
```
In [99]: def fun(x):
                 # f = np.sin(x) * (1 / (1 + np.exp(-x)))
                 f = np.abs(x) * np.sin(x)
                 return f
```

4.0.4 x is x-coordinate data and y1 is (noisy) y-coordinate data

```
In [100]: n        = np.random.rand(num)
          nn       = n - np.mean(n)
          x        = np.linspace(-10,10,num)
          y1       = fun(x)+ nn * std
          # x   : x-coordinate data
          # y1 : (noisy) y-coordinate data
```

4.0.5 Plot the noisy data (x, y1)

```
In [101]: plt.plot(x, y1, 'k.') #y1 : noise
          plt.show()
```



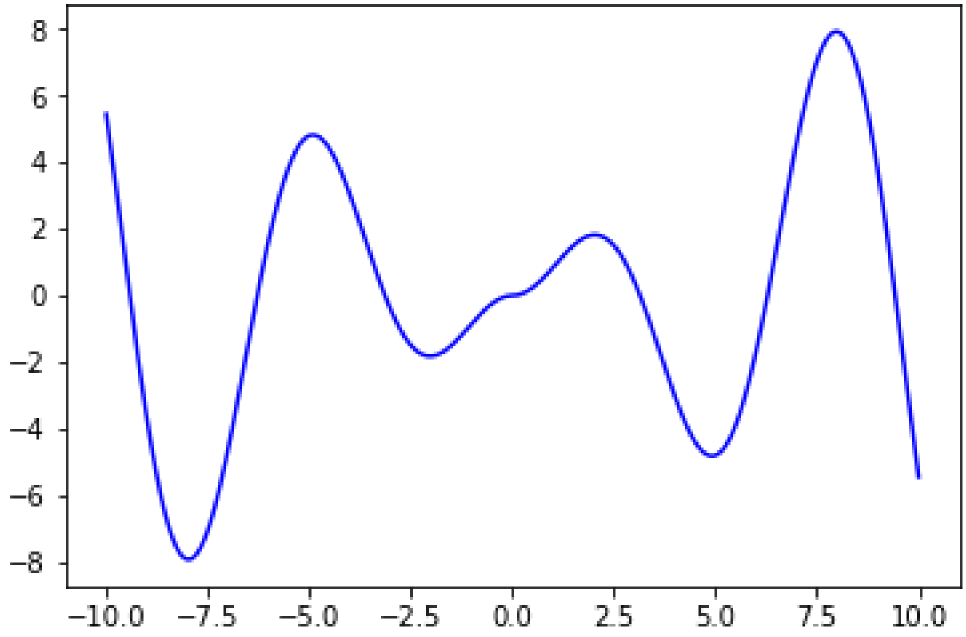
4.0.6 Defined a model polynomial function with model parameters

'para' is pseudo inverse

```
In [102]: def appr(X,y,p) :
              arr = np.ones((x.shape[0],p))
              for i in range(p):
                  arr.T[i]=X**i
              para = np.dot(np.dot(lin.inv(np.dot(arr.T,arr)), arr.T),y)
              return np.dot(arr,para)
```

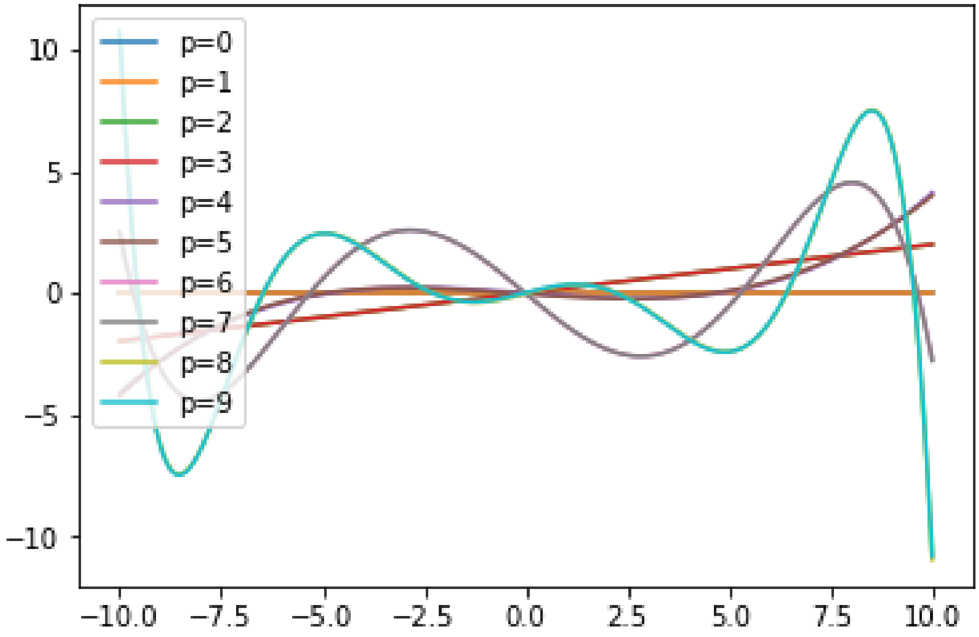
4.0.7 Plot the clean data (x, y2)

```
In [103]: y2 = fun(x) #y2 : clean
          plt.plot(x, y2, 'b')
          plt.show()
```



4.0.8 Plot the polynomial curves that fit the noisy data with varying p = 0,1,2,3,ůůů9

```
In [104]: for i in range(10): #y1 : noise
              y3 = appr(x,y1,i) #y3 :  polynomial curves
              string = "p="+ str(i)
              plt.plot(x, y3, label = string)
          plt.legend(loc='upper left')
          plt.show()
```

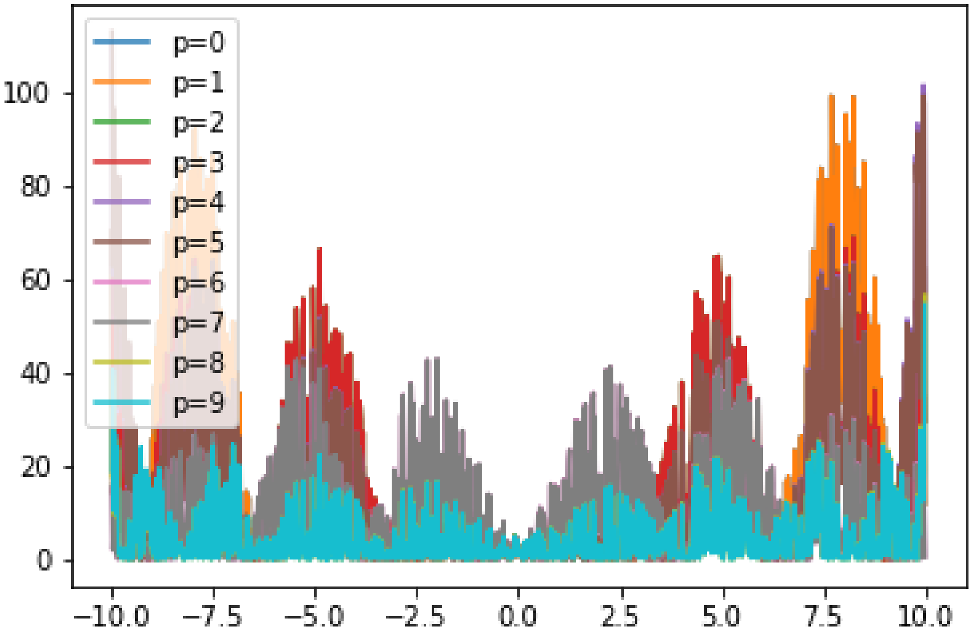


4.0.9 Defined error function

```
In [105]: def error(y1,y2):
              temp =y2-y1
              temp = (np.abs(temp))**2
              return temp
```

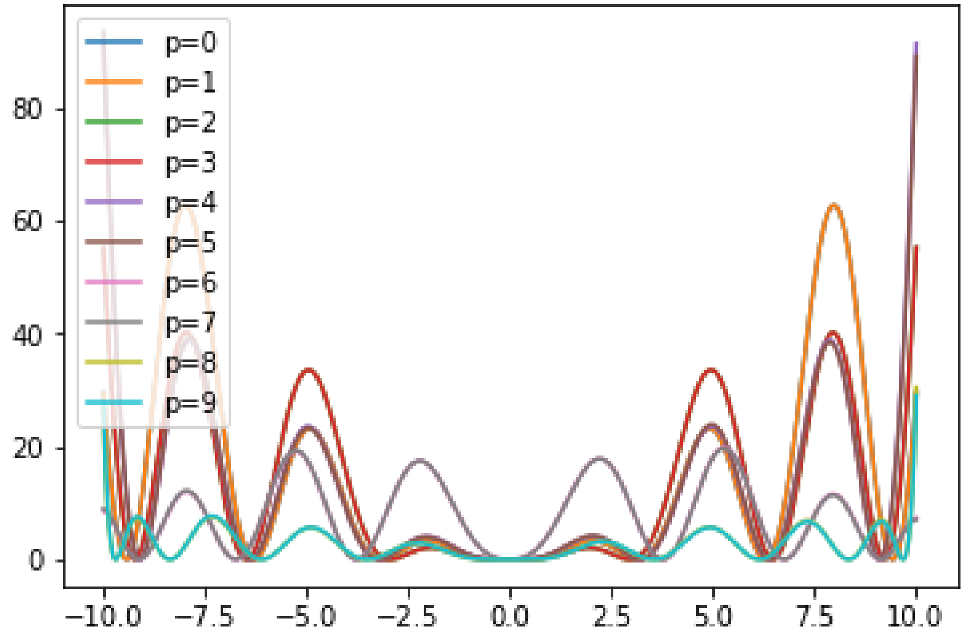
4.0.10 Plot the polynomial curves that fit the noisy data by the least square error with varying p = 0,1,2,3,ůůů9

```
In [106]: for i in range(10):
              y3 = appr(x,y1,i) #y1 : noise data, y3 : clean data
              string = "p="+ str(i)
              plt.plot(x, error(y3,y1), label = string) #by the least square error
          plt.legend(loc='upper left')
          plt.show()
```



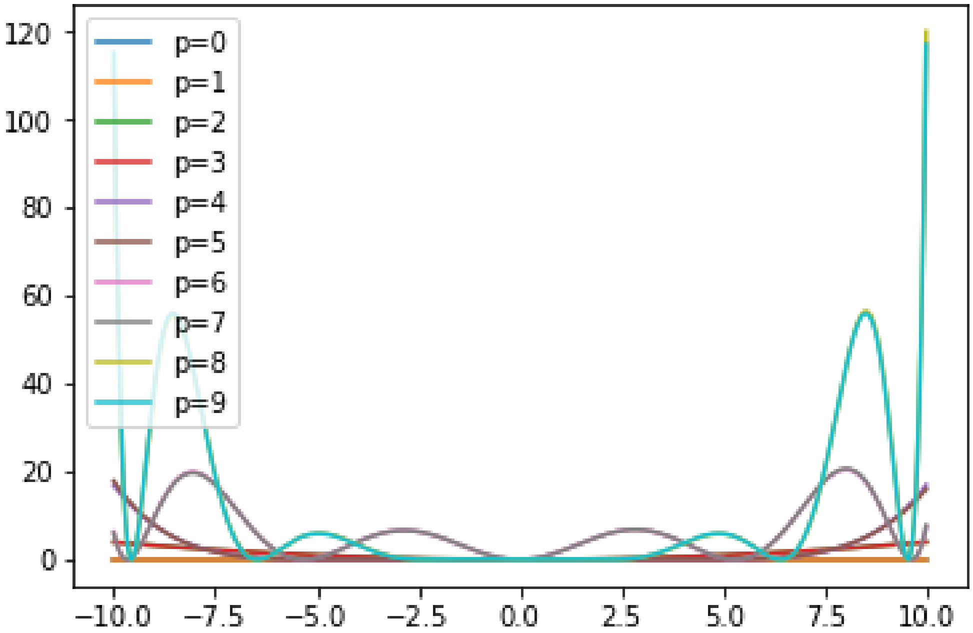
4.0.11 Plot the polynomial curves that fit the clean data by the least square error with varying p = 0,1,2,3,ůůů9

```
In [107]: for i in range(10):
              y3 = appr(x,y1,i)
              string = "p="+ str(i)
              plt.plot(x, error(y3, y2), label = string)
          plt.legend(loc='upper left')
          plt.show()
```



4.0.12 Find an optimal set of model parameters that provide the least square approximate solution

```
In [108]: for i in range(10):
              y4 = appr(x,y3,i)
              string = "p="+ str(i)
              plt.plot(x, error(y4,0), label = string)
          plt.legend(loc='upper left')
          plt.show()
```


120
100
80
60
40
20
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
p=0
p=1
p=2
p=3
p=4
p=5
p=6
p=7
p=8
p=9