20134570_Assign02

March 20, 2019

0.1 20134570 Jung Chaejin

This is an assignment 02 for class about Pattern Recognition Class.
My function is $f(x) = \frac{\sin(x)}{x^2+1}$
import numpy and pyplot

```
In [3]: import matplotlib.pyplot as plt
        import numpy as np
```

Define my Function

```
In [4]: def Function(x):
            f = np.sin(x) / (1 + x*x)
            return f
```

Define given funtion's derivative function
My Derivative function is $\frac{\partial f}{\partial x} = -\frac{2x\sin(x)+(-x^2-1)\cos(x)}{x^4+2x+1}$

```
In [5]: def Derivative(x):
            f = -1 * ((2*x*np.sin(x) + ((-1*x*x - 1)*np.cos(x)))/(x*x*x*x + 2*x*x + 1))
            return f
```

Define Taylor Approximation
$\hat{f}(x) = f(z) + \nabla f(z)^T (x - z)$

```
In [6]: def Taylor(x, z):
            f = Function(z) + Derivative(z)*(x-z)
            return f
```

Define the domain
I will take [-10 , 10, 0.1]

```
In [7]: x = np.arange(-10,10,0.1)
```

Set domain to my funtion

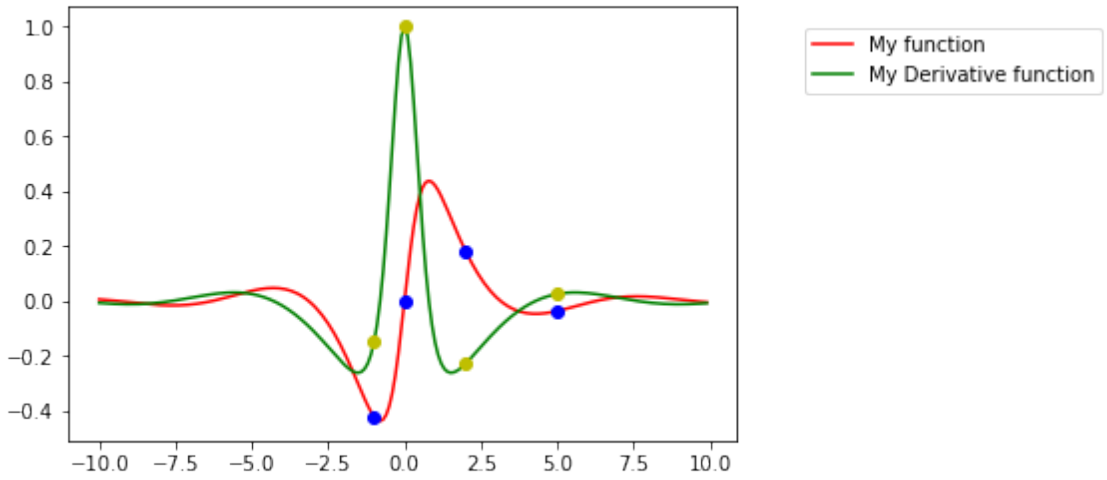
```
In [8]: myf = Function(x)
        myDf = Derivative(x)
```

Set points
$z1(-1, f(-1))$
$z2(0, f(0))$
$z3(2, f(2))$
$z4(5, f(5))$

```
In [9]: z = np.array([-1, 0, 2, 5])
        f = Function(z)
        Df = Derivative(z)
```

Plot graph of Function and Derivative funtion
Red Graph = My function
Green Graph = My Derivative Function
Yellow and Blue Dots = given Point z1, z3, z3, z4

```
In [10]: plt.figure(1)
         plt.plot(x, myf, 'R', label="My function")
         plt.plot(x, myDf, 'G', label="My Derivative function")
         plt.plot(z, f, 'bo')
         plt.plot(z, Df, 'yo')
         plt.legend(bbox_to_anchor=(1.1, 0.8), loc=3, borderaxespad=0)
         plt.show()
```



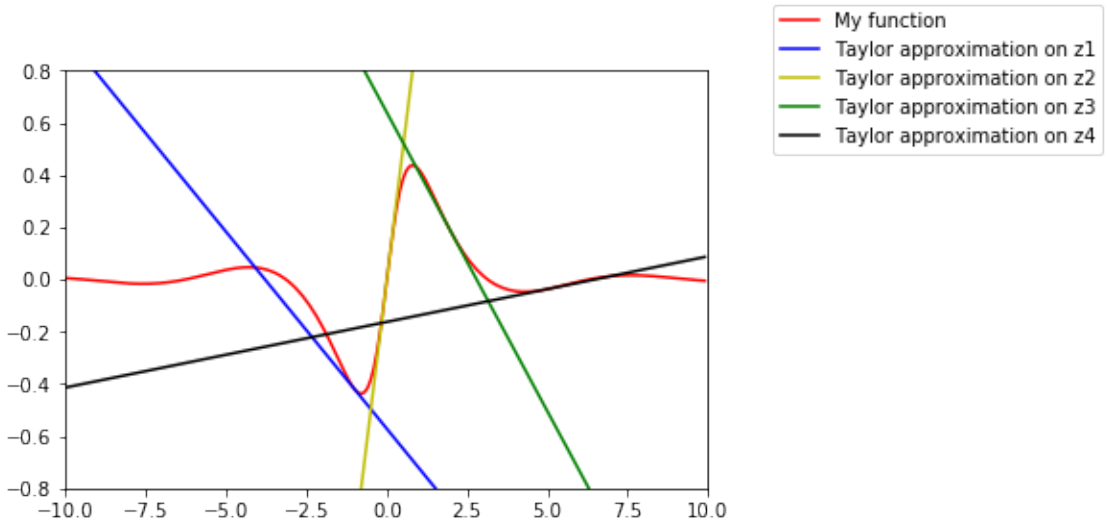
Adjust Taylor Approximation on $z1, z2, z3, z4$

```
In [11]: t1 = Taylor(x, z[0])
         t2 = Taylor(x, z[1])
         t3 = Taylor(x, z[2])
         t4 = Taylor(x, z[3])
```

Plot Graph with give Taylor points

```
In [12]: plt.figure(2)
         plt.plot(x, myf, 'R', label="My function")
         plt.plot(x, t1, 'b', label="Taylor approximation on z1")
         plt.plot(x, t2, 'y', label="Taylor approximation on z2")
         plt.plot(x, t3, 'g', label="Taylor approximation on z3")
         plt.plot(x, t4, 'k', label="Taylor approximation on z4")
         plt.legend(bbox_to_anchor=(1.1, 0.8), loc=3, borderaxespad=0)
         axes = plt.gca()
         axes.set_xlim([-10, 10])
         axes.set_ylim([-0.8, 0.8])

Out[12]: (-0.8, 0.8)
```



Now we get original Function graph and Taylor Approximate graph on given points!