hw2

April 28, 2018

1 HW2 Coding

This workbook will walk you through the plotting problem 2(f) in HW2. It will also provide familiarity with Jupyter Notebook and Python. Please print (to pdf) a completed version of this workbook for submission with HW2.

ECE C143A/C243A, Spring Quarter 2018, Prof. J.C. Kao, TAs T. Monsoor, X. Jiang and X. Yang

1.1 Import library

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

1.2 Define the function

The function below, `ptc()` accepts the average firing rates at three targets and learns the parameters c_0, c_1, and θ of the tuning curve. Please implement this function below. You may evaluate your execution by running the code under section "Plot the figure," which calculates the tuning parameters using your implementation of `ptc()`. The output should appear reasonable.

```
In [57]: def ptc(y0 , y1 , y2):
             #PTC calculates the tuning curve given average firing rates for certain directions.

             # ============================================================ #
             # YOUR CODE HERE:
             #  The function takes three inputs corresponding to the average
             #  firing rate of a neuron during a reach to 0 degrees (y0), 120
             #  degrees (y1) and 240 degrees (y2). The outputs, c0, c1, and
             #  theta0 are the parameters of the tuning curve.
             # ============================================================ #
             k0 = (y1+y2+y0)/3
             k1 = (y1-y2)/(np.sqrt(3))
             k2 = y0 - ((y1+y2+y0)/3)
             c0 = k0
             theta0 = np.arctan(k1/k2)
             c1 = k1/ np.sin(theta0)
             theta0 = theta0*180/(np.pi)
```

```
        # ================================================================ #
        # END YOUR CODE HERE
        # ================================================================ #

        return c0,c1,theta0
```

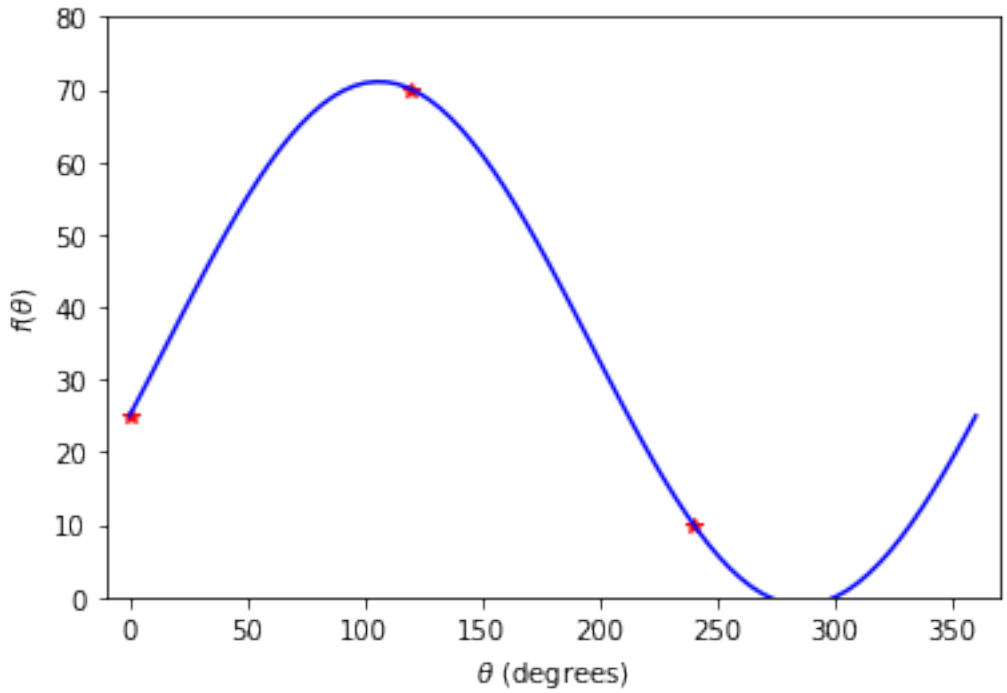
1.3 Plot the figure

The following cells execute your PTC function, printing out the values and plotting the tuning curve.

```
In [58]: c0,c1,theta0=ptc(25,70,10)
         print('c0 = ', c0)
         print('c1 = ', c1)
         print('theta0 = ', theta0)
```

```
c0 =  35.0
c1 =  -36.0555127546
theta0 =  -73.897886248
```

```
In [59]: theta = np.linspace(0, 2*np.pi, num=80)
         plt.plot([0,120,240],[25,70,10],'r*',10)
         plt.plot(theta * 180 / np.pi,c0 + c1 *np.cos(theta - theta0 * np.pi/180),'b',2)
         plt.xlim ([-10 ,370])
         plt.ylim ([0,80])
         plt.xlabel(r'$\theta$ (degrees)');
         plt.ylabel(r'$f(\theta)$');
         plt.show()
```




```
In [25]: a = np.array([[1, np.sin(0), np.cos(0)],
                       [1, np.sin((60*np.pi)/180), np.cos((60*np.pi)/180)],
                       [1, np.sin((120*np.pi)/180),np.cos((120*np.pi)/180)],
                       [1, np.sin((np.pi)), np.cos((np.pi))],
                       [1, np.sin((240*np.pi)/180), np.cos((240*np.pi)/180)],
                       [1, np.sin((300*np.pi)/180), np.cos((300*np.pi)/180)]
                      ])
         b = np.array([25, 40, 70, 30, 10, 15])
         k0,k1,k2 = np.linalg.lstsq(a,b)[0]
         c0 = k0
         theta0 = np.arctan(k1/k2)
         c1 = k1/ np.sin(theta0)
         theta0 = theta0*180/(np.pi)
         print('c0 = ',c0)
         print('c1 = ',c1)
         print('theta0 = ',theta0)

c0 =  31.6666666667
c1 =  -25.2212432507
theta0 =  -76.6271741919


In [17]: theta = np.linspace(0, 2*np.pi, num=80)
         plt.plot([0,60,120,180,240,300],[25, 40, 70, 30, 10, 15],'r*',10)
```

```
plt.plot(theta * 180 / np.pi,c0 + c1 *np.cos(theta - theta0 * np.pi/180),'b',2)
plt.xlim ([-10 ,370])
plt.ylim ([0,80])
plt.xlabel(r'$\theta$ (degrees)');
plt.ylabel(r'$f(\theta)$');
plt.show()
```