In [1]:

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

In [2]:

```
def func(X,y,w,b,lambdal):
    n,d = X.shape
    f = 0.0
    for i in range(n):
        inner = 0
        inner  = 1 - (np.dot(X[i],w) + b)*y[i]
        if inner > 0:
            t = inner **2
        else:
            t = 0
        f +=t
    #print (1/n*f + lambdal*(np.linalg.norm(w)**2))
    return 1/n*f + lambdal*(np.linalg.norm(w)**2)
```

In [3]:

```
def gradient(X,y,w,b,lambdal):
    n,d = X.shape
    g_w = np.zeros(d)
    g_b = 0.0
    for i in range(n):
        inner  = 1 - (np.inner(X[i],w) + b)*y[i]
        if inner > 0:
            t = inner
        else:
            t = 0
        g_w += t*(y[i]*X[i])
        g_b += t*y[i]
    g_w = -2*g_w/n + 2*lambdal*w
    g_b = -2*g_b/n
    #print ("g_w is : ", g_w)
    return g_w, g_b
```

In [4]:

```
def fastgd(X,y,w,b,lambdal, gtype, theta = 0, maxiter = 1000):
    z_w = w
    z_b = b

    f_list = []
    g_list = []
    L= 2*lambdal + 2 * 3

    #print ('eta is ', eta)
    for i in range(maxiter):
        f = func(X,y,w,b,lambdal)
        g_w, g_b = gradient(X,y,w,b,lambdal)

        g = np.append(g_w,g_b)
        #print("g is ", g)
        f_list.append(f)
        g_list.append(np.linalg.norm(g))

        #eta = 1/(math.sqrt(2)+2*lambdal)
        eta = 1/(L)
        z_w0 = z_w
        z_b0 = z_b
        z_w = w - eta*g_w
        z_b = b - eta*g_b
        if (gtype == 'gd' ):

            w = z_w
            b = z_b
            #print(g)
            #print ("i = ", i, ' and, w is: ', w, " and b is ", b)

        elif (gtype == 'fast'):
            old_theta = theta
            theta = (1+math.sqrt(1+4*old_theta**2))/2
            w = z_w + (old_theta-1)/theta*(z_w - z_w0)
            b = z_b + (old_theta-1)/theta*(z_b - z_b0)
        elif (gtype == 'opt'):
            old_theta = theta
            if (i < maxiter):
                theta = (1+math.sqrt(1+4*old_theta**2))/2
            else:
                theta = (1+math.sqrt(1+8*old_theta**2))/2


            w = z_w + (old_theta-1)/theta*(z_w - z_w0) + (old_theta)/t
            b = z_b + (old_theta-1)/theta*(z_b - z_b0) + (old_theta)/t

    return w, b, f_list, g_list
```

In [5]:

```
X = [[1,1],[1,-1],[-1,1],[-1,-1]]
X = np.array(X)

Y = np.array([1,1,1,-1])

w = np.array([0,0])
b = 0
```

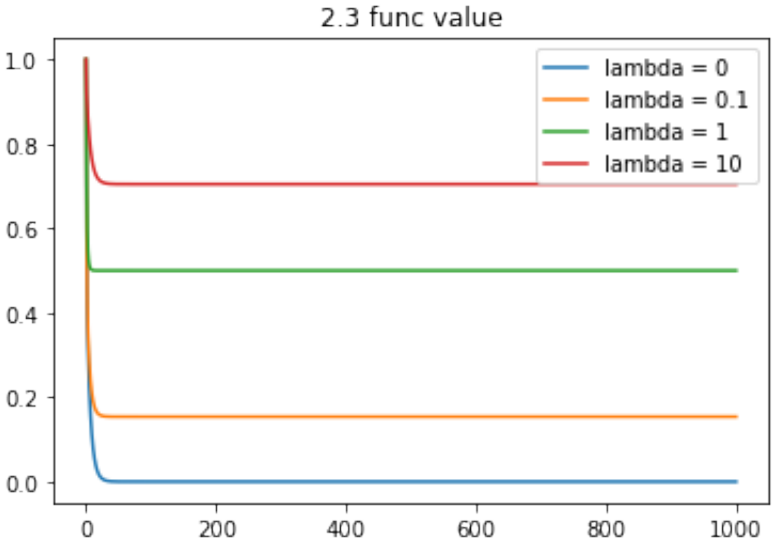
In [6]:

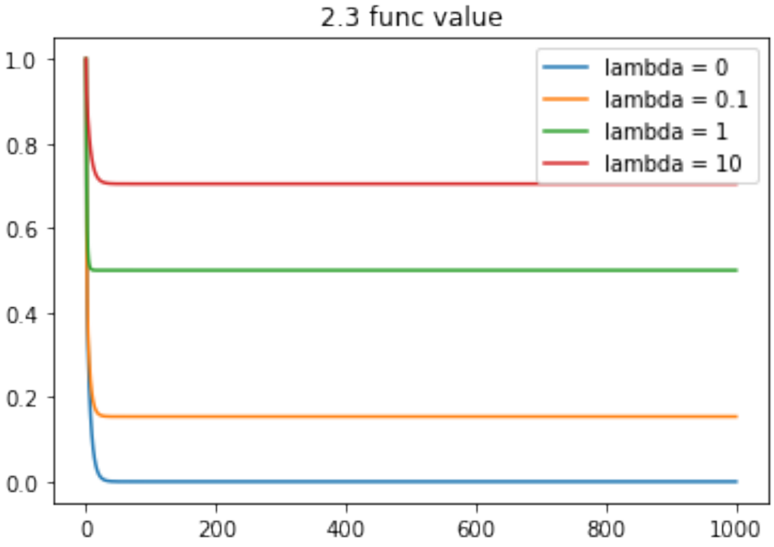
```
x_point = np.array([1,1,-1,-1])
y_point = np.array([1,-1,1,-1])
label = Y
```

In [7]:

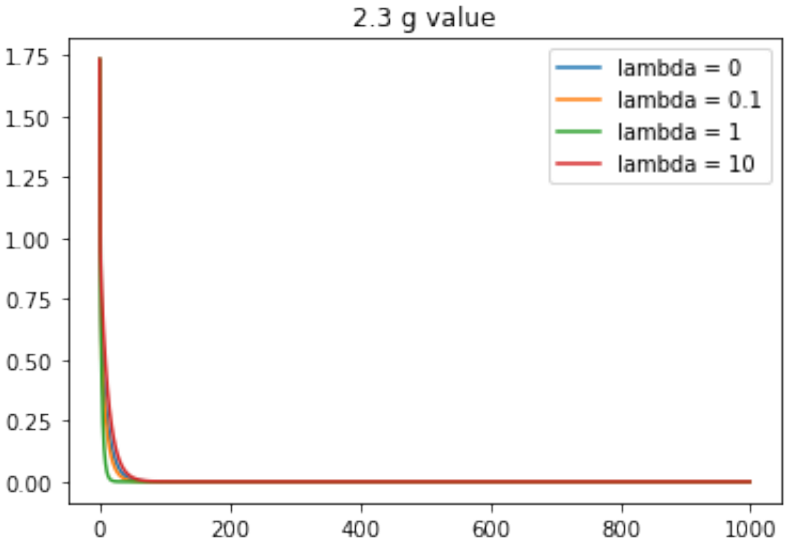
```
#2.3
w_1, b_1, f_1, g_1 = fastgd(X,Y,w,b,1, 'gd', theta = 0, maxiter = 1000
w_0, b_0, f_0, g_0 = fastgd(X,Y,w,b,0, 'gd', theta = 0, maxiter = 1000
w_01, b_01, f_01, g_01 = fastgd(X,Y,w,b,0.1, 'gd', theta = 0, maxiter
w_10, b_10, f_10, g_10 = fastgd(X,Y,w,b,10, 'gd', theta = 0, maxiter =
```

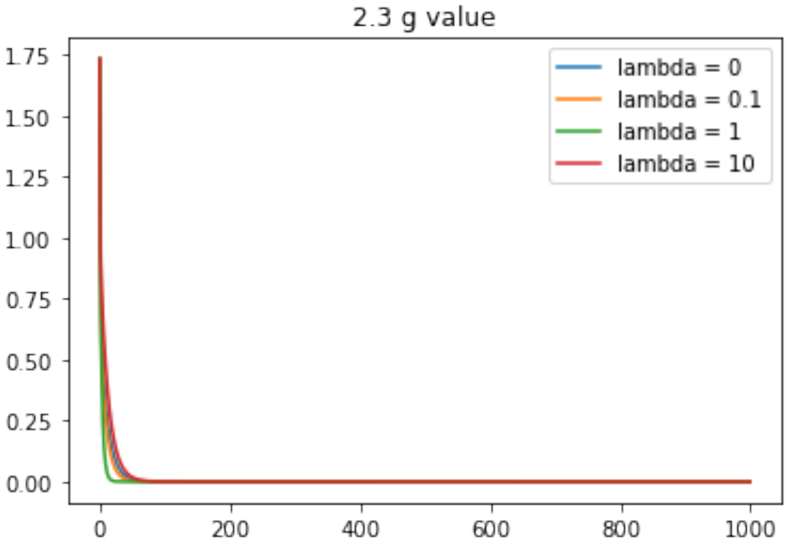
In [8]:
```
plt.plot(range(len(f_0)), f_0)
plt.plot(range(len(f_01)), f_01)
plt.plot(range(len(f_1)), f_1)
plt.plot(range(len(f_10)), f_10)
plt.gca().legend(('lambda = 0','lambda = 0.1','lambda = 1','lambda = 1
plt.title("2.3 func value")
plt.show()
```





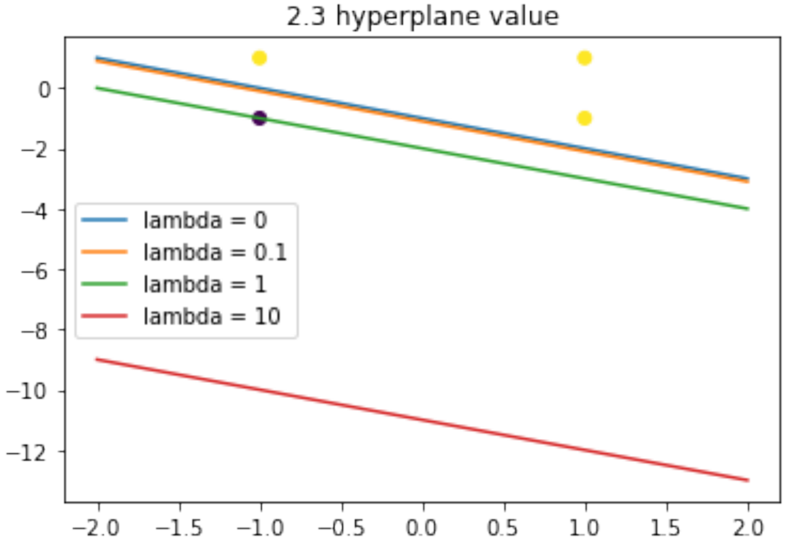
In [9]:
```
plt.plot(range(len(g_0)), g_0)
plt.plot(range(len(g_01)), g_01)
plt.plot(range(len(g_1)), g_1)
plt.plot(range(len(g_10)), g_10)
plt.gca().legend(('lambda = 0','lambda = 0.1','lambda = 1','lambda = 1
plt.title("2.3 g value")
plt.show()
```

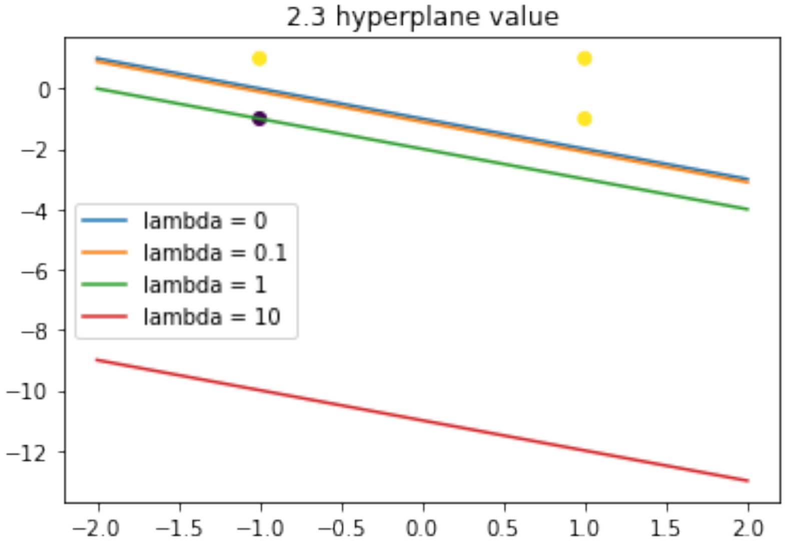




In [10]:

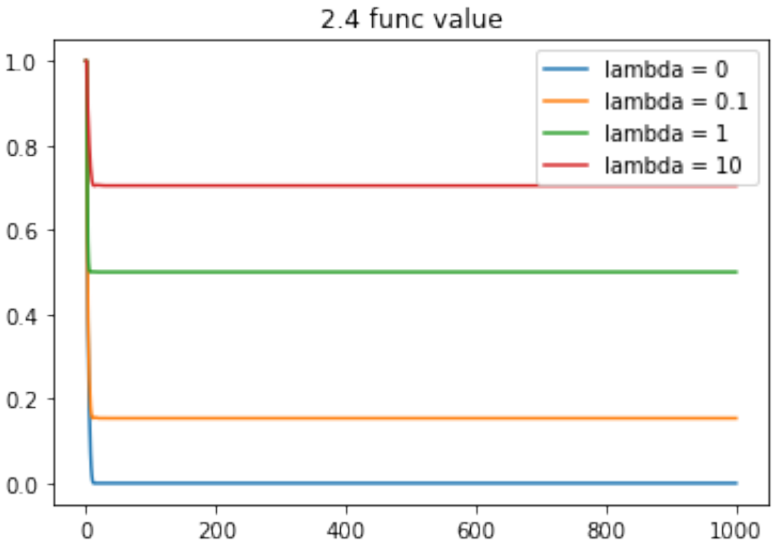
```
plt.scatter(x_point, y_point, c = label)
x = np.linspace(-2,2,120)
y_0 = -w_0[0]/w_0[1]*x -b_0/w_0[1]
y_01 = -w_01[0]/w_01[1] *x-b_01/w_01[1]
y_1 = -w_1[0]/w_1[1]*x -b_1/w_1[1]
y_10 = -w_10[0]/w_10[1]*x -b_10/w_10[1]
plt.plot(x,y_0)
plt.plot(x,y_01)
plt.plot(x,y_1)
plt.plot(x,y_10)
plt.gca().legend(('lambda = 0','lambda = 0.1','lambda = 1','lambda = 1
plt.title("2.3 hyperplane value")

plt.show()
```

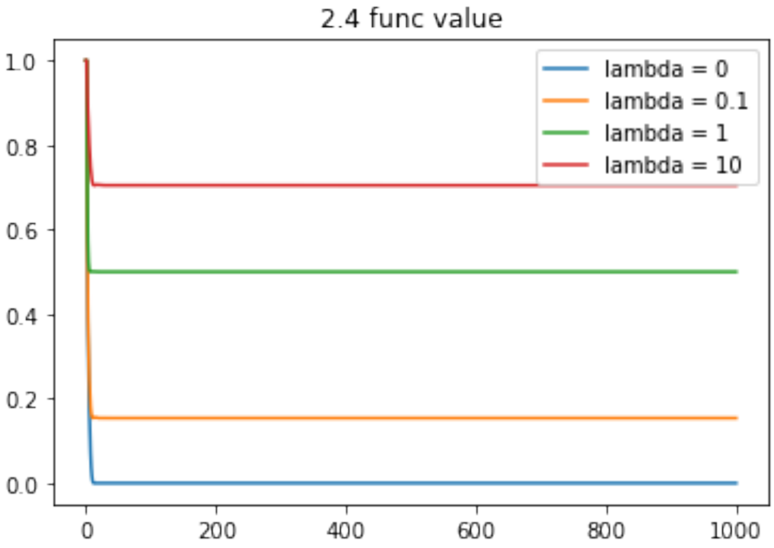




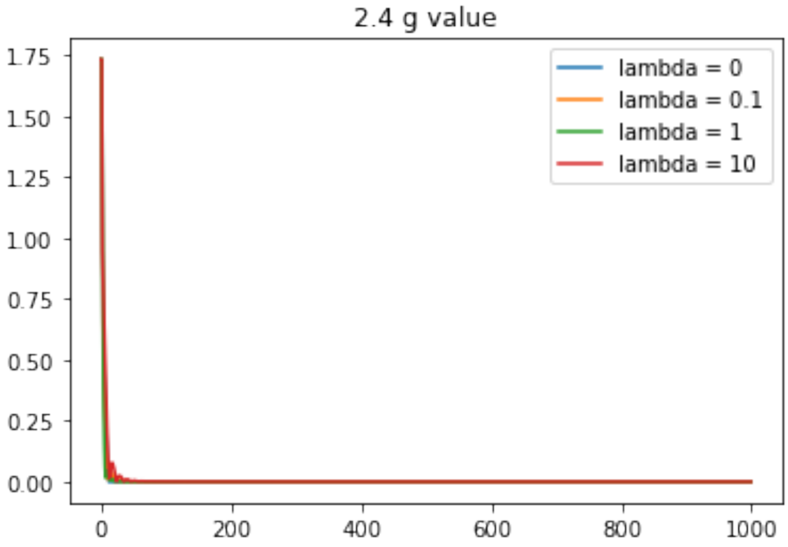
In [11]:
```
#2.4
w_1, b_1, f_1, g_1 = fastgd(X,Y,w,b,1, 'fast', theta = 0, maxiter = 10
w_0, b_0, f_0, g_0 = fastgd(X,Y,w,b,0, 'fast', theta = 0, maxiter = 10
w_01, b_01, f_01, g_01 = fastgd(X,Y,w,b,0.1, 'fast', theta = 0, maxite
w_10, b_10, f_10, g_10 = fastgd(X,Y,w,b,10, 'fast', theta = 0, maxiter
```

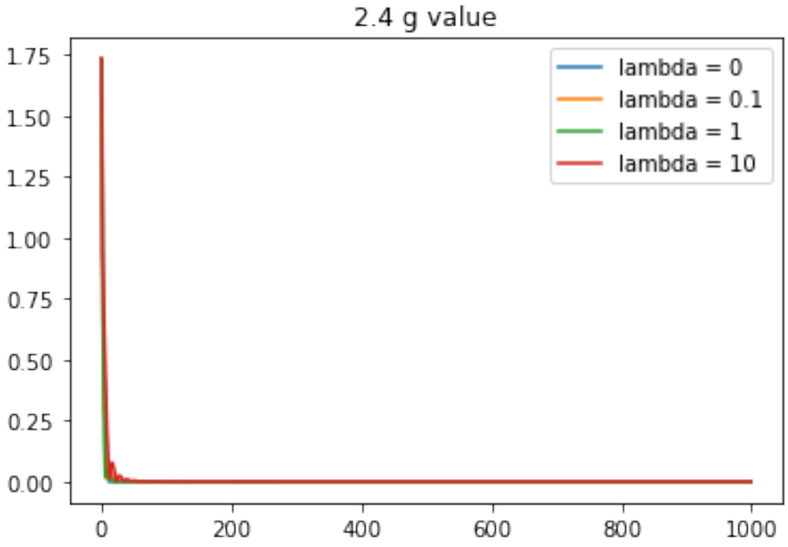
In [12]:
```
plt.plot(range(len(f_0)), f_0)
plt.plot(range(len(f_01)), f_01)
plt.plot(range(len(f_1)), f_1)
plt.plot(range(len(f_10)), f_10)
plt.gca().legend(('lambda = 0','lambda = 0.1','lambda = 1','lambda = 1
plt.title("2.4 func value")
plt.show()
```



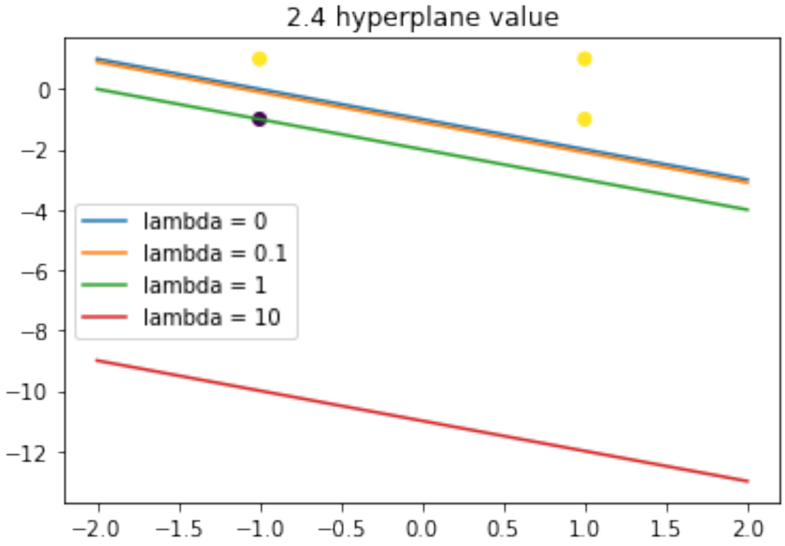


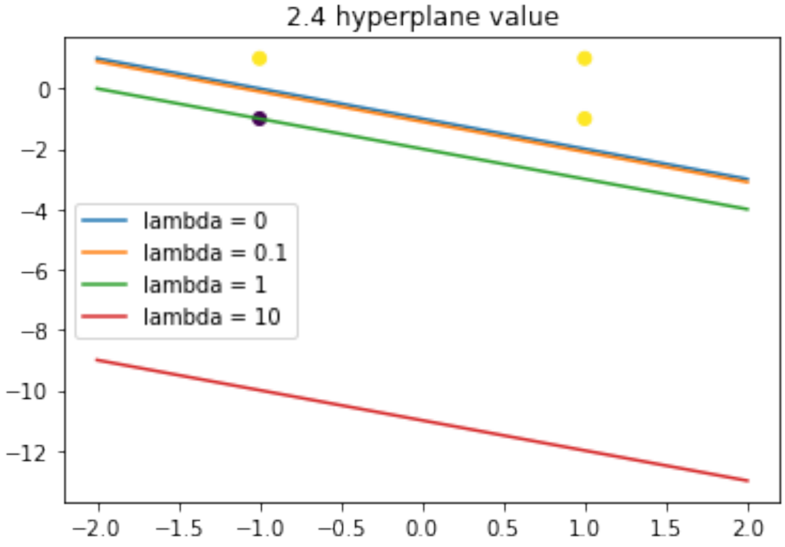
In [13]:
```
plt.plot(range(len(g_0)), g_0)
plt.plot(range(len(g_01)), g_01)
plt.plot(range(len(g_1)), g_1)
plt.plot(range(len(g_10)), g_10)
plt.gca().legend(('lambda = 0','lambda = 0.1','lambda = 1','lambda = 1
plt.title("2.4 g value")
plt.show()
```





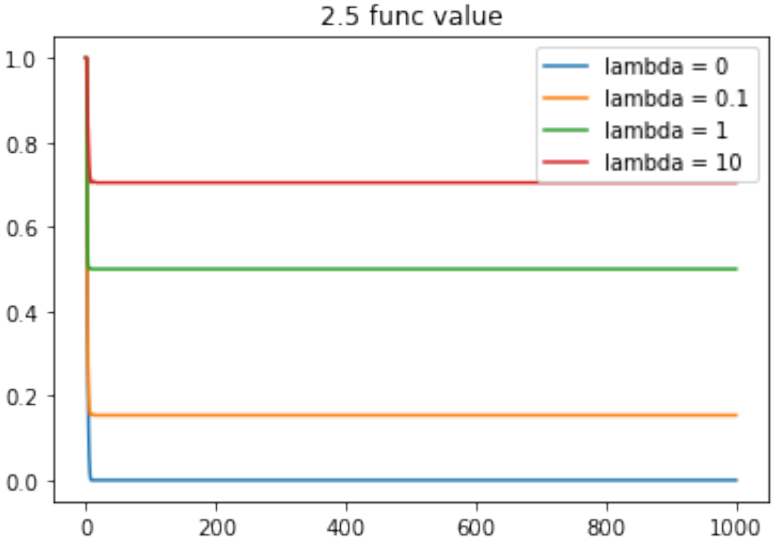
In [14]:
```
plt.scatter(x_point, y_point, c = label)
x = np.linspace(-2,2,120)
y_0 = -w_0[0]/w_0[1]*x -b_0/w_0[1]
y_01 = -w_01[0]/w_01[1] *x-b_01/w_01[1]
y_1 = -w_1[0]/w_1[1]*x -b_1/w_1[1]
y_10 = -w_10[0]/w_10[1]*x -b_10/w_10[1]
plt.plot(x,y_0)
plt.plot(x,y_01)
plt.plot(x,y_1)
plt.plot(x,y_10)
plt.gca().legend(('lambda = 0','lambda = 0.1','lambda = 1','lambda = 1
plt.title("2.4 hyperplane value")

plt.show()
```

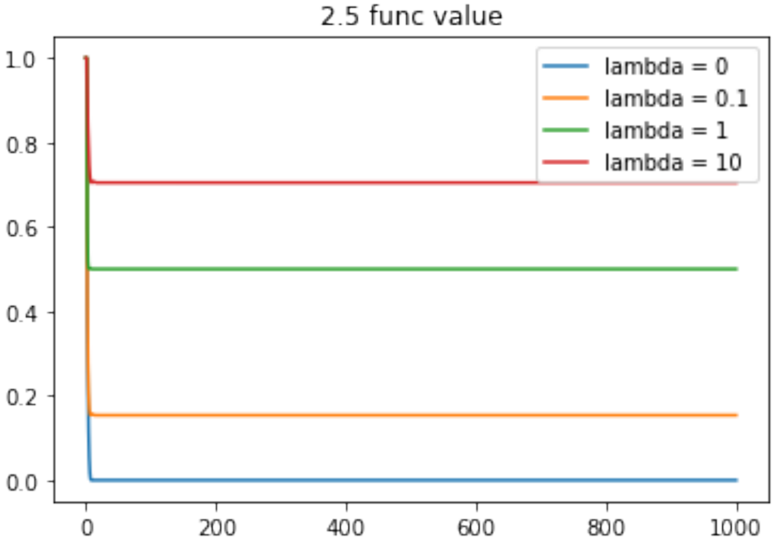




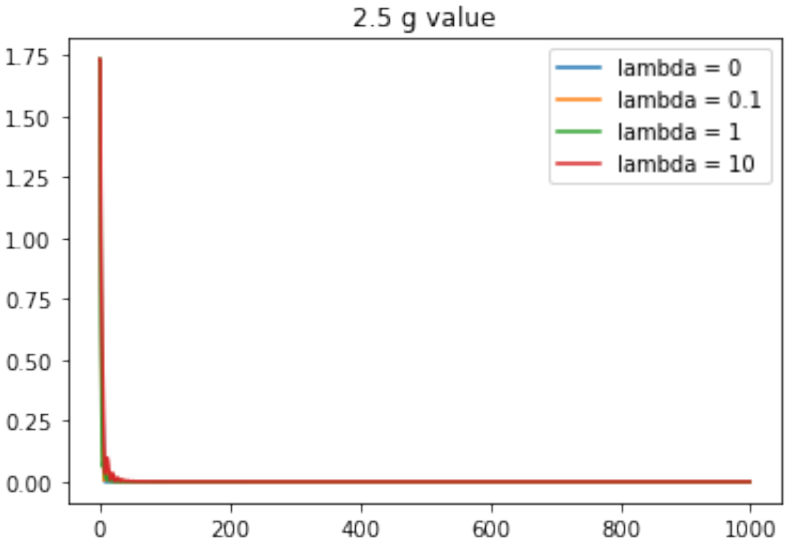
In [15]:
```
#2.5
w_1, b_1, f_1, g_1 = fastgd(X,Y,w,b,1, 'opt', theta = 0, maxiter = 100
w_0, b_0, f_0, g_0 = fastgd(X,Y,w,b,0, 'opt', theta = 0, maxiter = 100
w_01, b_01, f_01, g_01 = fastgd(X,Y,w,b,0.1, 'opt', theta = 0, maxiter
w_10, b_10, f_10, g_10 = fastgd(X,Y,w,b,10, 'opt', theta = 0, maxiter
```

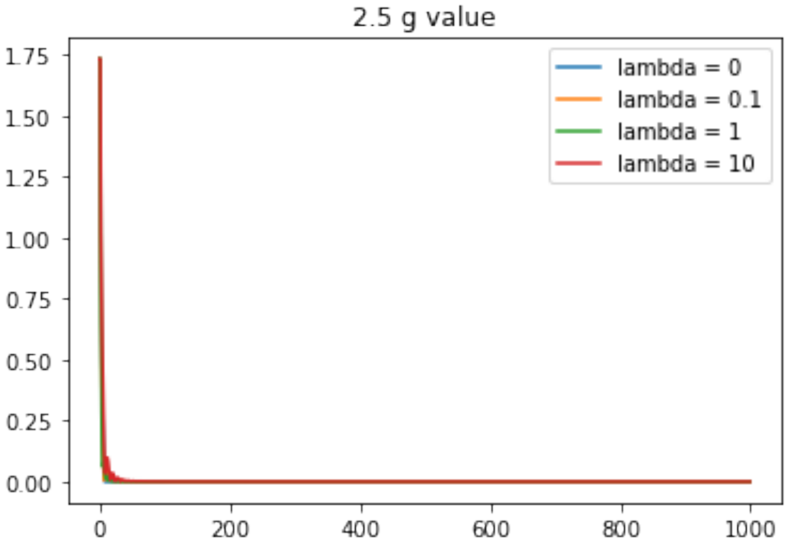
In [16]:
```
plt.plot(range(len(f_0)), f_0)
plt.plot(range(len(f_01)), f_01)
plt.plot(range(len(f_1)), f_1)
plt.plot(range(len(f_10)), f_10)
plt.gca().legend(('lambda = 0','lambda = 0.1','lambda = 1','lambda = 1
plt.title("2.5 func value")
plt.show()
```





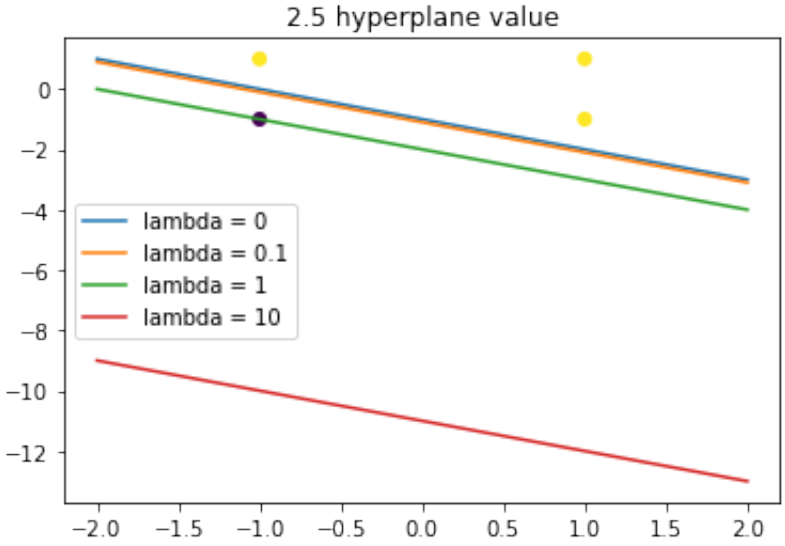
In [17]:
```
plt.plot(range(len(g_0)), g_0)
plt.plot(range(len(g_01)), g_01)
plt.plot(range(len(g_1)), g_1)
plt.plot(range(len(g_10)), g_10)
plt.gca().legend(('lambda = 0','lambda = 0.1','lambda = 1','lambda = 1
plt.title("2.5 g value")
plt.show()
```

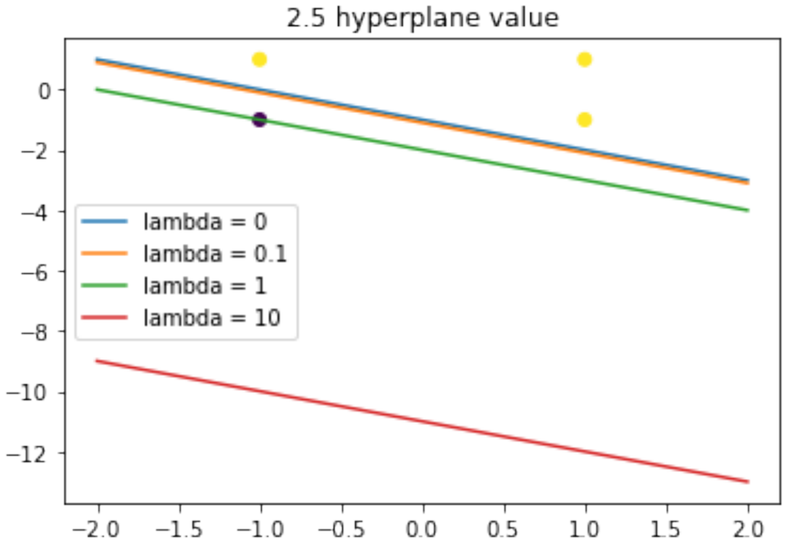




In [18]:

```
plt.scatter(x_point, y_point, c = label)
x = np.linspace(-2,2,120)
y_0 = -w_0[0]/w_0[1]*x -b_0/w_0[1]
y_01 = -w_01[0]/w_01[1] *x-b_01/w_01[1]
y_1 = -w_1[0]/w_1[1]*x -b_1/w_1[1]
y_10 = -w_10[0]/w_10[1]*x -b_10/w_10[1]
plt.plot(x,y_0)
plt.plot(x,y_01)
plt.plot(x,y_1)
plt.plot(x,y_10)
plt.gca().legend(('lambda = 0','lambda = 0.1','lambda = 1','lambda = 1
plt.title("2.5 hyperplane value")

plt.show()
```





In [ ]: