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

N, p = 30, 20
np.random.seed(0)
X = np.random.randn(N,p)
Y = 2*np.random.randint(2, size = N) - 1

Theta = np.random.randn(20)
lamb = 0.1

def calculate_loss(x,y,theta):
    return np.average(np.max( 1-y*(x@theta) , 0)) + lamb*np.sum(theta**2)

max_iter = 10000
lr = 0.001

# store loss each iteration
loss = []

# non-differentiability check variable
cnt = 0

for i in range(max_iter):
    # choose random 1 data
    index = np.random.randint(0,N)

    # just temporary instance
    val = Y[index]*X[index]@Theta

    if val >1 : gradient = 0
    elif val == 1:  # ReLU(0) moment
        gradient = 0
        cnt += 1
        loss.append(loss[-1])
        continue
    else: gradient = -Y[index]*X[index]

    # regularization part
    gradient += 2*lamb*Theta

    loss.append(calculate_loss(X,Y,Theta))
    Theta -= lr*gradient

print('non-differentiability count:',cnt)
print()
print(Theta)

plt.plot(np.arange(0,max_iter) , loss)
plt.savefig('p2_loss.png')
plt.show()
```

Result
non-differentiability count: 0

**[0.08378149 0.06544954 -0.41543206 0.22583367 -0.11944054 -0.18471714
 -0.34162701 -0.19718943 0.39294682 -0.12898355 0.04627642 0.04557828
  0.2061068 -0.17737847 0.2216098   0.05594214 -0.3023263 -0.08892688
 -0.06343256 -0.40010015]**