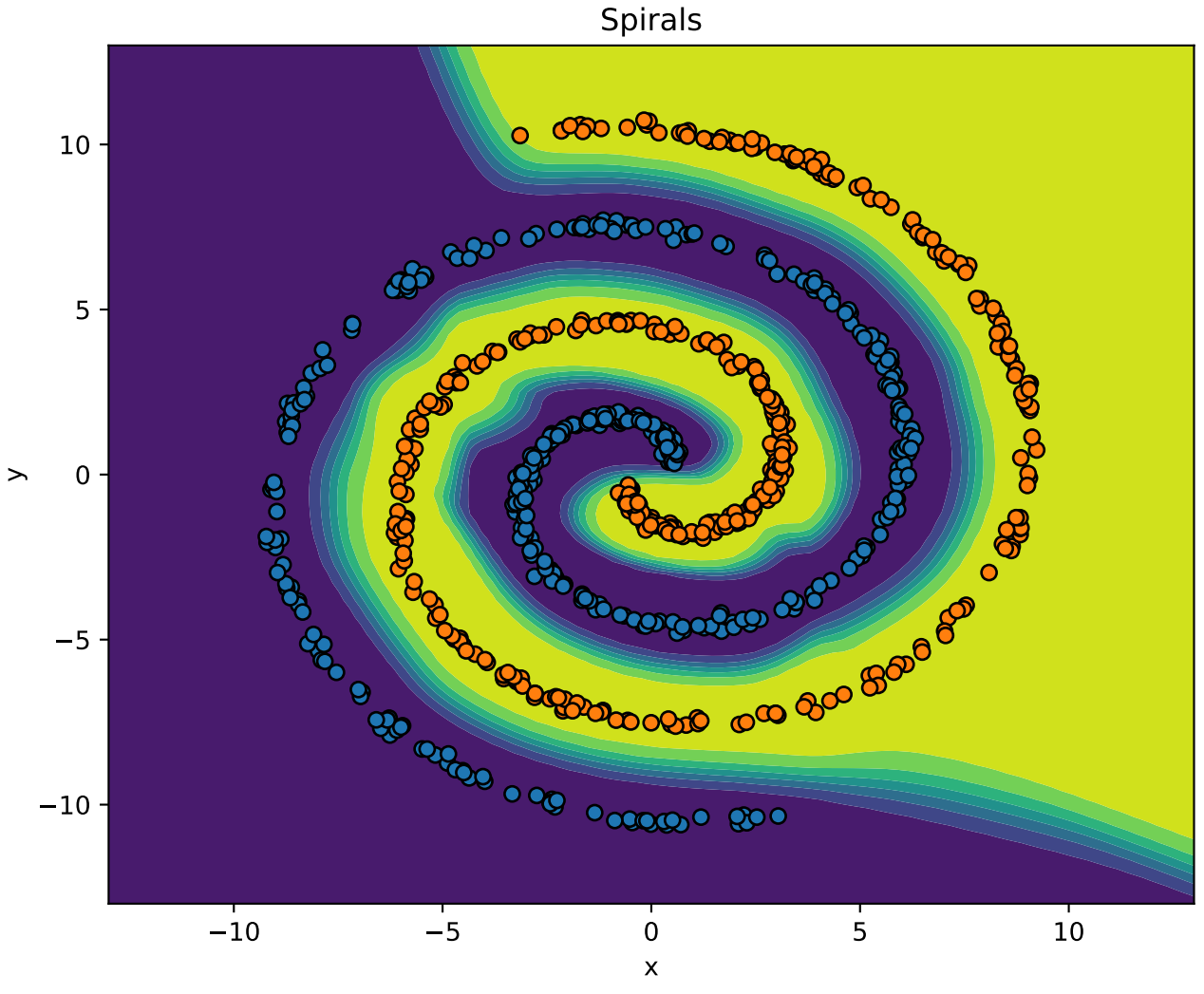
```
#Camille Chow
#ECE 471 Assignment 2

import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tqdm import tqdm

BATCH_SIZE = 32
NUM_BATCHES = 10000
NUM_SAMP = 400

class Data():
    def __init__(self):
        sigma = 0.1
        np.random.seed()

        #training data for label 0
        self.index = np.arange(NUM_SAMP)
        self.t0 = np.random.uniform(.1, 1.8, NUM_SAMP)
        self.x0 = np.atleast_2d(6 * self.t0 * np.cos(2*np.pi*self.t0) + np.random.normal(
0, sigma, NUM_SAMP)).T
        self.y0 = np.atleast_2d(6 * self.t0 * np.sin(2*np.pi*self.t0) + np.random.normal(
0, sigma, NUM_SAMP)).T
        self.class0 = np.hstack((self.x0, self.y0))
        #training data for label 1
        self.t1 = np.random.uniform(.1, 1.8, NUM_SAMP)
        self.x1 = np.atleast_2d(6 * self.t1 * np.cos(2*np.pi*self.t1 + np.pi) + np.random
.normal(0, sigma, NUM_SAMP)).T
        self.y1 = np.atleast_2d(6 * self.t1 * np.sin(2*np.pi*self.t1 + np.pi) + np.random
.normal(0, sigma, NUM_SAMP)).T
        self.class1 = np.hstack((self.x1, self.y1))

        self.labels = np.hstack((np.zeros(NUM_SAMP), np.ones(NUM_SAMP)))
        self.coords = np.vstack((self.class0, self.class1))

    def get_batch(self):
        index = np.arange(2 * NUM_SAMP)
        choices = np.random.choice(index, size=BATCH_SIZE)

        return self.coords[choices], self.labels[choices]

def f(x):   #takes x in R2 and outputs logits and sigmoid
    layer1_size = 50
    layer2_size = 50

    #trainable variables
    w1 = tf.get_variable('w1', [2, layer1_size], tf.float32, tf.random_normal_initializer
())
    b1 = tf.get_variable('b1', [layer1_size], tf.float32, tf.zeros_initializer())
    w2 = tf.get_variable('w2', [layer1_size, layer2_size], tf.float32, tf.random_normal_i
nitializer())
    b2 = tf.get_variable('b2', [layer2_size], tf.float32, tf.zeros_initializer())
    w3 = tf.get_variable('w3', [layer2_size, 1], tf.float32, tf.random_normal_initializer
())
    b3 = tf.get_variable('b3', [], tf.float32, tf.zeros_initializer())

    #perceptron
    layer1 = tf.add(tf.matmul(x, w1), b1)
    layer2 = tf.add(tf.matmul(tf.nn.elu(layer1), w2), b2)
    out = tf.add(tf.matmul(tf.nn.elu(layer2), w3), b3)

    return tf.squeeze(out), tf.squeeze(tf.sigmoid(out))

x = tf.placeholder(tf.float32, [None, 2])
t = tf.placeholder(tf.float32, [None])
logits, t_hat = f(x)

#optimization functions
lam = .001
```

```
loss = tf.losses.sigmoid_cross_entropy(t, logits) + tf.add_n([lam * tf.nn.l2_loss(v) for
v in tf.trainable_variables()])
optim = tf.train.MomentumOptimizer(learning_rate=.01, momentum=.95).minimize(loss)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

data = Data()

#training
for _ in tqdm(range(0, NUM_BATCHES)):
    x_np, t_np = data.get_batch()
    loss_np, _ = sess.run([loss, optim], feed_dict={x: x_np, t: t_np})

xaxis = np.linspace(-13, 13, 100)
yaxis = np.linspace(-13, 13, 100)
xx, yy = np.meshgrid(xaxis,yaxis)
coords = np.array(list(zip(xx.flatten(), yy.flatten())))

z = sess.run(t_hat, feed_dict={x: coords})
z = np.reshape(z, [100, 100])

plt.figure(1, figsize=[13,13])

plt.contourf(xx, yy, z)

plt.scatter(data.x0,data.y0, edgecolor='k')
plt.scatter(data.x1,data.y1, edgecolor='k')

plt.xlabel('x')
plt.ylabel('y')
plt.title('Spirals')

plt.show()
```


Spirals
10
5
0
−5
−10
y
−10
−5
0
5
10
x