Deep Learning Assignment 2

Nikita Teplitskiy

September 17, 2020

1 Introduction

This assignment was completed using the Keras library by constructing a perceptron with two hidden layers, each containing 16 neurons. The model is able to reliably classify its training data set with perfect accuracy and performs well on unfamiliar data. The form of the perceptron function can be seen in Figure 1; its outline was generated by plotting a selection of points from the X/Y plane on a contour plot.

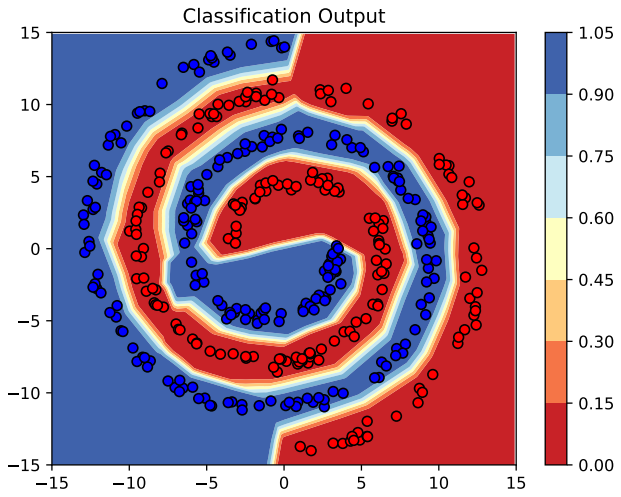

Figure 1: Output of multilayer perceptron classifier

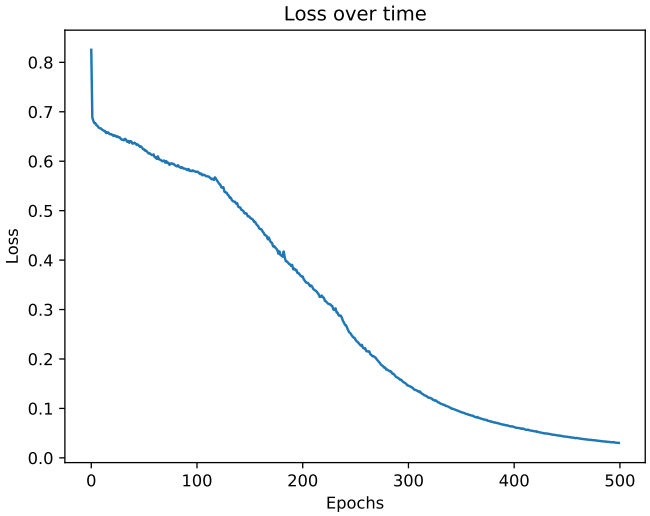

Figure 2: Loss over training iterations

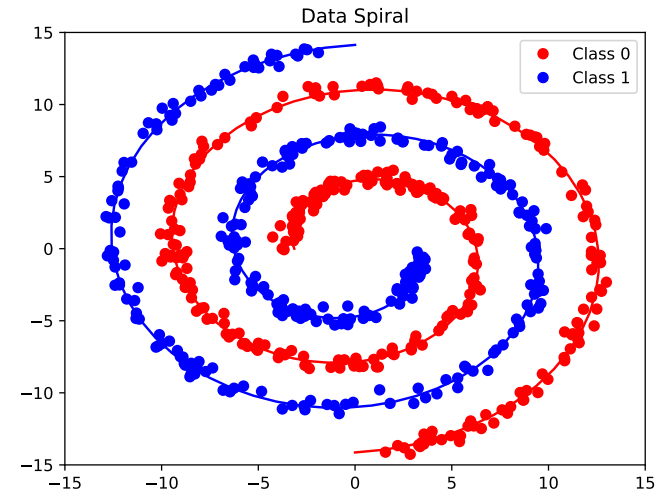

Figure 3: Starting Data

2 Code

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np
import matplotlib.pyplot as plt

def pol2cart(r,th):
    x = r * np.cos(th)
    y = r * np.sin(th)
    return x, y

def gen_data(n, noise=0.0):
    th = np.random.uniform(-np.pi, -9/2 * np.pi, size=[n,1])
    cl = np.random.randint(0,2,size=[n,1])
    fuzz = np.random.normal(scale=noise, size=[n,2])
    return np.hstack(pol2cart(th*(2*cl-1), th)) + fuzz, cl

#generate a clean spiral
th = np.linspace(-np.pi, -9/2 * np.pi, num=100)
plt.plot(*pol2cart(th,th), 'b')
plt.plot(*pol2cart(-th,th), 'r')

#generate and plot some noisy data
[x_test,y_test] = gen_data(600, .3)
y_test = y_test.T[0]
plt.plot(x_test[y_test==0,0], x_test[y_test==0,1], 'ro', label='Class 0')
plt.plot(x_test[y_test==1,0], x_test[y_test==1,1], 'bo', label='Class 1')
plt.title("Data Spiral")
plt.xlim([-15,15]); plt.ylim([-15,15])
plt.legend()
plt.savefig('data.pdf', format='pdf')
#plt.show()
plt.clf()

#tunable parameters
num_data = 1000
noise_level = 0.3
num_epochs = 500

#create training data
[x,y] = gen_data(num_data, noise_level)

#generate keras model
model = keras.Sequential(
    [
        layers.Dense(16, input_dim=2, activation="relu", name="hidden1"),
        layers.Dense(16, input_dim=2, activation="relu", name="hidden2"),
        layers.Dense(1,  activation='sigmoid', name="out", kernel_regularizer='l2'),
    ]
```

```
)
#run model
model.compile(loss='binary_crossentropy', optimizer='Adam', metrics=['accuracy'])
fit_data = model.fit(x,y,epochs=num_epochs,batch_size=10,verbose=0)

#loss statistics
model.summary()
plt.plot(fit_data.history['loss'])
plt.title("Loss over time")
plt.xlabel('Epochs'); plt.ylabel('Loss')
plt.savefig('loss.pdf', format='pdf')
plt.show()
plt.clf()

#Check that training data is classified correctly
results = model.evaluate(x, y, batch_size=10)
print("training loss, test acc:", results)

#result plot
[x_check,y_check] = gen_data(400,noise_level)
y_thresh = np.round(model.predict(x_check).T[0])

plt.plot(x_check[y_thresh==0,0], x_check[y_thresh==0,1], 'ro', markeredgecolor='black')
plt.plot(x_check[y_thresh==1,0], x_check[y_thresh==1,1], 'bo', markeredgecolor='black',)
#plt.show()

x = np.arange(-15, 15, 0.1)
y = np.arange(-15, 15, 0.1)
xx, yy = np.meshgrid(x, y)
xxx = xx.reshape(-1,1)
yyy = yy.reshape(-1,1)
inp = np.hstack((xxx,yyy))
z = model.predict(inp)
plt.contourf(xx,yy,z.reshape(xx.shape), cmap='RdYlBu')
plt.colorbar()
plt.title("Classification Output")
plt.xlim([-15,15]); plt.ylim([-15,15])
plt.savefig('result.pdf', format='pdf')
plt.show()
```