Шишкин Иван, гр. 8383

Задача (Вариант 4).

Необходимо дополнить следующий фрагмент кода моделью ИНС, которая способна провести бинарную классификацию по сгенерированным данным:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as mclr
from tensorflow.keras import layers
from tensorflow.keras import models

def genData(size=500):
    #Функцию выбрать в зависимости от варианта

def drawResults(data, label, prediction):
    p_label = np.array([round(x[0]) for x in prediction])
    plt.scatter(data[:, 0], data[:, 1], s=30, c=label[:, 0],
cmap=mclr.ListedColormap(['red', 'blue']))
    plt.scatter(data[:, 0], data[:, 1], s=10, c=p_label,
cmap=mclr.ListedColormap(['red', 'blue']))
    plt.grid()
    plt.show()

(train_data, train_label), (test_data, test_label) = genData()
#В данном месте необходимо создать модель и обучить ее
#Получение ошибки и точности в процессе обучения
loss = H.history['loss']
val_loss = H.history['val_loss']
acc = H.history['accuracy']
val_acc = H.history['val_accuracy']
epochs = range(1, len(loss) + 1)
#Построение графика ошибки
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
#Построение графика точности
plt.clf()
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
#Получение и вывод результатов на тестовом наборе
results = model.evaluate(test_data, test_label)
print(results)
#Вывод результатов бинарной классификации
all_data = np.vstack((train_data, test_data))
all_label = np.vstack((train_label, test_label))
pred = model.predict(all_data)
drawResults(all_data, all_label, pred)
```

Вариант 4

```
def genData(size=500):
    size1 = size//2
    size2 = size - size1

    t1 = np.random.rand(size1)
    x1 = np.asarray([i * math.cos(i*2*math.pi) + (np.random.rand(1)-
1)/2*i for i in t1])
    y1 = np.asarray([i * math.sin(i*2*math.pi) + (np.random.rand(1)-
1)/2*i for i in t1])
    data1 = np.hstack((x1, y1))
    label1 = np.zeros([size1, 1])
    div1 = round(size1*0.8)

    t2 = np.random.rand(size2)
    x2 = np.asarray([-i * math.cos(i*2*math.pi) + (np.random.rand(1)-
1)/2*i for i in t2])
    y2 = np.asarray([-i * math.sin(i*2*math.pi) + (np.random.rand(1)-
1)/2*i for i in t2])
    data2 = np.hstack((x2, y2))
    label2 = np.ones([size2, 1])
```

```
    div2 = round(size2*0.8)

    div = div1 + div2
    order = np.random.permutation(div)

    train_data = np.vstack((data1[:div1], data2[:div2]))
    test_data = np.vstack((data1[div1:], data2[div2:]))
    train_label = np.vstack((label1[:div1], label2[:div2]))
    test_label = np.vstack((label1[div1:], label2[div2:]))
    return (train_data[order, :], train_label[order, :]), (test_data,
test_label)
```

Выполнение работы.

Дата генерируется по следующему принципу: создается массив из случайных чисел с плавающей точкой от 0 до 1 размеров в size/2 символов. Затем, каждое число из этого массива учувствует в составлении массивов x1 и y1 (где i – число из массива сгенерированных чисел):

$$x1 = \frac{i * \cos(i * 2\pi) + random(1) - 1}{2 * i}$$

$$y1 = \frac{i * \sin(i * 2\pi) + random(1) - 1}{2 * i}$$

Эти 2 столбца x1 и y1 и являются данными data1. Для остальных size/2 символов формула x2 и y2 такова:

$$x2 = \frac{-i * \cos(i * 2\pi) + random(1) - 1}{2 * i}$$

$$y2 = \frac{-i * \sin(i * 2\pi) + random(1) - 1}{2 * i}$$

80% данных берутся для обучения, 20 – для тестирования.

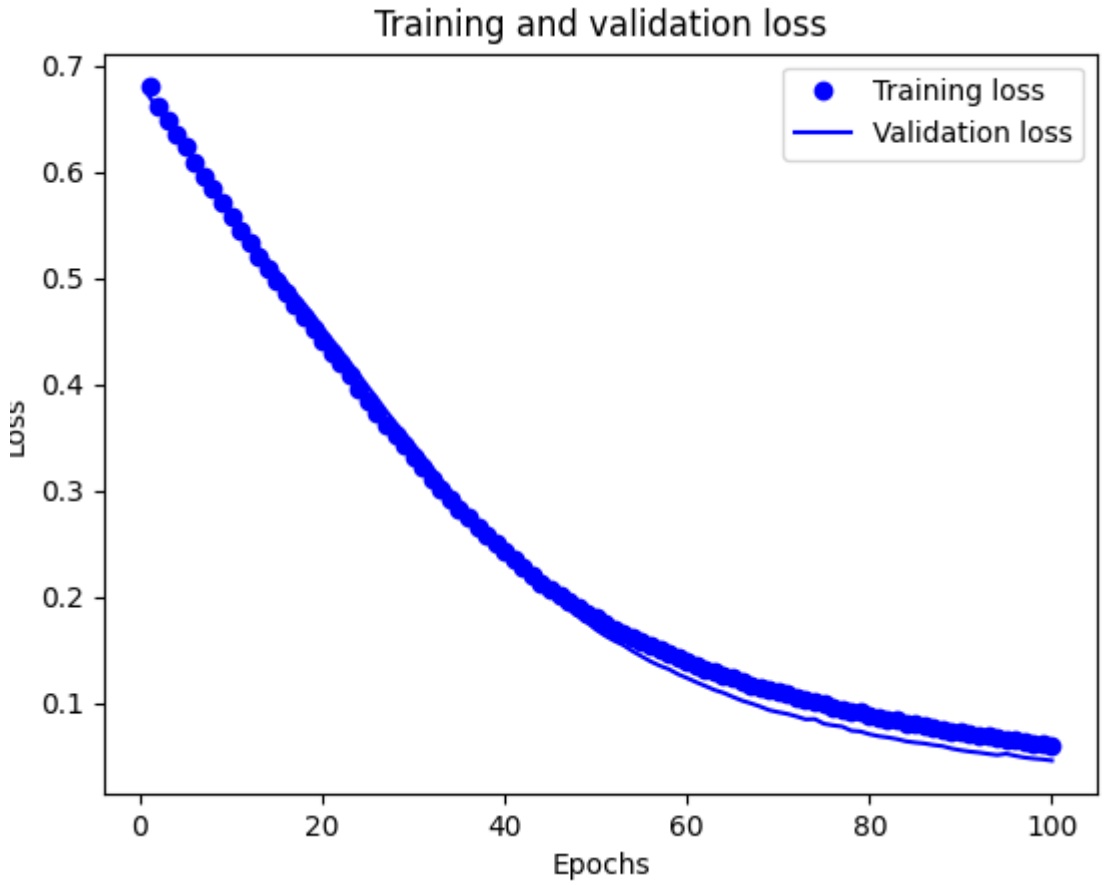
Затем, создается модель, состоящая из двух скрытых слоев, на каждый из которых подается по 32 нейрона (листинг 1).

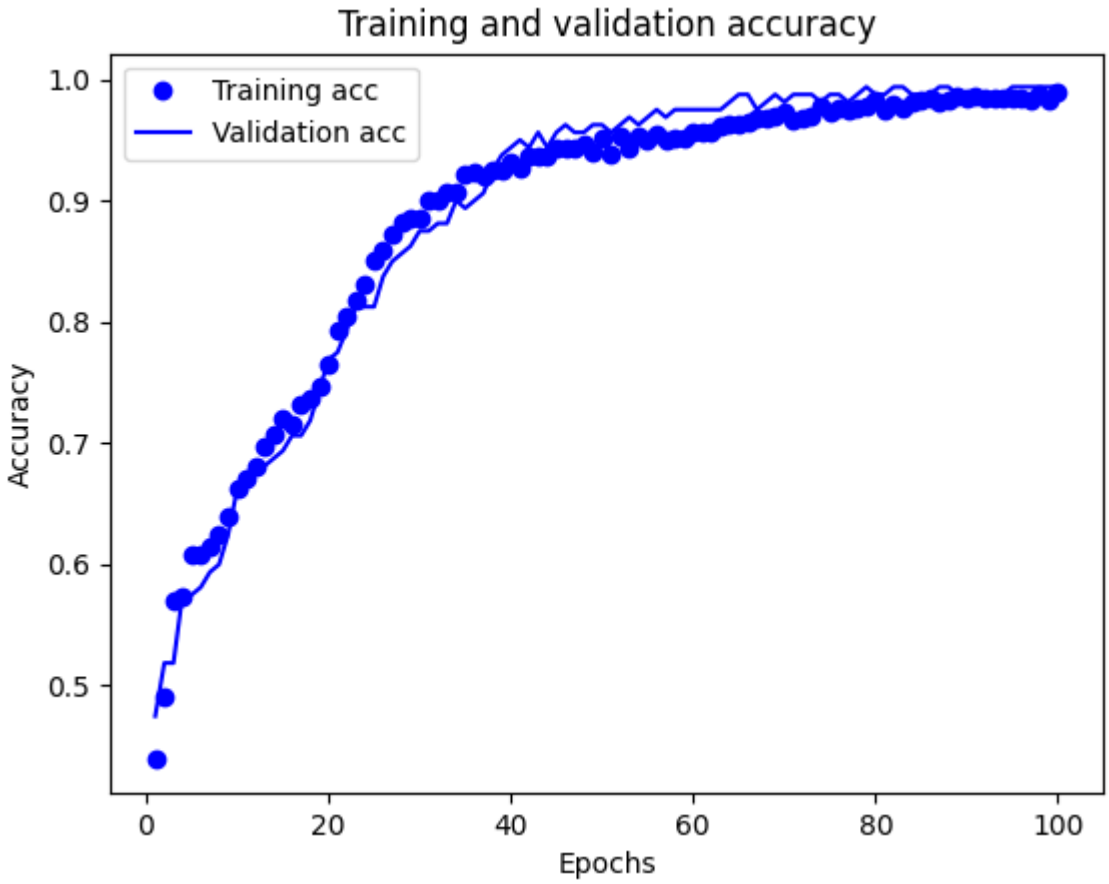
Листинг 1 – Создание модели

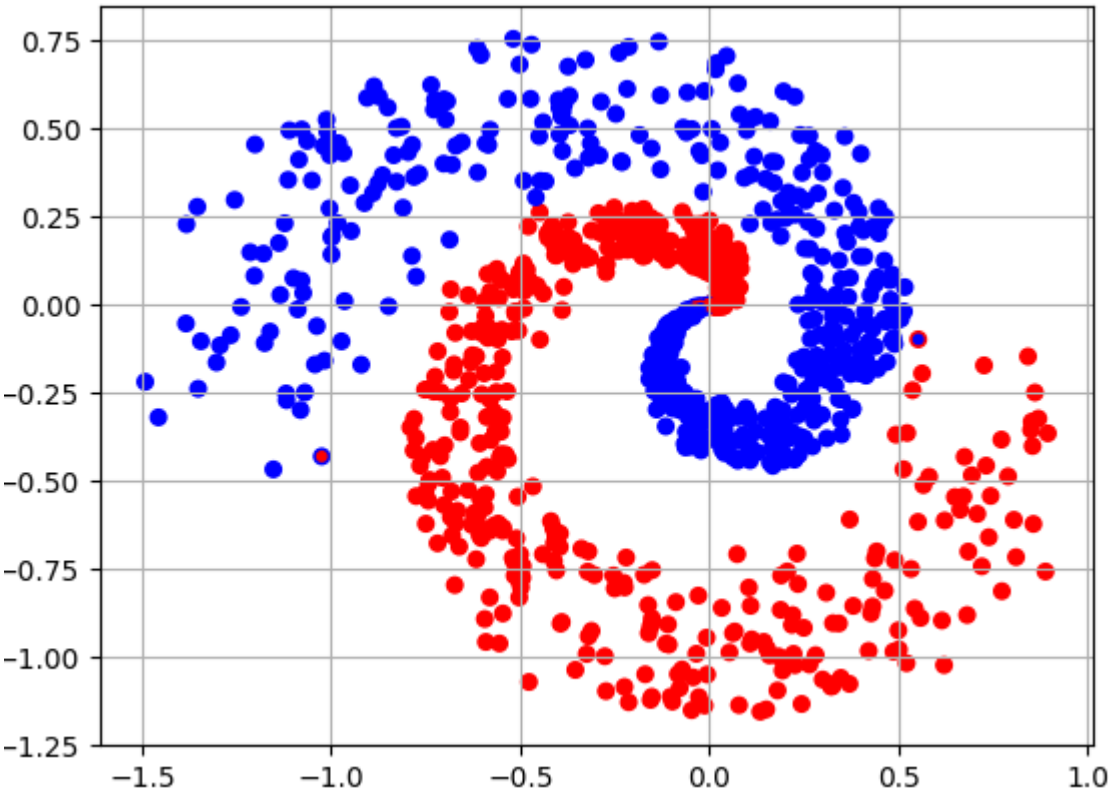
```
    model = models.Sequential()
# добавление слоев
    model.add(layers.Dense(32, activation='relu', input_shape=(2,)))
    model.add(layers.Dense(32, activation='relu'))
```

```
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
H = model.fit(partial_x_train, partial_y_train, epochs=100, batch_size=50, validation_data=(x_val, y_val))
```

В качестве оптимизатора будет использоваться RMSProp, функцией потерь бинарная кросс-энтропия (функция, которая в основном используется при бинарной классификации), а в качестве метрики используется точность. Обучение производится в течение 100 эпох пакетами по 50 образцов. В итоге получились графики, изображенные ниже.







Можно заметить, что точность модели приближается ко 100%.