Практическое задание №2

8383 Ишанина Людмила, вариант 2

Задание:

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)
```

```
def genData(size=500):
    data = np.random.rand(size, 2)*2 - 1
    label = np.zeros([size, 1])
```

```
    for i, p in enumerate(data):
        if p[0]*p[1] >= 0:
            label[i] = 1.
        else:
            label[i] = 0.
    div = round(size*0.8)
    train_data = data[:div, :]
    test_data = data[div:, :]
    train_label = label[:div, :]
    test_label = label[div:, :]
    return (train_data, train_label), (test_data, test_label)
```

Реализация:

Была создана модель со следующими слоями:

```
model.add(layers.Dense(16, activation='relu', input_shape = (2,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

В первой строчке добавляется слой с 16 нейронами, функцией активации Relu, и параметром `input_shape`, который определяет, что входной слой имеет 2 элемента.

Во второй строчке добавляется ещё один скрытый слой с 16 нейронами и функцией активации Relu.

В третьей строчке добавляется выходной слой с сигмоидной функцией активации Sigmoid, которая имеет область значений [0,1].

Пояснение:

`input_shape = (2,)` – запятая необходима, так как только одно измерение.

Далее была выполнена настройка обучения модели:

```
model.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['accuracy'])
```

В качестве оптимизатора используется RMSProp, функция потерь бинарная кросс-энтропия, а в качестве метрики используется точность.

Затем было выполнено обучение модели в течение 100 эпох пакетами по 10 образцов.

```
H = model.fit(train_data, train_label, epochs=100, batch_size=10,
validation_data=(test_data, test_label))
```

Графики:

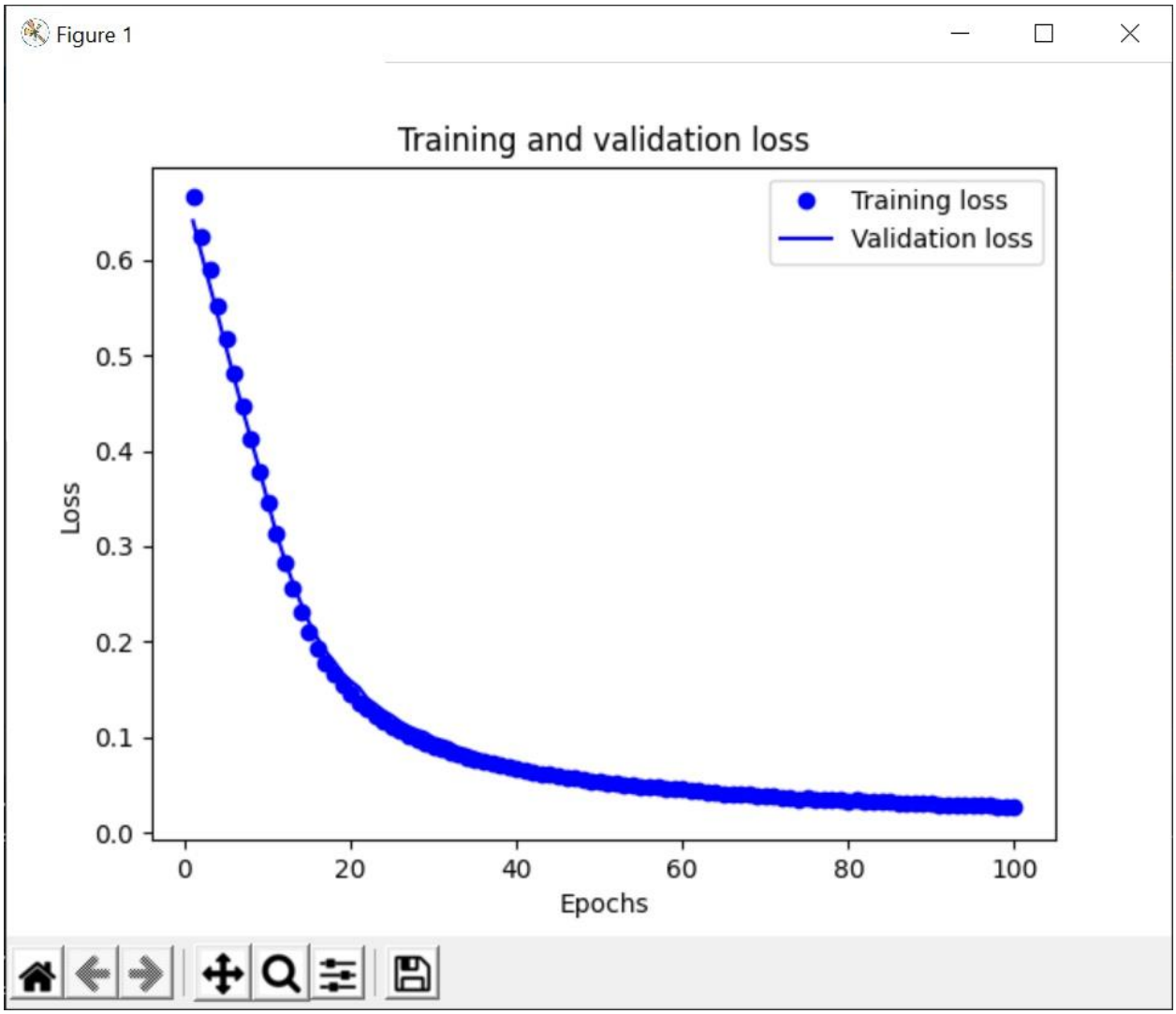

Рисунок 1 – График потерь

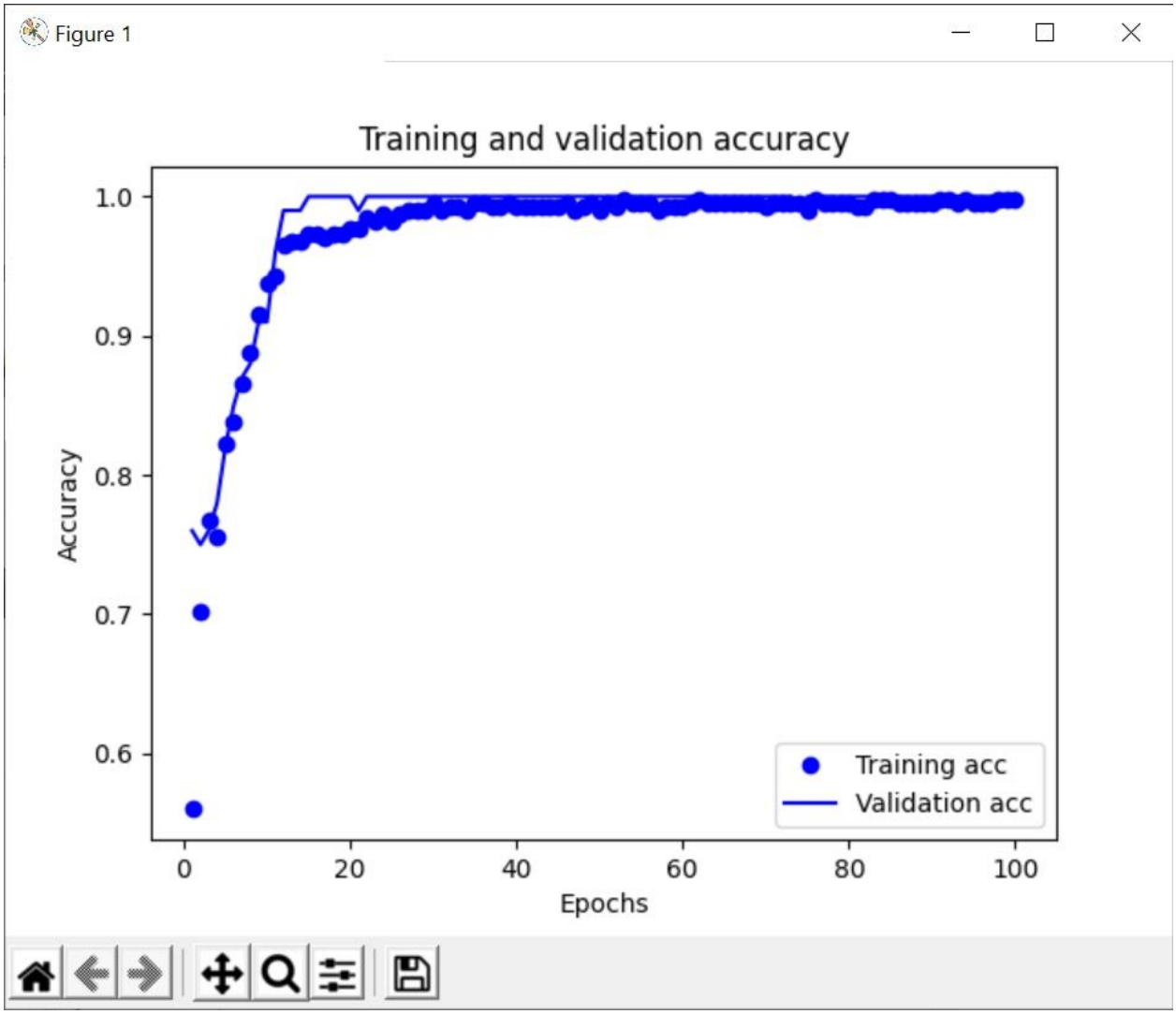

Рисунок 2 – График точности

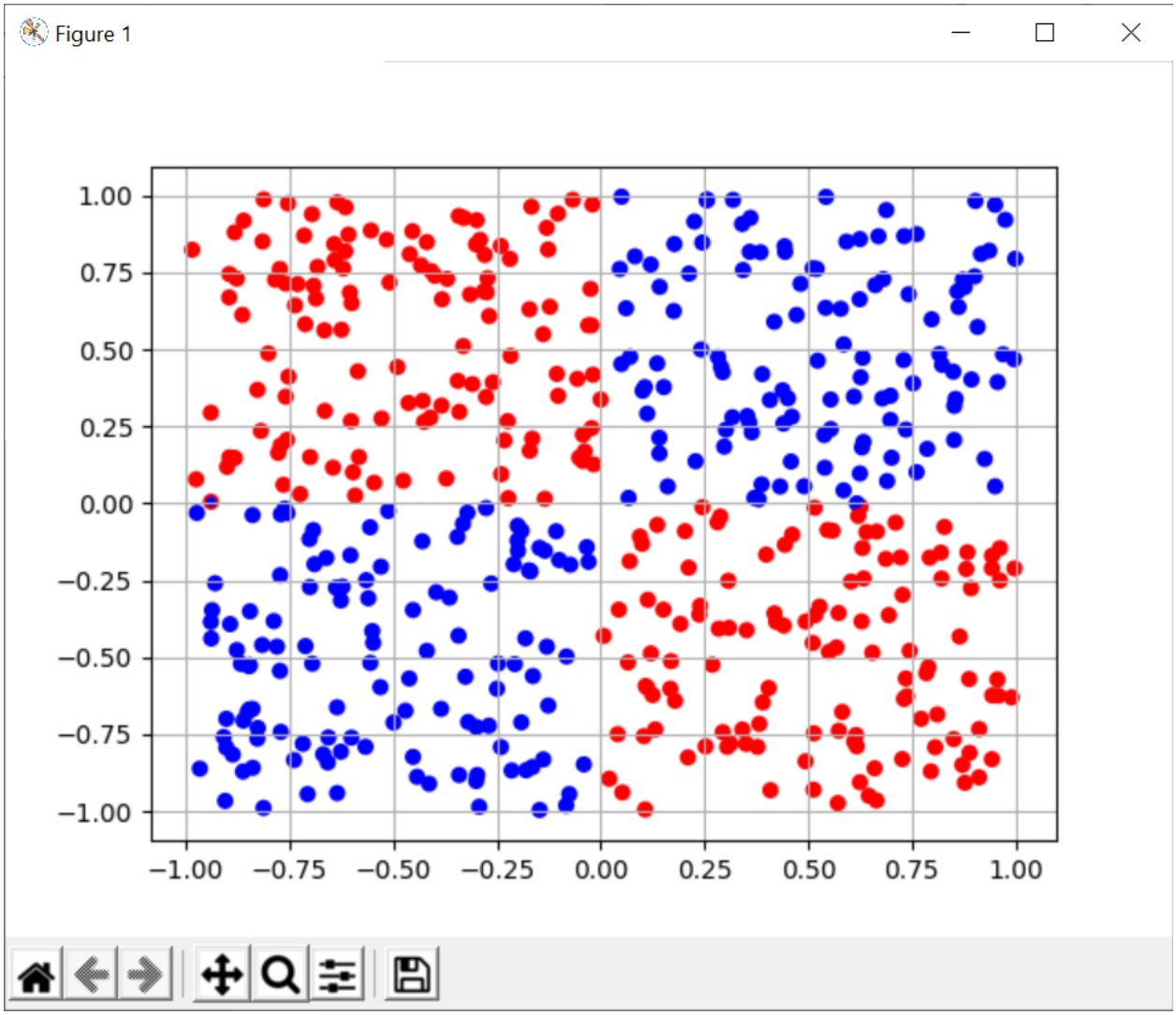

Рисунок 3 – График бинарной классификации

ИНС достигла следующих показателей на 100 эпохе:

loss: 0.0274 - acc: 0.9975 - val_loss: 0.0250 - val_acc: 1.0000