Московский авиационный институт
(национальный исследовательский университет)

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №3 по курсу «Нейроинформатика»

Студент: В. В. Бирюков
Преподаватель: И. А. Рожлейс
Группа: М8О-407Б-19
Дата:
Оценка:
Подпись:

Москва, 2022

Лабораторная работа №3

Тема: Многослойные сети. Алгоритм обратного распространения ошибки.

Цель работы: Исследование свойств многослойной нейронной сети прямого распространения и алгоритмов ее обучения, применение сети в задачах классификации и аппроксимации функции.

Основные этапы работы:

1. Использовать многослойную нейронную сеть для классификации точек в случае, когда классы не являются линейно разделимыми.
2. Использовать многослойную нейронную сеть для аппроксимации функции.

Вариант: 9

Алгебраические линии:

- Эллипс: $a = 0.2,\ b = 0.2,\ \alpha = 0,\ x_0 = -0.2,\ y_0 = 0$
- Эллипс: $a = 0.7,\ b = 0.5,\ \alpha = -\pi/3,\ x_0 = 0,\ y_0 = 0$
- Эллипс: $a = 1,\ b = 1,\ \alpha = 0,\ x_0 = 0,\ y_0 = 0$

Функция: $x = \sin(t^2 - 2t + 5)$

1 Исходный код

Лабораторная работа № 3

Вариант: 9

[1]:
```
import numpy as np
from tensorflow import keras
import matplotlib.pyplot as plt
from numpy import sin, cos, pi
from sklearn.model_selection import train_test_split
from matplotlib.colors import LinearSegmentedColormap
```

[2]:
```
%matplotlib inline
import matplotlib_inline
matplotlib_inline.backend_inline.set_matplotlib_formats('svg', 'pdf')
```

[3]:
```
def plot_history(h, *metrics):
    for metric in metrics:
        print(f"{metric}: {h.history[metric][-1]:.4f}")
    figure = plt.figure(figsize=(5.5 * len(metrics), 3.5))
    for i, metric in enumerate(metrics, 1):
        ax = figure.add_subplot(1, len(metrics), i)
        ax.xaxis.get_major_locator().set_params(integer=True)
        plt.title(metric)
        plt.plot(h.history[metric], '-')
    plt.show()
```

Классификация

[4]:
```
def ellipse(t, a, b, x0, y0, alpha):
    x = a * cos(t)
    y = b * sin(t)
    x, y = rotate(x, y, alpha)
    return np.array((x + x0, y + y0)).T

def rotate(x, y, alpha):
    xr = x * cos(alpha) - y * sin(alpha)
    yr = x * sin(alpha) + y * cos(alpha)
    return xr, yr
```

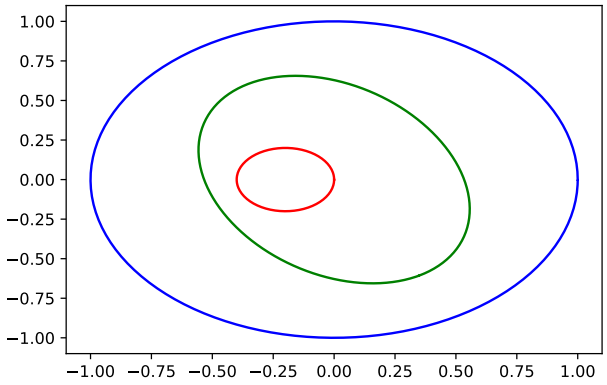
[5]:
```
def plot_three_classes(data, labels, colors):
    plt.scatter(data1[:, 0], data1[:, 1], c=[colors[i[1]+i[2]*2] for i in labels1],␣
 ↪marker='.')
COLORS = ['red', 'green', 'blue']
```

[6]:
```
a1 = 0.2; b1 = 0.2; alpha1 = 0;       x01 = -0.2; y01 = 0
a2 = 0.7; b2 = 0.5; alpha2 = -pi / 3; x02 = 0;    y02 = 0
```

```
a3 = 1;   b3 = 1;   alpha3 = 0;       x03 = 0;    y03 = 0
```

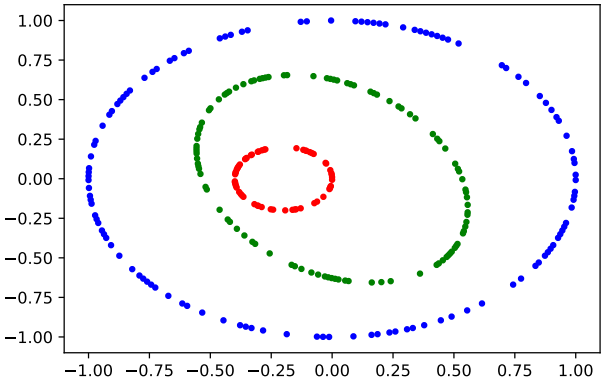
[7]:
```
t = np.arange(0, 2 * pi, 0.025)
ellipse1 = ellipse(t, a1, b1, x01, y01, alpha1)
ellipse2 = ellipse(t, a2, b2, x02, y02, alpha2)
ellipse3 = ellipse(t, a3, b3, x03, y03, alpha3)
```

[8]:
```
plt.plot(ellipse1[:, 0], ellipse1[:, 1], COLORS[0])
plt.plot(ellipse2[:, 0], ellipse2[:, 1], COLORS[1])
plt.plot(ellipse3[:, 0], ellipse3[:, 1], COLORS[2])
plt.show()
```



[9]:
```
rng = np.random.default_rng()
data1 = np.array((*rng.choice(ellipse1, 60, False, axis=0),
                  *rng.choice(ellipse2, 100, False, axis=0),
                  *rng.choice(ellipse3, 120, False, axis=0)))
labels1 = np.array((*[[1, 0, 0] for _ in range(60)],
                    *[[0, 1, 0] for _ in range(100)],
                    *[[0, 0, 1] for _ in range(120)]))
```

[10]:
```
plot_three_classes(data1, labels1, COLORS)
plt.show()
```

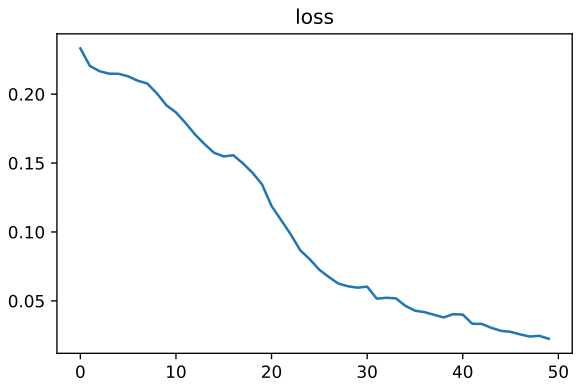


[11]:
```
train_data1, test_data1, train_labels1, test_labels1 = train_test_split(data1,␣
↪labels1, train_size=0.8)
```

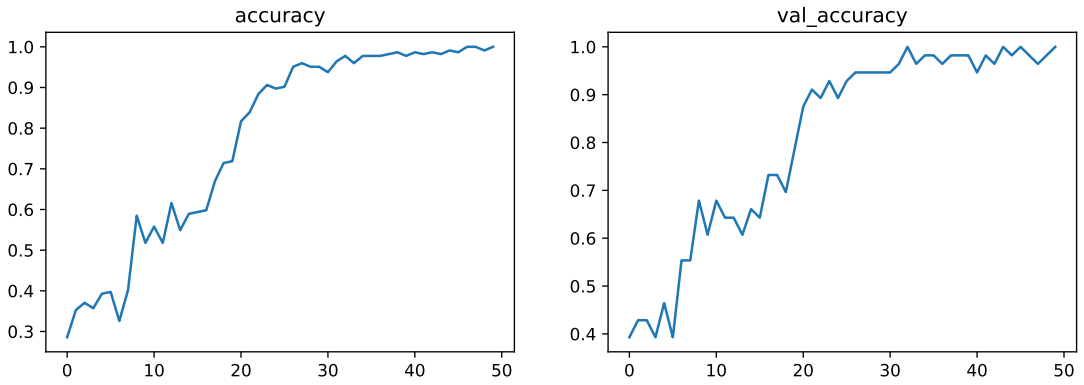
[12]:
```
model1 = keras.models.Sequential([
    keras.layers.Dense(20, input_dim=2, activation='tanh'),
    keras.layers.Dense(50, activation='tanh'),
    keras.layers.Dense(3, activation='sigmoid')
])
model1.compile(keras.optimizers.Adam(0.01), 'mse', ['accuracy'])

hist1 = model1.fit(train_data1, train_labels1, validation_data=(test_data1,␣
↪test_labels1), batch_size=20, epochs=50, verbose=0)
```

[13]:
```
plot_history(hist1, 'loss')
plot_history(hist1, 'accuracy', 'val_accuracy')
```

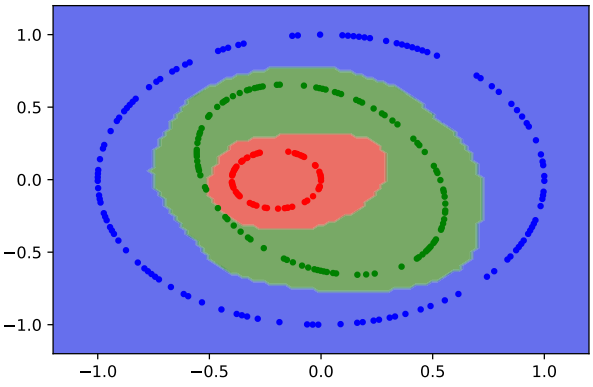
```
loss: 0.0225
```



```
accuracy: 1.0000
val_accuracy: 1.0000
```

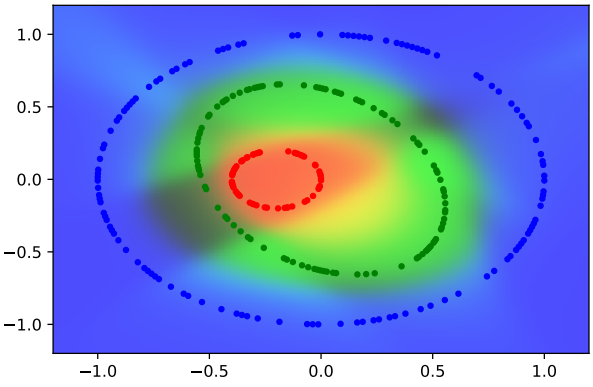


[14]:
```
n = 100
x = np.linspace(-1.2, 1.2, n)
y = np.linspace(-1.2, 1.2, n)

xv, yv = np.meshgrid(x, y)
z = model1.predict(np.c_[xv.ravel(), yv.ravel()]).argmax(axis=1).reshape(n, n)

cmap = LinearSegmentedColormap.from_list('cmap', COLORS)
plt.contourf(xv, yv, z, alpha = 0.6, cmap=cmap)
plot_three_classes(train_data1, train_labels1, COLORS)
plt.show()
```



[15]:
```
n = 100
x = np.linspace(-1.2, 1.2, n)
y = np.linspace(-1.2, 1.2, n)
```

```
xv, yv = np.meshgrid(x, y)
z = model1.predict(np.c_[xv.ravel(), yv.ravel()]).reshape(n, n, 3)

plt.imshow(z, extent=(-1.2, 1.2, -1.2, 1.2), alpha=0.7, origin='lower',␣
 ↪aspect='auto')
plot_three_classes(test_data1, test_labels1, COLORS)
plt.show()
```

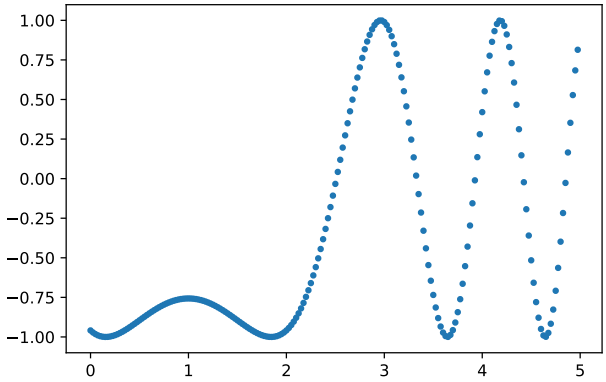


Аппроксимация

[16]:
```
def f(t):
    return sin(t ** 2 - 2 * t + 5)
```

[18]:
```
h = 0.025
train_data2 = np.arange(0, 5, h)
train_labels2 = f(train_data2)
```

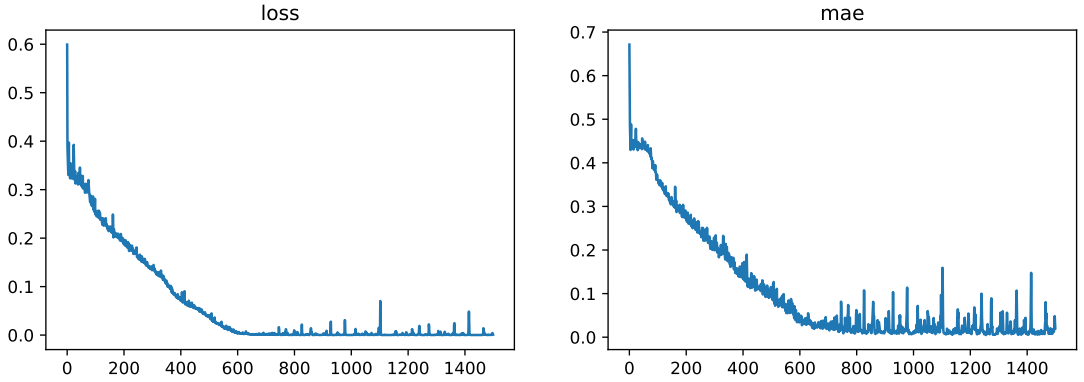
[19]:
```
plt.plot(train_data2, train_labels2, '.')
plt.show()
```



[20]:
```
model2 = keras.models.Sequential([
    keras.layers.Dense(50, input_dim=1, activation='tanh'),
    keras.layers.Dense(100, activation='tanh'),
    keras.layers.Dense(1, activation='linear')
])
model2.compile(keras.optimizers.Adam(0.001), 'mse', ['mae'])

hist2 = model2.fit(train_data2, train_labels2, batch_size=5, epochs=1500, verbose=0,␣
↪shuffle=True)
```

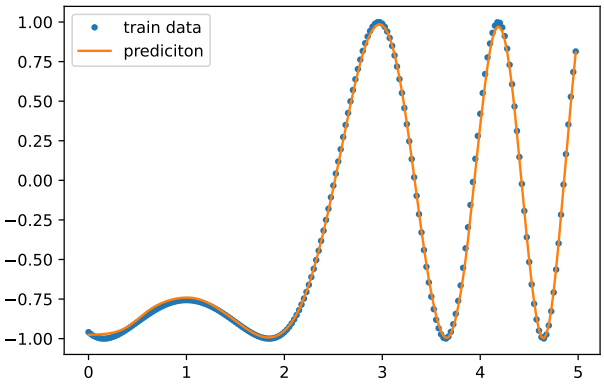
[21]:
```
plot_history(hist2, 'loss', 'mae')
```

```
loss: 0.0009
mae: 0.0203
```

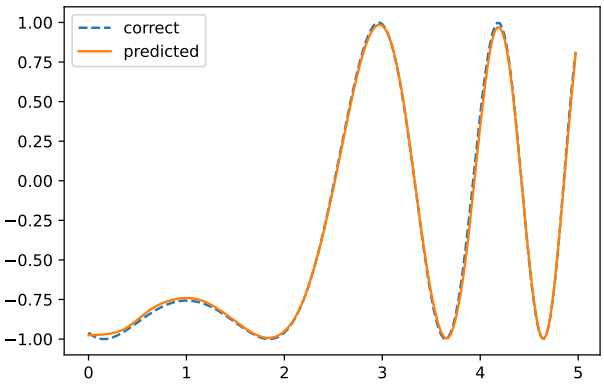


[22]:
```
plt.plot(train_data2, train_labels2, '.', label='train data')
plt.plot(train_data2, model2.predict(train_data2).flat, label='prediction')
plt.legend()
```

```
plt.show()
```



[23]:
```
test_data2 = np.arange(0, 5, h / 2)
test_labels2 = f(test_data2)

plt.plot(train_data2, train_labels2, '--', label='correct')
plt.plot(train_data2, model2.predict(train_data2).flat, label='predicted')
plt.legend()
plt.show()
```



2 Выводы

В ходе выполнения лабораторной работы я познакомился с многослойными сетями. Многослойные сети способны аппроксимировать и разделять различные нелинейные данные. Однако они требуют гораздо больше данных и времени для обучения.