Лабораторная работа № 3

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

Студент Мариничев И.А.

Группа М8О-408Б-19

Вариант 5

Определим три группы точек (эллипсы с поворотом), соответствующие трем классам

```
def ellipse(t, a, b, x0, y0):
    x = x0 + a * np.cos(t)
    y = y0 + b * np.sin(t)
    return x, y

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

t = np.linspace(0, 2 * np.pi, 200)

# Эллипс: a = 0.4, b = 0.15, α = π/6, x0 = -0.1, y0 = 0.15
x1, y1 = ellipse(t, a=0.4, b=0.15, x0=-0.1, y0=0.15)
x1, y1 = rotate(x1, y1, np.pi / 6.)

# Эллипс: a = 0.7, b = 0.5, α = -π/3, x0 = 0, y0 = 0
x2, y2 = ellipse(t, a=0.7, b=0.5, x0=0., y0=0.)
x2, y2 = rotate(x2, y2, -np.pi / 3.)

# Эллипс: a = 1, b = 1, α = 0, x0 = 0, y0 = 0
x3, y3 = ellipse(t, a=1., b=1., x0=0., y0=0.)
x3, y3 = rotate(x3, y3, 0.)

points1 = [[x, y] for x, y in zip(x1, y1)]
points2 = [[x, y] for x, y in zip(x2, y2)]
points3 = [[x, y] for x, y in zip(x3, y3)]

classes1 = [[1., 0., 0.] for _ in range(len(points1))]
classes2 = [[0., 1., 0.] for _ in range(len(points2))]
```

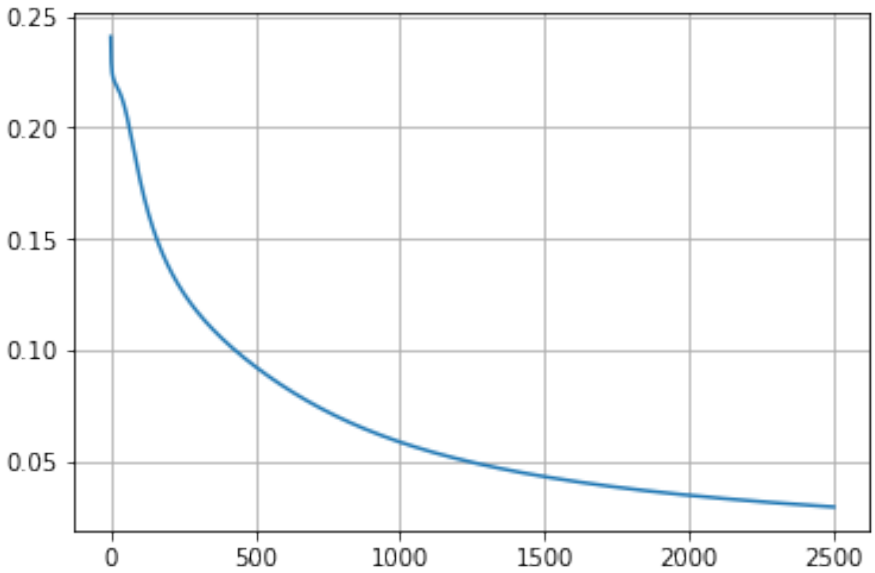
```
classes3 = [[0., 0., 1.] for _ in range(len(points3))]

X = points1 + points2 + points3
y = classes1 + classes2 + classes3
```

Будем обучать нашу сеть батчами размера batch_size, так как данных уже довольно много.

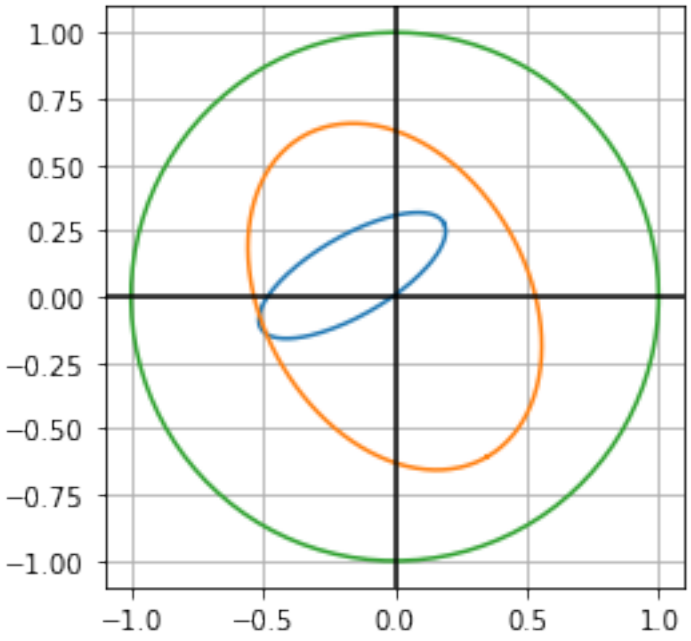
Создадим класс двухслойной сети

```
class TwoLayerNet(nn.Module):
    def __init__(self, in_features: int, hidden_layer: int, out_features:
int):
        super().__init__()
        self.fc1 = nn.Linear(in_features, hidden_layer)
        self.fc2 = nn.Linear(hidden_layer, out_features)

    def forward(self, x):
        x = self.fc1(x)
        x = torch.tanh(x)
        x = self.fc2(x)
        x = torch.sigmoid(x)
        return x
```

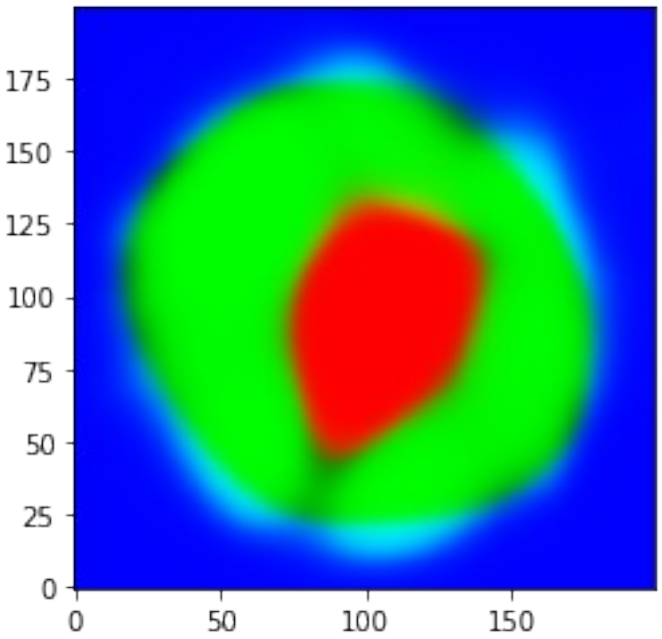
Наша сеть будет принимать на вход два признака, координаты (x, y), а на выходе будет выдавать три значения в диапазоне [0, 1], чтобы их можно было интерпретировать как цветовую компоненту RGB. В качестве функции потерь берем nn.MSELoss(). Обучим модель. Посмотрим на график функции потерь, вычисляющей MSE между исходными и полученными данными.



Теперь соберем предсказания модели для каждой точки области [-1, 1] x [-1, 1], построим наши три класса



И посмотрим на цветовое представление того, как наша сеть справилась с разделением области на три класса, которые линейно неразделимы



Теперь перейдем к задаче аппроксимации функции. Создадим разреженную дискретную версию нашей исходной функции и будем использовать ее для

обучения, чтобы потом получить при увеличенни шага приближение исходной функции

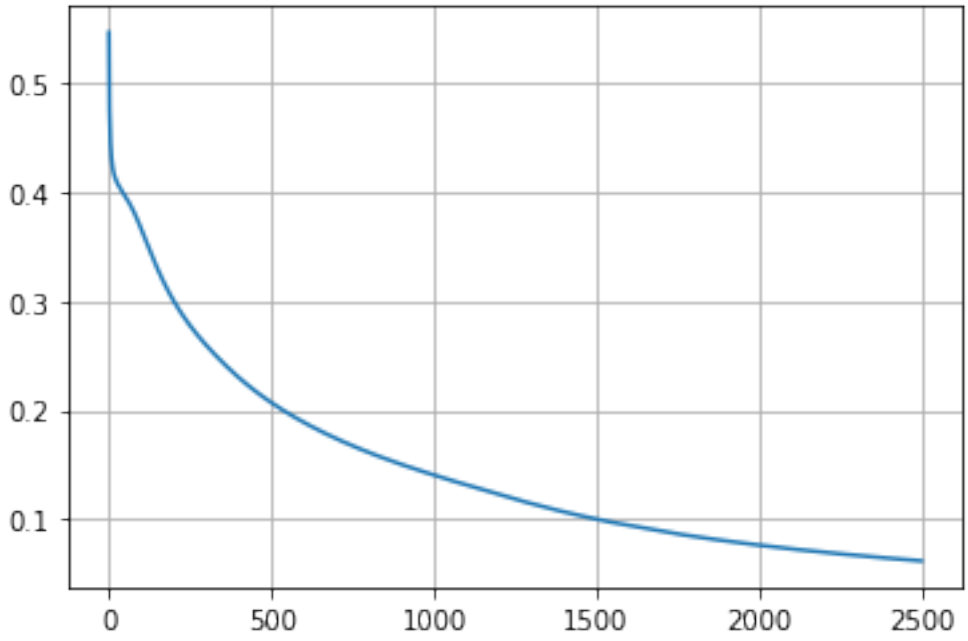
```
def function(t):
    return np.cos(-3 * t**2 + 5 * t + 10)

t1 = np.linspace(0, 2.5, 150)
f1 = function(t1)

t2 = np.linspace(0, 2.5, 2000)
f2 = function(t2)
```

Создадим класс трехслойной сети

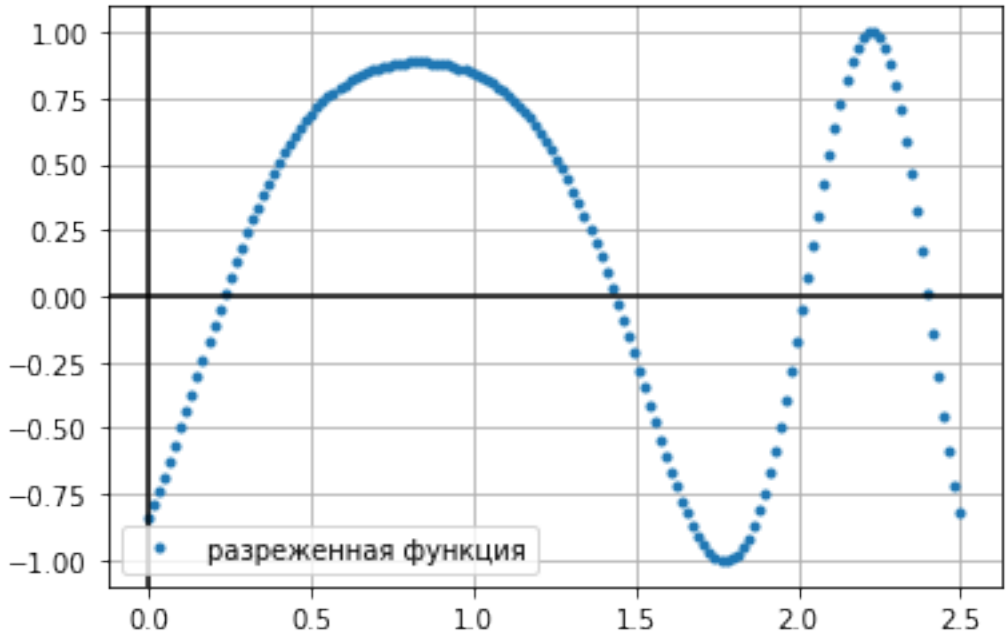
```
class ThreeLayerNet(nn.Module):
    def __init__(self, in_features: int, hidden_layer: list, out_features:
int):
        super().__init__()
        assert len(hidden_layer) == 2
        self.fc1 = nn.Linear(in_features, hidden_layer[0])
        self.fc2 = nn.Linear(hidden_layer[0], hidden_layer[1])
        self.fc3 = nn.Linear(hidden_layer[1], out_features)

    def forward(self, x):
        x = self.fc1(x)
        x = torch.tanh(x)
        x = self.fc2(x)
        x = torch.tanh(x)
        x = self.fc3(x)
        return x
```

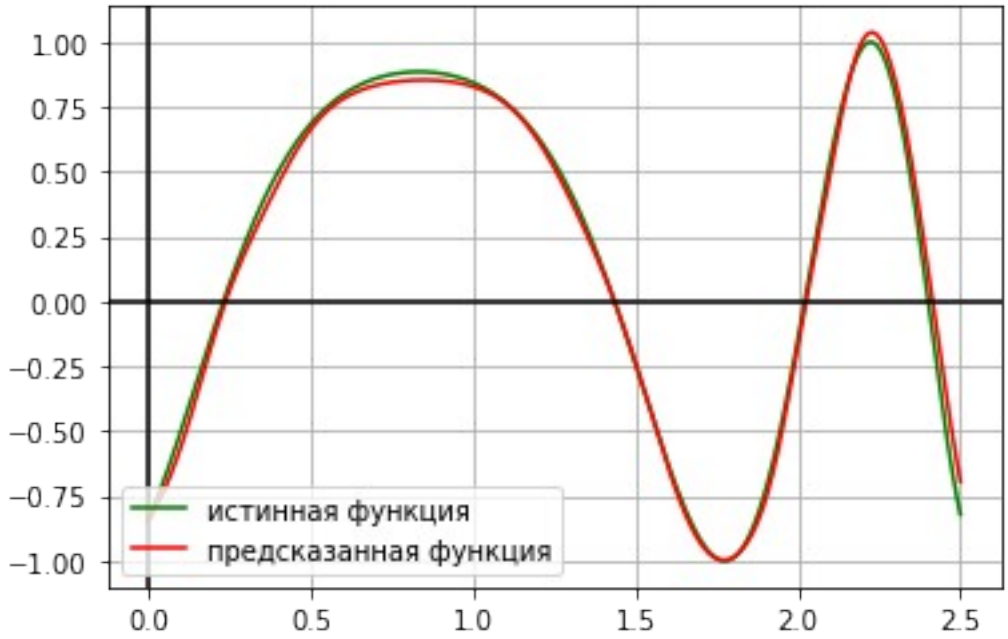
Наша сеть будет принимать на вход один признак, координату x, а на выходе будет выдавать значение функции в этой точке. В качестве функции потерь берем nn.MSELoss().Будем обучать нашу сеть батчами размера batch_size, так как данных уже довольно много. Обучим модель.

Посмотрим на график функции потерь, вычисляющей MSE между исходными и полученными данными



Соберем предсказания модели и визуализируем полученные результаты, сравним приближение, разреженный вариант и истинную функцию





Выводы: в ходе данной работы были построены две многослойные сети, которые были использованы для решения двух типов задач:

- классификация (линейно неразделимые данные)
- аппроксимация

После обучения были получены довольно неплохие результаты, правда обучать пришлось уже 2500 эпох.