Домашнее задание по теме “Введение в нейросети”

Выполнила: Китова Екатерина Денисовна, 324 группа

Постановка задачи:
*№4**: Придумайте собственную несложную функцию, которую нельзя реализовать с помощью одного нейрона, но можно с помощью нескольких (2-4). Доказательство, программа, эксперименты.*

Решение:
Я выбрала булеву функцию от трёх входов x1, x2, x3 из {0,1}:
f(x1, x2, x3) = 1, если (x1 XOR x2) И x3 = 1;
иначе f = 0.

То есть функция даёт 1 только в двух случаях: когда первые два бита разные и третий бит равен 1.

Полная таблица истинности для функции:

№	X1	X2	X3	X1 XOR X2	F
1	0	0	0	0	0
2	0	0	1	0	0
3	0	1	0	1	0
4	0	1	1	1	1
5	1	0	0	1	0
6	1	0	1	1	1
7	1	1	0	0	0
8	1	1	1	0	0

То есть целевой класс = 1 только для векторов: (0, 1, 1), (1, 0, 1)

Почему же эту задачу нельзя решить с помощью одного нейрона?
Один нейрон с линейной функцией принятия решения умеет разделять пространство входов одной гиперплоскостью. Классический пример того, что так нельзя сделать, - функция XOR: точки (0,1) и (1,0) надо отнести к классу 1, а (0,0) и (1,1) - к классу 0, но одной прямой это не разделяется.
В моей функции внутри прямо используется XOR на первых двух входах, а затем результат ещё фильтруется третьим входом. Если уже подзадача «x1 XOR x2» не линейно разделима, то и вся функция целиком не может быть реализована одним линейным нейроном.

Значит, нужна сеть из двух слоёв: первый слой учится выделять нужные паттерны во входных векторах, второй - собирать их в окончательный ответ.

Код решения задачи:

```
import numpy as np

# сигмоида и её производная
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def sigmoid_pr(y):
    return y * (1 - y)

# все 8 входов из таблицы истинности
training_inputs = np.array([
    [0,0,0],
    [0,0,1],
    [0,1,0],
    [0,1,1],
    [1,0,0],
    [1,0,1],
    [1,1,0],
    [1,1,1],
])

# наша функция (x1 XOR x2) AND x3
def target(x):
    x1, x2, x3 = x
    xor = (x1 != x2)
    return 1 if (xor and x3 == 1) else 0

training_outputs = np.array([[target(x)] for x in training_inputs])

# размеры
input_size = 3
hidden_size = 2 # возьмем два нейрона
output_size = 1 # на выход класс 0 или 1
np.random.seed(42)
m1 = 2 * np.random.random((input_size, hidden_size)) - 1
m2 = 2 * np.random.random((hidden_size, output_size)) - 1

for iter in range(50000):
    L0 = training_inputs
    L1 = sigmoid(np.dot(L0, m1))
    L2 = sigmoid(np.dot(L1, m2)) # предсказания
    L2_er = training_outputs - L2
    if iter % 10000 == 0:
        print("iter:", iter, "error:", np.mean(L2_er**2))
    L2_1= L2_er * sigmoid_pr(L2)
    L1_er = L2_1.dot(m2.T)
    L1_1= L1_er * sigmoid_pr(L1)
    m2 += L1.T.dot(L2_1)
    m1 += L0.T.dot(L1_1)

print("Результат на обучающем наборе:")
L1 = sigmoid(np.dot(training_inputs, m1))
L2 = sigmoid(np.dot(L1, m2))
for x, y_true, y_pred in zip(training_inputs, training_outputs, L2):
    print(f"{x}: должно быть {int(y_true[0])}, сеть предсказала
{y_pred.item():.6f}")
tests = np.array([
    [0,1,1],
    [1,0,1],
    [1,1,1],
])
```

```
print("Проверка:")
for t in tests:
    l1 = sigmoid(np.dot(t, m1))
    l2 = sigmoid(np.dot(l1, m2))
    print(f"{t}: {l2[0]:.6f}")
```

Результаты работы программы:

iter: 0 error: 0.250360677342107
iter: 10000 error: 0.0005012041385888391
iter: 20000 error: 0.00020060111310980306
iter: 30000 error: 0.0001247408074865624
iter: 40000 error: 9.035753773044043e-05
Результат на обучающем наборе:
[0 0 0]: должно быть 0, сеть предсказала 0.003894
[0 0 1]: должно быть 0, сеть предсказала 0.014203
[0 1 0]: должно быть 0, сеть предсказала 0.003306
[0 1 1]: должно быть 1, сеть предсказала 0.989985
[1 0 0]: должно быть 0, сеть предсказала 0.003306
[1 0 1]: должно быть 1, сеть предсказала 0.989985
[1 1 0]: должно быть 0, сеть предсказала 0.000069
[1 1 1]: должно быть 0, сеть предсказала 0.011264
Проверка:
[0 1 1]: 0.989985
[1 0 1]: 0.989985
[1 1 1]: 0.011264

Как можно видеть из вывода программы и трех тестов, наша нейросеть успешно выучила нашу функцию и выдает качественные результаты.