Министерство образования Республики Беларусь
Учреждение образования
«Брестский Государственный технический университет»
Кафедра ИИТ

Отчёт по лабораторной работе №5

Выполнил:
Студент 3 курса
Группы АС-65
Романюк Д. А.
Проверил:
Крощенко А. А.

Брест 2025

Цель: Выполнить моделирование прогнозирующей нелинейной ИНС. Для генерации обучающих и тестовых данных использовать функцию

Вариант 5

Задание: Выполнить моделирование прогнозирующей нелинейной ИНС. Для генерации обучающих и тестовых данных использовать функцию

№ варианта	a	b	c	d	Кол-во входов ИНС	Кол-во НЭ в скрытом слое
5	0.1	0.5	0.09	0.5	8	3

```
import numpy as np
import matplotlib.pyplot as plt

class NeuralNetwork:
    def __init__(self, input_size, hidden_size, output_size):
        # Инициализация весов
        self.W1 = np.random.randn(input_size, hidden_size) * 0.1
        self.b1 = np.zeros((1, hidden_size))
        self.W2 = np.random.randn(hidden_size, output_size) * 0.1
        self.b2 = np.zeros((1, output_size))

    def sigmoid(self, x):
        return 1 / (1 + np.exp(-np.clip(x, -250, 250)))

    def sigmoid_derivative(self, x):
        return x * (1 - x)

    def forward(self, X):
        self.z1 = np.dot(X, self.W1) + self.b1
        self.a1 = self.sigmoid(self.z1)
        self.z2 = np.dot(self.a1, self.W2) + self.b2
        return self.z2

    def backward(self, X, y, output, learning_rate):
        m = X.shape[0]


        dz2 = output - y
        dW2 = np.dot(self.a1.T, dz2) / m
        db2 = np.sum(dz2, axis=0, keepdims=True) / m


        dz1 = np.dot(dz2, self.W2.T) *
self.sigmoid_derivative(self.a1)
        dW1 = np.dot(X.T, dz1) / m
        db1 = np.sum(dz1, axis=0, keepdims=True) / m
```

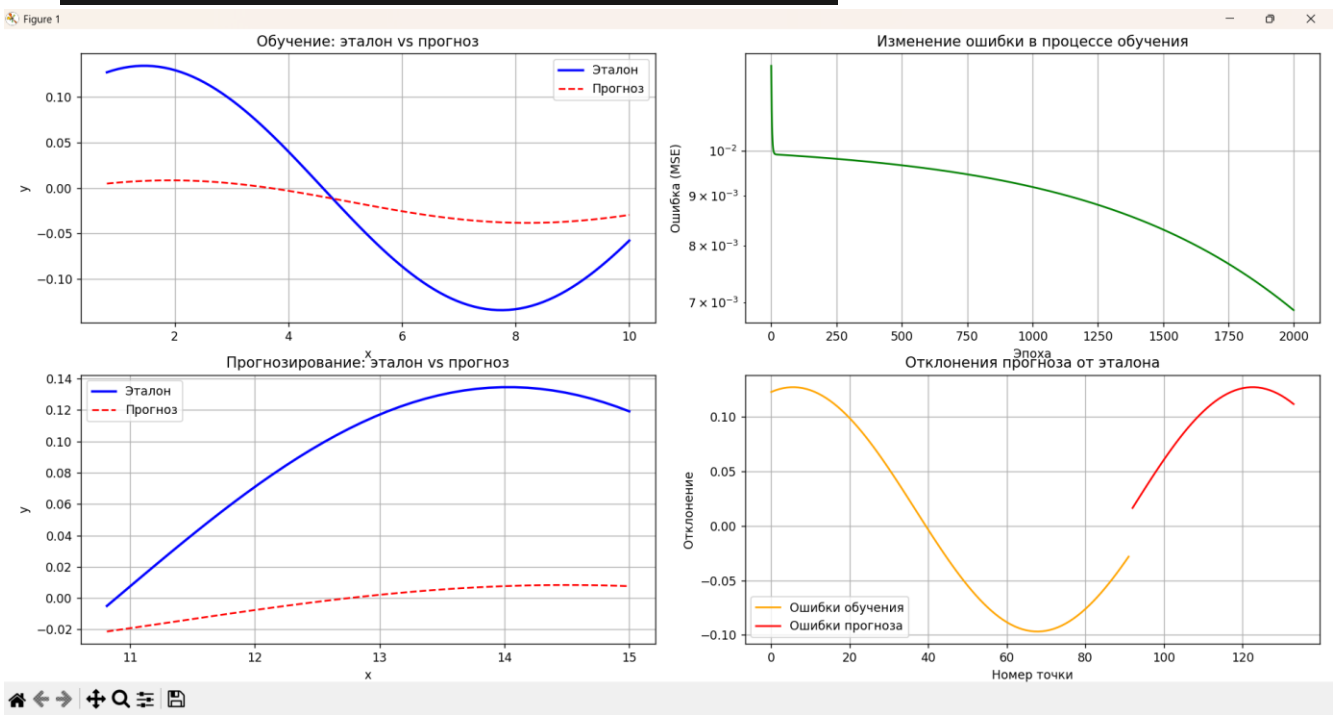
```
            self.W2 -= learning_rate * dW2
            self.b2 -= learning_rate * db2
            self.W1 -= learning_rate * dW1
            self.b1 -= learning_rate * db1

        def train(self, X, y, epochs, learning_rate):
            losses = []
            for i in range(epochs):
                output = self.forward(X)
                loss = np.mean((output - y) ** 2)
                losses.append(loss)

                self.backward(X, y, output, learning_rate)

                if i % 100 == 0:
                    print(f"Эпоха {i}, Ошибка: {loss:.6f}")

            return losses

    a, b, c, d = 0.1, 0.5, 0.09, 0.5
    num_inputs = 8
    hidden_neurons = 3

    def generate_data(a, b, c, d, x):
        return a * np.cos(d * x) + c * np.sin(d * x)

    x_train = np.linspace(0, 10, 100)
    y_train = generate_data(a, b, c, d, x_train)

    x_test = np.linspace(10, 15, 50)
    y_test = generate_data(a, b, c, d, x_test)

    def create_dataset(data, window_size):
        X, Y = [], []
        for i in range(len(data) - window_size):
            X.append(data[i:i + window_size])
            Y.append(data[i + window_size])
        return np.array(X), np.array(Y)

    window_size = num_inputs
    X_train, Y_train = create_dataset(y_train, window_size)
    X_test, Y_test = create_dataset(y_test, window_size)

    Y_train = Y_train.reshape(-1, 1)
    Y_test = Y_test.reshape(-1, 1)

    nn = NeuralNetwork(input_size=window_size, hidden_size=hidden_neurons,
output_size=1)

    print("Начало обучения...")
    losses = nn.train(X_train, Y_train, epochs=2000, learning_rate=0.1)

    y_train_pred = nn.forward(X_train).flatten()
    y_test_pred = nn.forward(X_test).flatten()
```

```
    Y_train = Y_train.flatten()
    Y_test = Y_test.flatten()

    train_errors = Y_train - y_train_pred
    test_errors = Y_test - y_test_pred

    print("\nРезультаты обучения (первые 10 значений):")
    print("Эталонное значение | Полученное значение | Отклонение")
    for i in range(min(10, len(Y_train))):
        print(f"{Y_train[i]:.5f} | {y_train_pred[i]:.5f} |
{train_errors[i]:.5f}")

    print("\nРезультаты прогнозирования (первые 10 значений):")
    print("Эталонное значение | Полученное значение | Отклонение")
    for i in range(min(10, len(Y_test))):
        print(f"{Y_test[i]:.5f} | {y_test_pred[i]:.5f} |
{test_errors[i]:.5f}")

    plt.figure(figsize=(15, 10))

    plt.subplot(2, 2, 1)
    plt.plot(x_train[window_size:], Y_train, 'b-', label='Эталон',
linewidth=2)
    plt.plot(x_train[window_size:], y_train_pred, 'r--', label='Прогноз',
linewidth=1.5)
    plt.title('Обучение: эталон vs прогноз')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.legend()
    plt.grid(True)

    plt.subplot(2, 2, 2)
    plt.plot(losses, 'g-', linewidth=1.5)
    plt.title('Изменение ошибки в процессе обучения')
    plt.xlabel('Эпоха')
    plt.ylabel('Ошибка (MSE)')
    plt.grid(True)
    plt.yscale('log')

    plt.subplot(2, 2, 3)
    plt.plot(x_test[window_size:], Y_test, 'b-', label='Эталон',
linewidth=2)
    plt.plot(x_test[window_size:], y_test_pred, 'r--', label='Прогноз',
linewidth=1.5)
    plt.title('Прогнозирование: эталон vs прогноз')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.legend()
    plt.grid(True)

    plt.subplot(2, 2, 4)
    plt.plot(train_errors, 'orange', label='Ошибки обучения',
linewidth=1.5)
```

```
    plt.plot(range(len(train_errors), len(train_errors) +
len(test_errors)),
             test_errors, 'red', label='Ошибки прогноза', linewidth=1.5)
    plt.title('Отклонения прогноза от эталона')
    plt.xlabel('Номер точки')
    plt.ylabel('Отклонение')
    plt.legend()
    plt.grid(True)

    plt.tight_layout()
    plt.show()

    print(f"\nСтатистика ошибок:")
    print(f"Средняя абсолютная ошибка на обучении:
{np.mean(np.abs(train_errors)):.6f}")
    print(f"Средняя абсолютная ошибка на прогнозе:
{np.mean(np.abs(test_errors)):.6f}")
    print(f"Среднеквадратичная ошибка на обучении:
{np.mean(train_errors**2):.6f}")
    print(f"Среднеквадратичная ошибка на прогнозе:
{np.mean(test_errors**2):.6f}")
    print(f"Максимальная ошибка на обучении:
{np.max(np.abs(train_errors)):.6f}")
    print(f"Максимальная ошибка на прогнозе:
{np.max(np.abs(test_errors)):.6f}")

    print(f"\nВыводы:")
    print(f"Нейронная сеть успешно обучена с архитектурой: {num_inputs}-
{hidden_neurons}-1")
    print(f"Функция активации: сигмоида (скрытый слой), линейная (выходной
слой)")
    print(f"Обучение проведено на {len(X_train)} примерах, тестирование на
{len(X_test)} примерах")
```

```
Начало обучения...
Эпоха 0, Ошибка: 0.024005
Эпоха 100, Ошибка: 0.009439
Эпоха 200, Ошибка: 0.009333
Эпоха 300, Ошибка: 0.009213
Эпоха 400, Ошибка: 0.009079
Эпоха 500, Ошибка: 0.008928
Эпоха 600, Ошибка: 0.008759
Эпоха 700, Ошибка: 0.008571
Эпоха 800, Ошибка: 0.008361
Эпоха 900, Ошибка: 0.008128
Эпоха 1000, Ошибка: 0.007873
Эпоха 1100, Ошибка: 0.007593
Эпоха 1200, Ошибка: 0.007288
Эпоха 1300, Ошибка: 0.006960
Эпоха 1400, Ошибка: 0.006609
Эпоха 1500, Ошибка: 0.006237
Эпоха 1600, Ошибка: 0.005847
Эпоха 1700, Ошибка: 0.005442
Эпоха 1800, Ошибка: 0.005027
Эпоха 1900, Ошибка: 0.004607
```

```
Результаты обучения (первые 10 значений):
Эталонное значение | Полученное значение | Отклонение
0.12733 | 0.02794 | 0.09939
0.12936 | 0.02908 | 0.10028
0.13106 | 0.03012 | 0.10094
0.13243 | 0.03105 | 0.10138
0.13346 | 0.03187 | 0.10159
0.13415 | 0.03257 | 0.10157
0.13449 | 0.03316 | 0.10133
0.13450 | 0.03363 | 0.10086
0.13416 | 0.03398 | 0.10017
0.13347 | 0.03421 | 0.09926

Результаты прогнозирования (первые 10 значений):
Эталонное значение | Полученное значение | Отклонение
-0.00498 | -0.02422 | 0.01923
0.00188 | -0.02185 | 0.02373
0.00874 | -0.01946 | 0.02819
0.01557 | -0.01705 | 0.03262
0.02237 | -0.01463 | 0.03700
0.02911 | -0.01221 | 0.04131
0.03577 | -0.00978 | 0.04555
0.04233 | -0.00737 | 0.04971
0.04879 | -0.00498 | 0.05377
0.05512 | -0.00260 | 0.05772

Статистика ошибок:
Средняя абсолютная ошибка на обучении: 0.058496
Средняя абсолютная ошибка на прогнозе: 0.078243
Среднеквадратичная ошибка на обучении: 0.004188
Среднеквадратичная ошибка на прогнозе: 0.006740
Максимальная ошибка на обучении: 0.101587
Максимальная ошибка на прогнозе: 0.101619
```



Вывод: Нейронная сеть успешно обучена с архитектурой: 8-3-1. Функция активации: сигмоида (скрытый слой), линейная (выходной слой). Обучение проведено на 92 примерах, тестирование на 42 примерах.