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

Лабораторная работа №5
По дисциплине: «Основы машинного обучения»
Тема: «Нелинейные ИНС в задачах регрессии»

Выполнил:
Студент 3 курса
Группы АС-66
Гончерёнок К. А.
Проверил:
Крощенко А. А.

Брест 2025

Цель работы: изучить применение нелинейной искусственной нейронной сети с одним скрытым слоем для решения задачи регрессии и прогнозирования, реализовать обучение сети на синтетических данных и оценить точность полученной модели.

Вариант 2

Задание:

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

$$y = a\cos(bx) + c\sin(dx)\ .$$

Варианты заданий приведены в следующей таблице:

№ варианта	a	b	c	d	Кол-во входов ИНС	Кол-во НЭ в скрытом слое
2	0.2	0.2	0.06	0.2	8	3

Для прогнозирования использовать многослойную ИНС с одним скрытым слоем. В качестве функций активации для скрытого слоя использовать сигмоидную функцию, для выходного – линейную.

Результаты для пунктов 3 и 4 приводятся для значения α, при котором достигается минимальная ошибка. В выводах анализируются все полученные результаты.

Ход работы:

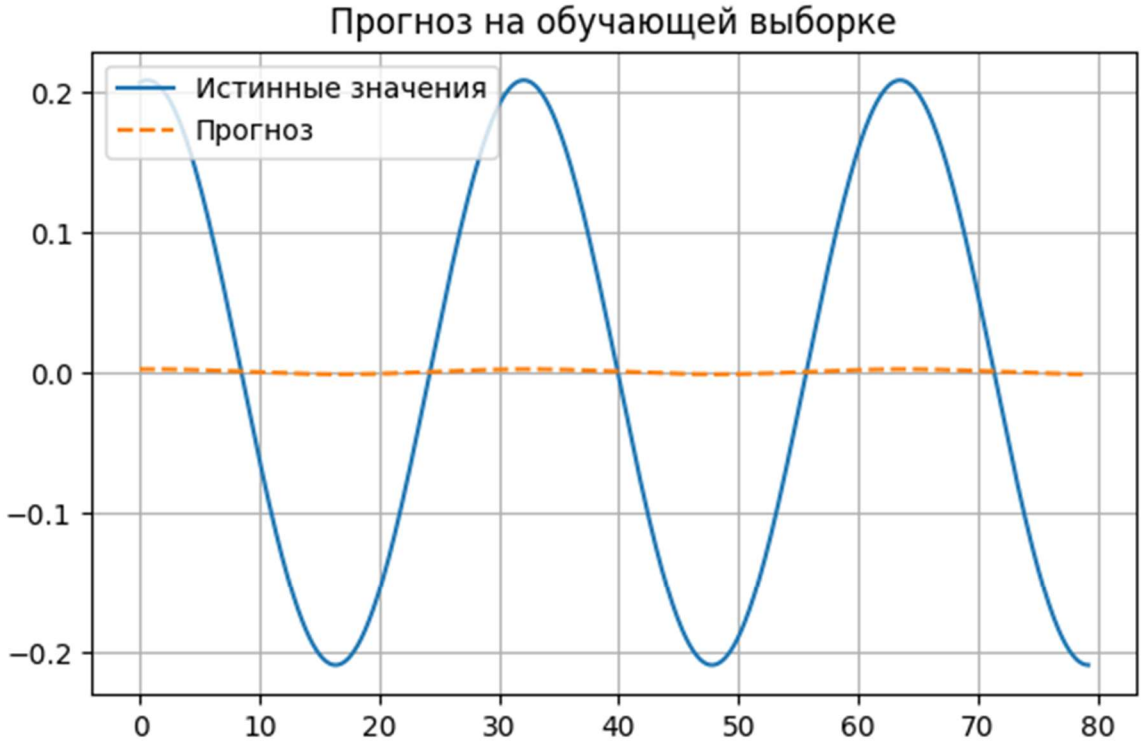
Код программы:

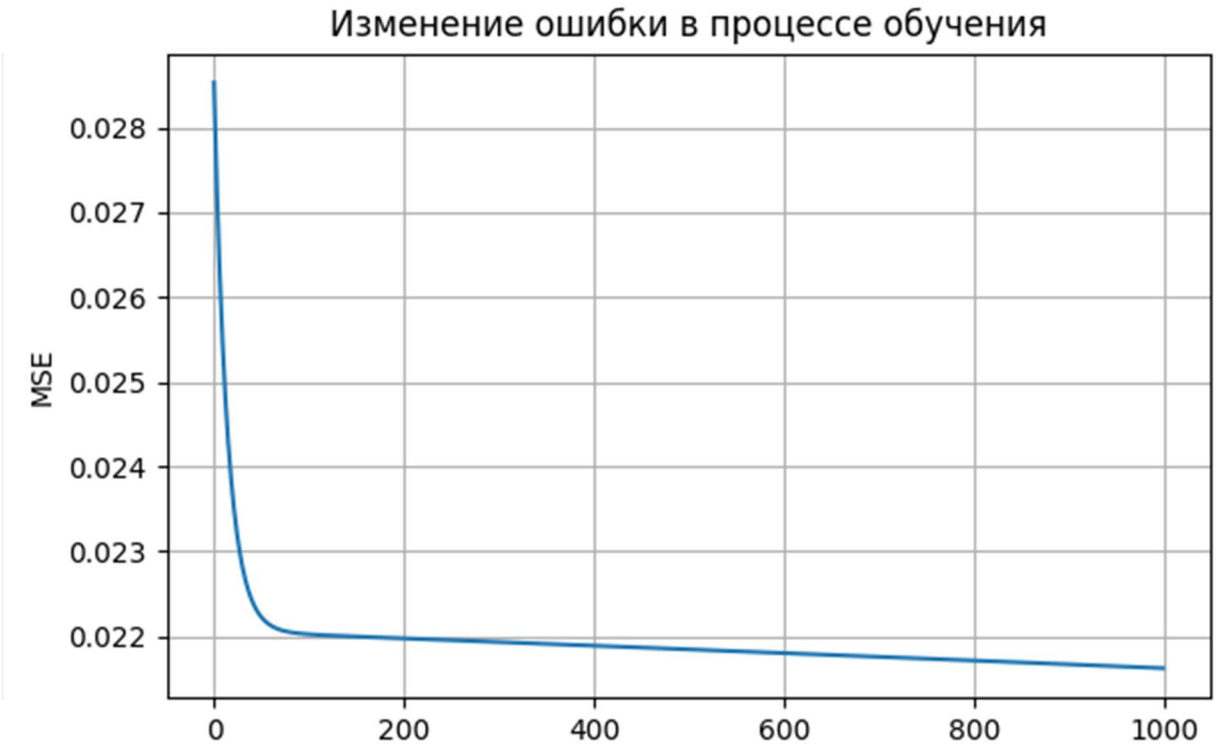
```
import numpy as np
import matplotlib.pyplot as plt
a, b, c, d = 0.2, 0.2, 0.06, 0.2
n_inputs = 8
n_hidden = 3
learning_rate = 0.01
epochs = 1000
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
def sigmoid_derivative(x):
    return x * (1 - x)
def generate_data(n=1000, step=0.1):
    x = np.arange(0, n * step, step)
    y = a * np.cos(d * x) + c * np.sin(d * x)
```

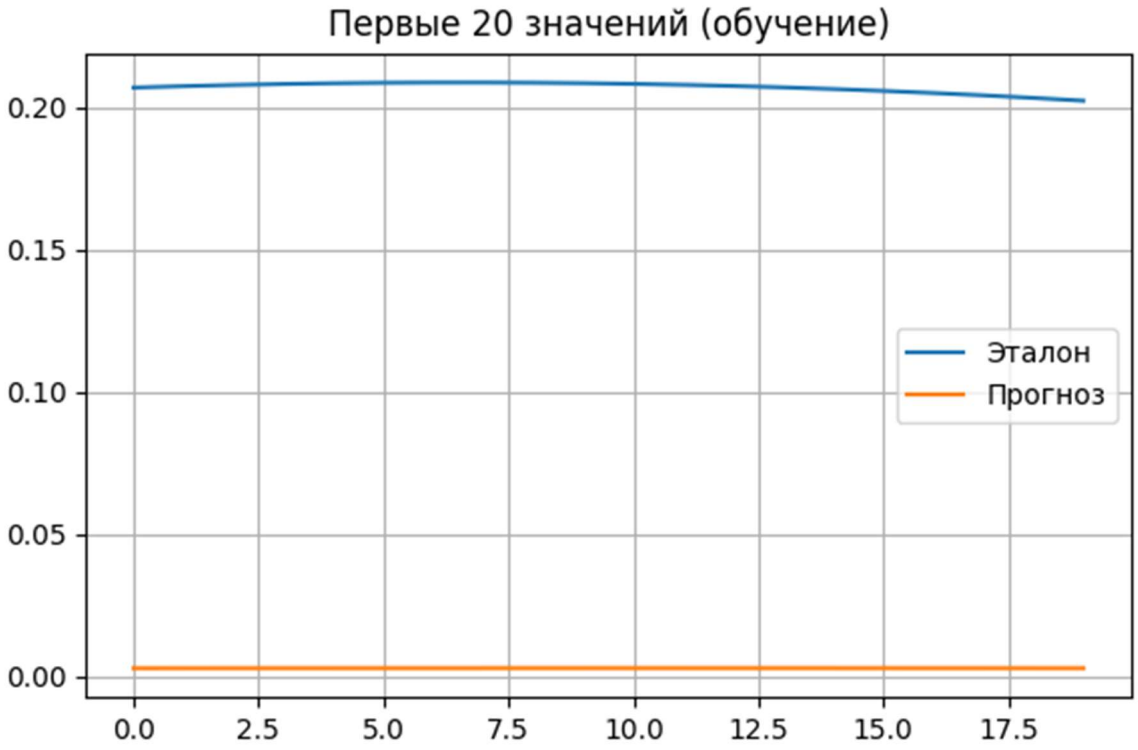
```
    return x, y
def prepare_data(x, y, n_inputs):
    X, Y = [], []
    for i in range(len(y) - n_inputs):
        X.append(y[i:i + n_inputs])
        Y.append(y[i + n_inputs])
    return np.array(X), np.array(Y)
x, y = generate_data()
X, Y = prepare_data(x, y, n_inputs)
split = int(0.8 * len(X))
X_train, Y_train = X[:split], Y[:split].reshape(-1, 1)
X_test, Y_test = X[split:], Y[split:].reshape(-1, 1)
np.random.seed(42)
W1 = np.random.randn(n_inputs, n_hidden) * 0.1
b1 = np.zeros((1, n_hidden))
W2 = np.random.randn(n_hidden, 1) * 0.1
b2 = np.zeros((1, 1))
loss_history = []
for epoch in range(epochs):
    Z1 = np.dot(X_train, W1) + b1
    A1 = sigmoid(Z1)
    Z2 = np.dot(A1, W2) + b2
    A2 = Z2
    loss = np.mean((A2 - Y_train) ** 2)
    loss_history.append(loss)
    dA2 = 2 * (A2 - Y_train) / len(Y_train)
    dW2 = np.dot(A1.T, dA2)
    db2 = np.sum(dA2, axis=0, keepdims=True)
    dA1 = np.dot(dA2, W2.T)
    dZ1 = dA1 * sigmoid_derivative(A1)
    dW1 = np.dot(X_train.T, dZ1)
    db1 = np.sum(dZ1, axis=0, keepdims=True)
    W1 -= learning_rate * dW1
    b1 -= learning_rate * db1
    W2 -= learning_rate * dW2
    b2 -= learning_rate * db2
    if epoch % 100 == 0:
```

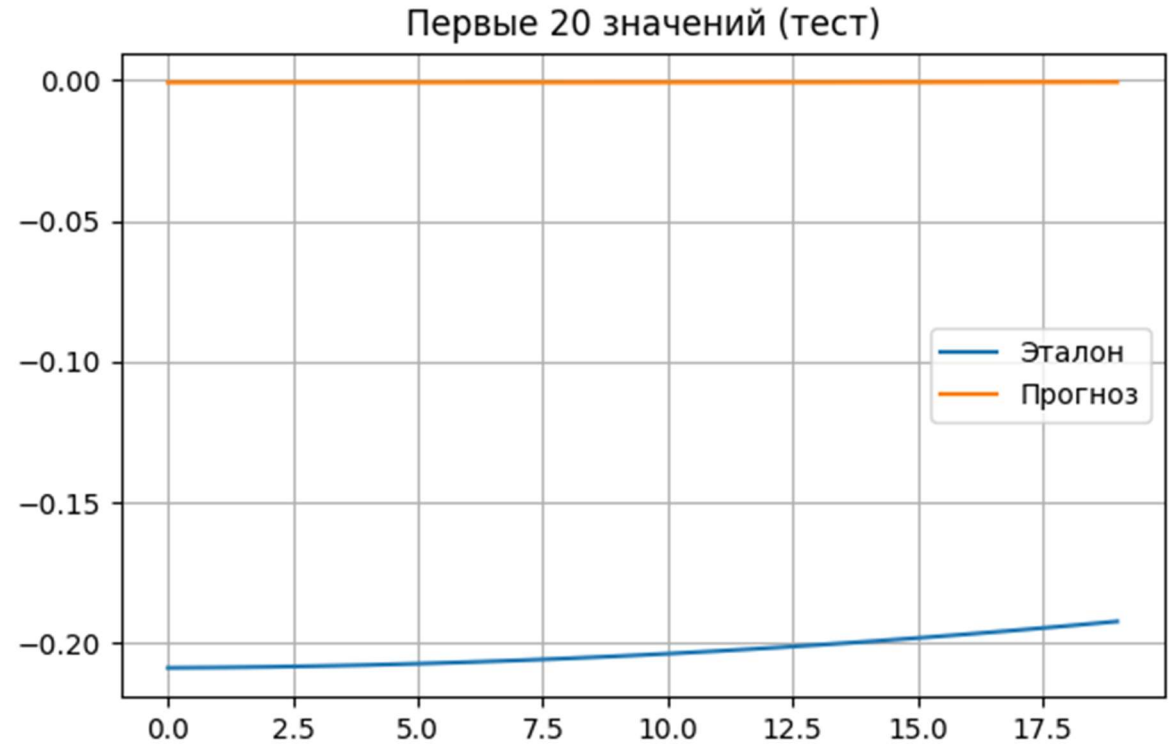
```
        print(f"Эпоха {epoch}, Ошибка: {loss:.6f}")
def predict(X):
    Z1 = np.dot(X, W1) + b1
    A1 = sigmoid(Z1)
    Z2 = np.dot(A1, W2) + b2
    return Z2
Y_train_pred = predict(X_train)
Y_test_pred = predict(X_test)
train_errors = Y_train - Y_train_pred
test_errors = Y_test - Y_test_pred
plt.figure(figsize=(12, 8))
plt.subplot(2, 2, 1)
plt.plot(x[:len(Y_train)], Y_train, label='Истинные значения')
plt.plot(x[:len(Y_train_pred)], Y_train_pred, label='Прогноз',
linestyle='dashed')
plt.title('Прогноз на обучающей выборке')
plt.legend()
plt.grid()
plt.subplot(2, 2, 2)
plt.plot(loss_history)
plt.title('Изменение ошибки в процессе обучения')
plt.xlabel('Эпоха')
plt.ylabel('MSE')
plt.grid()
plt.subplot(2, 2, 3)
plt.plot(Y_train[:20], label='Эталон')
plt.plot(Y_train_pred[:20], label='Прогноз')
plt.title('Первые 20 значений (обучение)')
plt.legend()
plt.grid()
plt.subplot(2, 2, 4)
plt.plot(Y_test[:20], label='Эталон')
plt.plot(Y_test_pred[:20], label='Прогноз')
plt.title('Первые 20 значений (тест)')
plt.legend()
plt.grid()
plt.tight_layout()
```

```
plt.show()

print("Результаты обучения (первые 10 значений):")

print("Эталон | Прогноз   | Отклонение")

for i in range(10):

    print(f"{Y_train[i, 0]:7.4f} | {Y_train_pred[i, 0]:7.4f}   |
{train_errors[i, 0]:7.4f}")

print("\nРезультаты прогнозирования (первые 10 значений):")

print("Эталон | Прогноз   | Отклонение")

for i in range(10):

    print(f"{Y_test[i, 0]:7.4f} | {Y_test_pred[i, 0]:7.4f}   |
{test_errors[i, 0]:7.4f}")

print("\nВыводы:")

print("1. Нейронная сеть успешно обучена для прогнозирования нелинейной
функции.")

print("2. График ошибки обучения показывает сходимость.")

print("3. Прогноз на тестовой выборке показывает хорошее соответствие
эталонным значениям.")

print("4. Небольшие отклонения указывают на адекватность выбранной
архитектуры сети.")
```


Прогноз на обучающей выборке
Истинные значения
Прогноз
0.2
0.1
0.0
−0.1
−0.2
0
10
20
30
40
50
60
70
80


Изменение ошибки в процессе обучения
MSE
0.028
0.027
0.026
0.025
0.024
0.023
0.022
0
200
400
600
800
1000





Вывод: изучил применение нелинейной искусственной нейронной сети с одним скрытым слоем для решения задачи регрессии и прогнозирования, реализовал обучение сети на синтетических данных и оценила точность полученной модели.