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

Лабораторная работа №5
По дисциплине «ОМО»

Выполнил:
Студент 3-го курса
Группы АС-65
Егоренков Н. Д.
Проверил:
Крощенко А.А.

Брест 2025

Цель: научиться моделировать прогнозирующую нелинейную ИНС.

Вариант 5

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

Варианты заданий приведены в следующей таблице:

№ варианта	a	b	c	d	Кол-во входов ИНС	Кол-во НЭ в скрытом слое
1	0.1	0.1	0.05	0.1	6	2
2	0.2	0.2	0.06	0.2	8	3
3	0.3	0.3	0.07	0.3	10	4
4	0.4	0.4	0.08	0.4	6	2
5	0.1	0.5	0.09	0.5	8	3
6	0.2	0.6	0.05	0.6	10	4
7	0.3	0.1	0.06	0.1	6	2
8	0.4	0.2	0.07	0.2	8	3
9	0.1	0.3	0.08	0.3	10	4
10	0.2	0.4	0.09	0.4	6	2
11	0.3	0.5	0.05	0.5	8	3

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

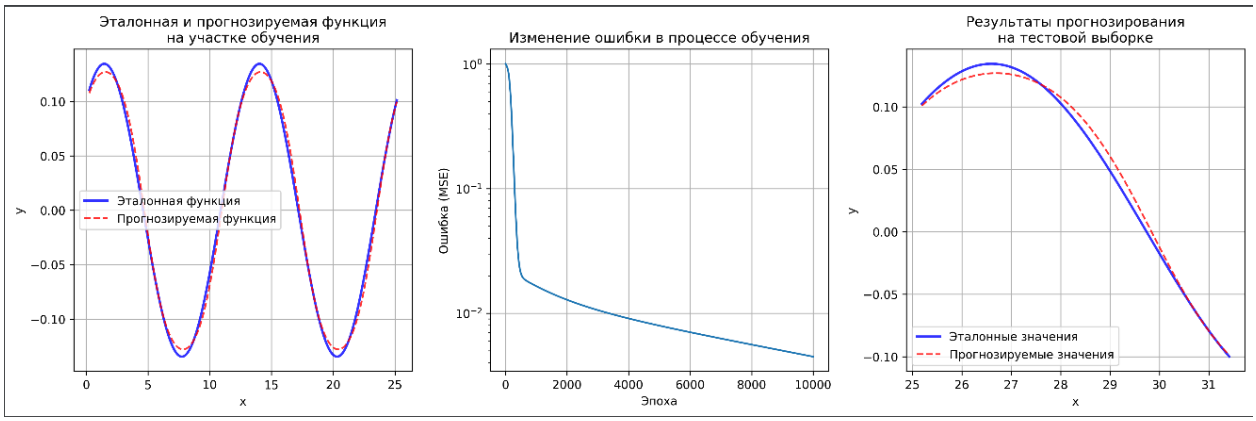
Ход работы

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error

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

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

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

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

def linear(x):
    return x
```

```
x_full = np.linspace(0, 10 * np.pi, 1000)
y_full = generate_data(a, b, c, d, x_full)

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

X, y = create_dataset(y_full, num_inputs)

train_size = int(0.8 * len(X))
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

x_train_plot = x_full[num_inputs:num_inputs + train_size]
x_test_plot = x_full[num_inputs + train_size:num_inputs + len(X)]

X_mean = np.mean(X_train)
X_std = np.std(X_train)
y_mean = np.mean(y_train)
y_std = np.std(y_train)

X_train_norm = (X_train - X_mean) / X_std
X_test_norm = (X_test - X_mean) / X_std
y_train_norm = (y_train - y_mean) / y_std

np.random.seed(42)
W1 = np.random.randn(num_inputs, num_neurons_hidden) * 0.1
b1 = np.zeros((1, num_neurons_hidden))
W2 = np.random.randn(num_neurons_hidden, 1) * 0.1
b2 = np.zeros((1, 1))

learning_rate = 0.01
epochs = 10000
loss_history = []

for epoch in range(epochs):
    hidden_input = np.dot(X_train_norm, W1) + b1
    hidden_output = sigmoid(hidden_input)
    output = linear(np.dot(hidden_output, W2) + b2)

    loss = np.mean((output - y_train_norm.reshape(-1, 1)) ** 2)
    loss_history.append(loss)

    d_output = 2 * (output - y_train_norm.reshape(-1, 1)) / len(X_train_norm)
    d_W2 = np.dot(hidden_output.T, d_output)
    d_b2 = np.sum(d_output, axis=0, keepdims=True)

    d_hidden = np.dot(d_output, W2.T) * sigmoid_derivative(hidden_output)
    d_W1 = np.dot(X_train_norm.T, d_hidden)
    d_b1 = np.sum(d_hidden, axis=0, keepdims=True)

    W2 -= learning_rate * d_W2
    b2 -= learning_rate * d_b2
    W1 -= learning_rate * d_W1
```

```
    b1 -= learning_rate * d_b1

    if epoch % 1000 == 0:
        print(f"Эпоха {epoch}, Ошибка: {loss:.6f}")

hidden_output_train = sigmoid(np.dot(X_train_norm, W1) + b1)
y_train_pred_norm = linear(np.dot(hidden_output_train, W2) + b2)
y_train_pred = y_train_pred_norm * y_std + y_mean

hidden_output_test = sigmoid(np.dot(X_test_norm, W1) + b1)
y_test_pred_norm = linear(np.dot(hidden_output_test, W2) + b2)
y_test_pred = y_test_pred_norm * y_std + y_mean

print("\nРезультаты обучения:")
print("MSE на обучающей выборке:", mean_squared_error(y_train,
y_train_pred.flatten()))
print("MSE на тестовой выборке:", mean_squared_error(y_test,
y_test_pred.flatten()))

plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
plt.plot(x_train_plot, y_train, label='Эталонная функция', linewidth=2, alpha=0.8,
color='blue')
plt.plot(x_train_plot, y_train_pred, label='Прогнозируемая функция',
linewidth=1.5, alpha=0.8, color='red', linestyle='--')
plt.title('Эталонная и прогнозируемая функция\nна участке обучения')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)

plt.subplot(1, 3, 2)
plt.plot(loss_history)
plt.title('Изменение ошибки в процессе обучения')
plt.xlabel('Эпоха')
plt.ylabel('Ошибка (MSE)')
plt.yscale('log')
plt.grid(True)

plt.subplot(1, 3, 3)
plt.plot(x_test_plot, y_test, label='Эталонные значения', linewidth=2, alpha=0.8,
color='blue')
plt.plot(x_test_plot, y_test_pred, label='Прогнозируемые значения', linewidth=1.5,
alpha=0.8, color='red', linestyle='--')
plt.title('Результаты прогнозирования\nна тестовой выборке')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)

plt.tight_layout()

plt.savefig('neural_network_results.png', dpi=300, bbox_inches='tight')
plt.close()

print("\nРезультаты прогнозирования на обучающей выборке (первые 20 значений):")
```

```
print("Эталонное значение | Полученное значение | Отклонение")
print("-" * 55)
train_deviations = y_train - y_train_pred.flatten()
for i in range(min(20, len(y_train))):
    print(f"{y_train[i]:17.6f} | {y_train_pred.flatten()[i]:19.6f} |
{train_deviations[i]:10.6f}")

print(f"\nСтатистика отклонений на обучающей выборке:")
print(f"Среднее отклонение: {np.mean(train_deviations):.6f}")
print(f"Стандартное отклонение: {np.std(train_deviations):.6f}")
print(f"Максимальное отклонение: {np.max(np.abs(train_deviations)):.6f}")
print(f"Средняя абсолютная ошибка: {np.mean(np.abs(train_deviations)):.6f}")

print("\nГрафики сохранены в файл: neural_network_results.png")
```



Вывод: мы научились моделировать прогнозирующую нелинейную ИНС.