

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

Лабораторная работа №5
По дисциплине: «ОМО»
Тема:” Нелинейные ИНС в задачах регрессии”

Выполнил:
Студент 3-го курса
Группы АС-66
Янчук А.Ю.
Проверил:
Крощенко А.А.

Брест 2025

Цель: Выполнить моделирование прогнозирующей нелинейной ИНС.

Вариант 13

Задание:

- 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
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

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

a, b, c, d = 0.2, 0.2, 0.06, 0.2
window_size = 8
hidden_size = 3
epochs = 2500
lr = 0.05
train_ratio = 0.7

OUT_DIR = "lab5_variant2_results"
os.makedirs(OUT_DIR, exist_ok=True)

def sigmoid(x):
    x_clip = np.clip(x, -50, 50)
    return 1.0 / (1.0 + np.exp(-x_clip))

def sigmoid_deriv_from_activation(a_sigmoid):
    return a_sigmoid * (1.0 - a_sigmoid)

def generate_series(a, b, c, d, x_min=-10, x_max=10, step=0.1):
    x_vals = np.arange(x_min, x_max + step, step)
    y_vals = a * np.cos(b * x_vals) + c * np.sin(d * x_vals)
    return x_vals, y_vals
```

```

def make_supervised_from_series(y_vals, window):
    X, Y = [], []
    for i in range(len(y_vals) - window):
        X.append(y_vals[i:i + window])
        Y.append(y_vals[i + window])
    X = np.array(X, dtype=np.float64)
    Y = np.array(Y, dtype=np.float64).reshape(-1, 1)
    return X, Y

def init_weights(input_size, hidden_size, output_size=1, seed=42):
    rng = np.random.default_rng(seed)
    W1 = rng.normal(0.0, np.sqrt(1.0 / input_size), size=(input_size, hidden_size))
    b1 = np.zeros((1, hidden_size), dtype=np.float64)

    W2 = rng.normal(0.0, np.sqrt(1.0 / hidden_size), size=(hidden_size, output_size))
    b2 = np.zeros((1, output_size), dtype=np.float64)
    return W1, b1, W2, b2

def forward(X, W1, b1, W2, b2):
    z1 = X @ W1 + b1
    a1 = sigmoid(z1)
    z2 = a1 @ W2 + b2
    y_pred = z2
    cache = (X, z1, a1, z2)
    return y_pred, cache

def predict(X, W1, b1, W2, b2):
    _, _, _, _ = forward(X, W1, b1, W2, b2)
    z1 = X @ W1 + b1
    a1 = sigmoid(z1)
    return a1 @ W2 + b2

def compute_mse(y_pred, y_true):
    err = y_pred - y_true
    return np.mean(err ** 2), err

def backward_and_update(W1, b1, W2, b2, cache, err, lr):
    X, z1, a1, z2 = cache
    N = X.shape[0]

    grad_out = 2.0 * err / N

    dW2 = a1.T @ grad_out
    db2 = np.sum(grad_out, axis=0, keepdims=True)

    da1 = grad_out @ W2.T
    dz1 = da1 * sigmoid_deriv_from_activation(a1)
    dW1 = X.T @ dz1
    db1 = np.sum(dz1, axis=0, keepdims=True)

    W1 -= lr * dW1
    b1 -= lr * db1
    W2 -= lr * dW2
    b2 -= lr * db2

    return W1, b1, W2, b2

def train(X_train, y_train, W1, b1, W2, b2, epochs, lr):
    loss_history = []
    for epoch in range(1, epochs + 1):
        y_pred, cache = forward(X_train, W1, b1, W2, b2)
        loss, err = compute_mse(y_pred, y_train)
        loss_history.append(loss)

```

```

        loss_history.append(loss)

    W1, b1, W2, b2 = backward_and_update(W1, b1, W2, b2, cache, err, lr)

    if epoch % 250 == 0 or epoch == 1:
        print(f"Epoch {epoch:4d} | MSE={loss:.8f}")

    return W1, b1, W2, b2, loss_history

def main():
    x_vals, y_vals = generate_series(a, b, c, d, x_min=-10, x_max=10,
                                     step=0.1)

    X, Y = make_supervised_from_series(y_vals, window_size)
    split = int(train_ratio * len(X))
    X_train, X_test = X[:split], X[split:]
    y_train, y_test = Y[:split], Y[split:]

    W1, b1, W2, b2 = init_weights(window_size, hidden_size, output_size=1,
                                   seed=42)

    W1, b1, W2, b2, loss_history = train(X_train, y_train, W1, b1, W2, b2,
                                          epochs, lr)

    y_train_pred = predict(X_train, W1, b1, W2, b2)
    y_test_pred = predict(X_test, W1, b1, W2, b2)

    train_df = pd.DataFrame({
        "Эталонное значение": y_train.flatten(),
        "Полученное значение": y_train_pred.flatten(),
        "Отклонение": (y_train_pred - y_train).flatten()
    })
    test_df = pd.DataFrame({
        "Эталонное значение": y_test.flatten(),
        "Полученное значение": y_test_pred.flatten(),
        "Отклонение": (y_test_pred - y_test).flatten()
    })

    print("\n==== Обучающая выборка (первые 10) ===")
    print(train_df.head(10))
    print("\n==== Тестовая выборка (первые 10) ===")
    print(test_df.head(10))

    train_df.to_csv(os.path.join(OUT_DIR, "train_results.csv"), index=False,
                    encoding="utf-8-sig")
    test_df.to_csv(os.path.join(OUT_DIR, "test_results.csv"), index=False,
                   encoding="utf-8-sig")

    plt.figure(figsize=(8, 4))
    plt.plot(loss_history, color="tab:blue", label="MSE")
    plt.title("График изменения ошибки (MSE) по эпохам")
    plt.xlabel("Эпоха")
    plt.ylabel("MSE")
    plt.grid(True, alpha=0.3)
    plt.legend()
    plt.tight_layout()
    plt.savefig(os.path.join(OUT_DIR, "loss_curve.png"), dpi=150)
    plt.show()

    plt.figure(figsize=(10, 5))
    plt.plot(range(len(y_vals)), y_vals, lw=2, label="Эталонная функция")
    plt.plot(range(window_size, window_size + len(y_train_pred)),
             y_train_pred.flatten(), "--", label="Прогноз (train)")
    plt.plot(range(window_size + len(y_train_pred),

```

```

        window_size + len(y_train_pred) + len(y_test_pred)),
        y_test_pred.flatten(), "--", label="Прогноз (test)")
plt.title("Прогнозируемая функция на участке обучения и теста")
plt.xlabel("Индекс точки")
plt.ylabel("Значение y")
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig(os.path.join(OUT_DIR, "prediction_vs_true.png"), dpi=150)
plt.show()

print(f"\nФайлы сохранены в: {OUT_DIR}")
print(" - train_results.csv, test_results.csv")
print(" - loss_curve.png, prediction_vs_true.png")

if __name__ == "__main__":
    main()

```

Результаты:

```

Epoch    1 | MSE=0.04330865
Epoch   250 | MSE=0.01033210
Epoch   500 | MSE=0.00974246
Epoch   750 | MSE=0.00908502
Epoch  1000 | MSE=0.00830223
Epoch  1250 | MSE=0.00735931
Epoch  1500 | MSE=0.00625692
Epoch  1750 | MSE=0.00504398
Epoch  2000 | MSE=0.00381674
Epoch  2250 | MSE=0.00269333
Epoch  2500 | MSE=0.00177069

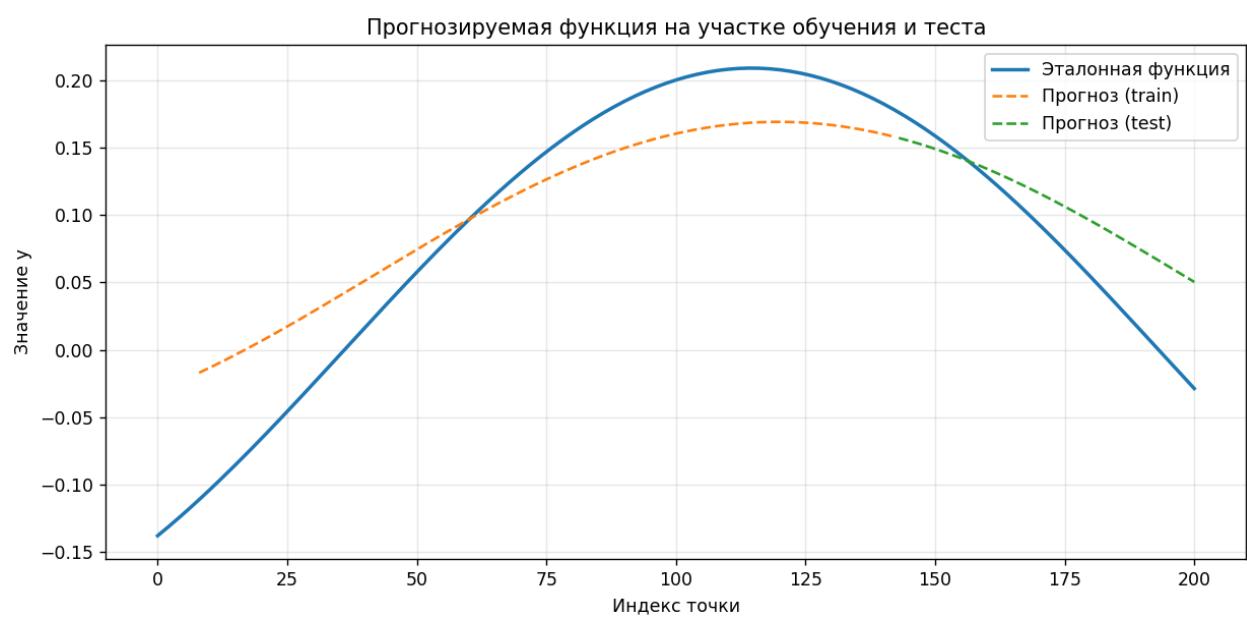
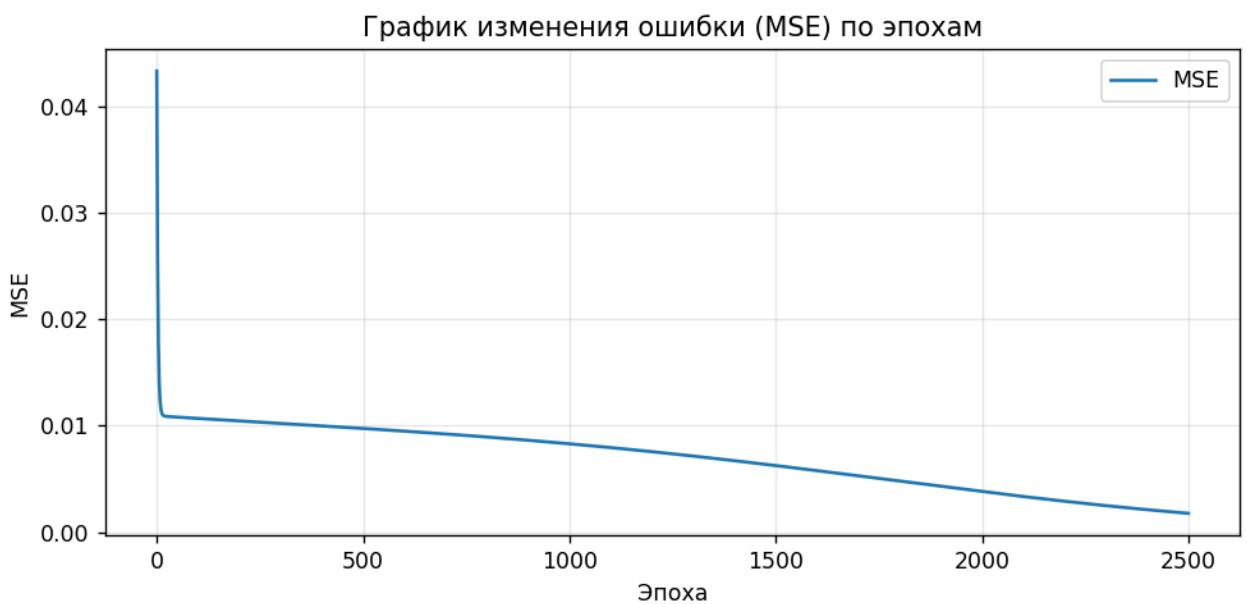
```

== Обучающая выборка (первые 10) ==

	Эталонное значение	Полученное значение	Отклонение
0	-0.111032	-0.017001	0.094031
1	-0.107473	-0.015181	0.092292
2	-0.103871	-0.013333	0.090538
3	-0.100228	-0.011457	0.088771
4	-0.096545	-0.009554	0.086990
5	-0.092823	-0.007625	0.085198
6	-0.089064	-0.005670	0.083393
7	-0.085269	-0.003691	0.081578
8	-0.081440	-0.001687	0.079753
9	-0.077579	0.000339	0.077918

== Тестовая выборка (первые 10) ==

	Эталонное значение	Полученное значение	Отклонение
0	0.175958	0.157015	-0.018943
1	0.173675	0.155980	-0.017695
2	0.171322	0.154903	-0.016418
3	0.168900	0.153786	-0.015114
4	0.166411	0.152630	-0.013781
5	0.163855	0.151434	-0.012421
6	0.161234	0.150199	-0.011035
7	0.158549	0.148926	-0.009623
8	0.155800	0.147614	-0.008185
9	0.152988	0.146265	-0.006723



Вывод: На практике выполнили моделирование прогнозирующей нелинейной ИНС.