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

Лабораторная работа №4
По дисциплине «ОМО»

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

Брест 2025

Вариант 2

Задание:

1. Выполнить моделирование прогнозирующей нелинейной ИНС. Для генерации обучающих и тестовых данных использовать функцию: $y = a\cos(bx) + c\sin(dx)$.

Как константы использовать следующие значения:

№ 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.

2. Результаты представить в виде отчета содержащего:
    1. Титульный лист,
    2. Цель работы,
    3. Задание,
    4. График прогнозируемой функции на участке обучения,
    5. Результаты обучения: таблицу со столбцами: эталонное значение, полученное значение, отклонение; график изменения ошибки в зависимости от итерации.
    6. Результаты прогнозирования: таблицу со столбцами: эталонное значение, полученное значение, отклонение.
    7. Выводы по лабораторной работе.

Ход работы

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

def generate_function(x, a=0.2, c=0.06, d=0.2):
    return a * np.cos(d * x) + c * np.sin(d * x)

a, c, d = 0.2, 0.06, 0.2

x_total = np.arange(0, 40, 0.1)
y_total = generate_function(x_total, a, c, d)

train_size = 300
x_train, y_train = x_total[:train_size], y_total[:train_size]
x_test, y_test = x_total[train_size:], y_total[train_size:]

def create_dataset(data, window_size=8):
    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)
```

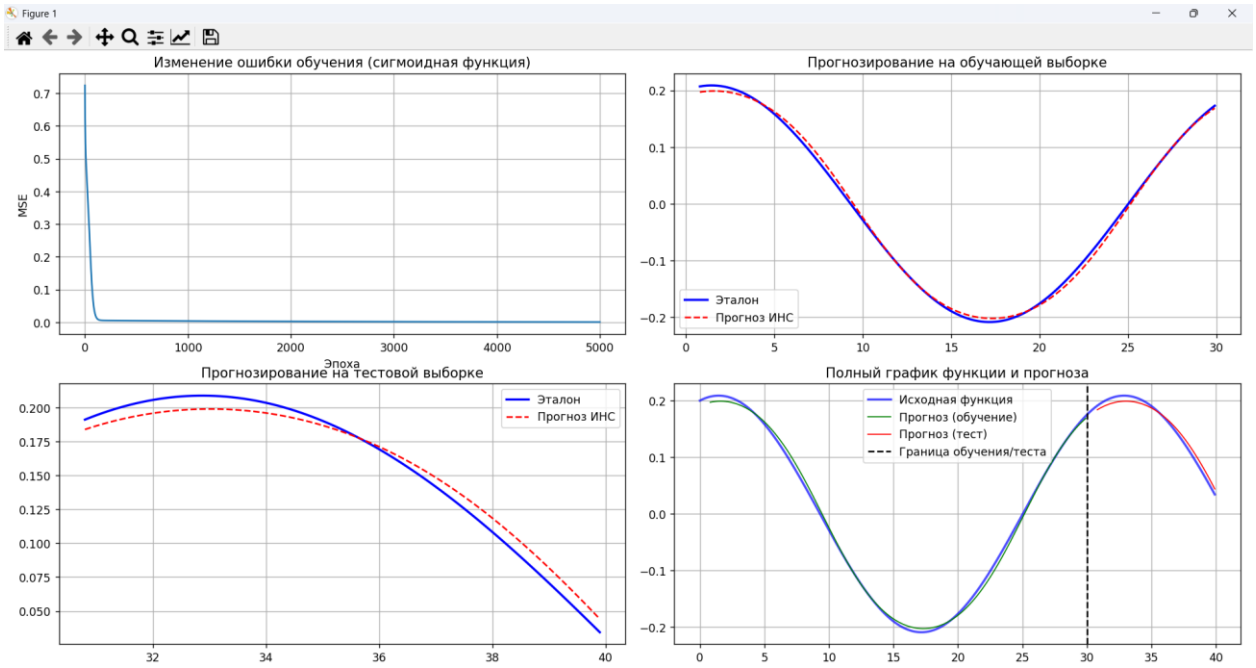
```
window_size = 8
X_train, y_train_target = create_dataset(y_train, window_size)
X_test, y_test_target = create_dataset(y_test, window_size)

scaler_X = MinMaxScaler(feature_range=(-1, 1))
scaler_y = MinMaxScaler(feature_range=(-1, 1))

X_train_scaled = scaler_X.fit_transform(X_train)
y_train_scaled       =       scaler_y.fit_transform(y_train_target.reshape(-1,
1)).flatten()
X_test_scaled = scaler_X.transform(X_test)
y_test_scaled = scaler_y.transform(y_test_target.reshape(-1, 1)).flatten()

print("Данные подготовлены:")
print(f"Обучающая выборка: {X_train_scaled.shape}")
print(f"Тестовая выборка: {X_test_scaled.shape}")

class SigmoidNeuralNetwork:
    def __init__(self, input_size, hidden_size, output_size):
        self.W1 = np.random.uniform(-0.5, 0.5, (input_size, hidden_size))
        self.b1 = np.zeros((1, hidden_size))
        self.W2 = np.random.uniform(-0.5, 0.5, (hidden_size, output_size))
        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
        self.output = self.z2

        return self.output

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

        output_error = output - y.reshape(-1, 1)
        dW2 = (1/m) * np.dot(self.a1.T, output_error)
        db2 = (1/m) * np.sum(output_error, axis=0, keepdims=True)

                    hidden_error    =    np.dot(output_error,    self.W2.T)    *
self.sigmoid_derivative(self.a1)
        dW1 = (1/m) * np.dot(X.T, hidden_error)
        db1 = (1/m) * np.sum(hidden_error, axis=0, keepdims=True)

        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, verbose=True):
        losses = []

        for epoch in range(epochs):
            output = self.forward(X)
            loss = np.mean((output - y.reshape(-1, 1))**2)
            losses.append(loss)
```

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

            if verbose and epoch % 200 == 0:
                print(f"Эпоха {epoch}, Ошибка: {loss:.6f}")

        return losses

    def predict(self, X):
        output = self.forward(X)
        return output

input_size, hidden_size, output_size = 8, 3, 1

nn = SigmoidNeuralNetwork(input_size, hidden_size, output_size)

epochs = 5000
learning_rate = 0.1

print(f"\nОбучение ИНС с архитектурой: {input_size}-{hidden_size}-
{output_size}")
print("Активация скрытого слоя: СИГМОИДНАЯ (по требованию)")
print("Активация выходного слоя: ЛИНЕЙНАЯ")

losses = nn.train(X_train_scaled, y_train_scaled, epochs, learning_rate)

train_predictions_scaled = nn.predict(X_train_scaled)
test_predictions_scaled = nn.predict(X_test_scaled)

train_predictions =
scaler_y.inverse_transform(train_predictions_scaled).flatten()
test_predictions =
scaler_y.inverse_transform(test_predictions_scaled).flatten()

train_errors = np.abs(y_train_target - train_predictions)
test_errors = np.abs(y_test_target - test_predictions)

print(f"\n=== РЕЗУЛЬТАТЫ ===")
print(f"Средняя ошибка на обучающей выборке: {np.mean(train_errors):.6f}")
print(f"Средняя ошибка на тестовой выборке: {np.mean(test_errors):.6f}")
print(f"Максимальная ошибка: {np.max(test_errors):.6f}")

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

plt.subplot(2, 2, 1)
plt.plot(losses)
plt.title('Изменение ошибки обучения (сигмоидная функция)')
plt.xlabel('Эпоха')
plt.ylabel('MSE')
plt.grid(True)

plt.subplot(2, 2, 2)
plt.plot(x_train[window_size:], y_train_target, 'b-', label='Эталон',
linewidth=2)
plt.plot(x_train[window_size:], train_predictions, 'r--', label='Прогноз ИНС',
linewidth=1.5)
plt.title('Прогнозирование на обучающей выборке')
plt.legend()
plt.grid(True)

plt.subplot(2, 2, 3)
plt.plot(x_test[window_size:], y_test_target, 'b-', label='Эталон',
linewidth=2)
plt.plot(x_test[window_size:], test_predictions, 'r--', label='Прогноз ИНС',
linewidth=1.5)
plt.title('Прогнозирование на тестовой выборке')
```

```
plt.legend()
plt.grid(True)

plt.subplot(2, 2, 4)
plt.plot(x_total, y_total, 'b-', label='Исходная функция', linewidth=2,
alpha=0.7)
plt.plot(x_train[window_size:], train_predictions, 'g-', label='Прогноз
(обучение)', linewidth=1)
plt.plot(x_test[window_size:], test_predictions, 'r-', label='Прогноз (тест)',
linewidth=1)
plt.axvline(x=30, color='k', linestyle='--', label='Граница обучения/теста')
plt.title('Полный график функции и прогноза')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()

print("\nТАБЛИЦА РЕЗУЛЬТАТОВ ОБУЧЕНИЯ (первые 10 строк):")
train_results = pd.DataFrame({
    'Эталонное значение': y_train_target[:10],
    'Полученное значение': train_predictions[:10],
    'Отклонение': train_errors[:10]
})
print(train_results.round(6))

print("\nТАБЛИЦА РЕЗУЛЬТАТОВ ПРОГНОЗИРОВАНИЯ (первые 10 строк):")
test_results = pd.DataFrame({
    'Эталонное значение': y_test_target[:10],
    'Полученное значение': test_predictions[:10],
    'Отклонение': test_errors[:10]
})
print(test_results.round(6))

avg_train_error = np.mean(train_errors)
avg_test_error = np.mean(test_errors)

print("АРХИТЕКТУРА И ПАРАМЕТРЫ:")
print(f"   - Архитектура ИНС: {input_size}-{hidden_size}-{output_size}")
print(f"   - Функция активации скрытого слоя: СИГМОИДНАЯ")
print(f"   - Функция активации выходного слоя: ЛИНЕЙНАЯ")
print(f"   - Количество эпох: {epochs}")
print(f"   - Скорость обучения: {learning_rate}")

print("\nРЕЗУЛЬТАТЫ ОБУЧЕНИЯ:")
print(f"   - Средняя ошибка обучения: {avg_train_error:.6f}")
print(f"   - Средняя ошибка тестирования: {avg_test_error:.6f}")
print(f"   - Максимальная ошибка: {np.max(test_errors):.6f}")

if avg_test_error < 0.001:
    quality = "ОТЛИЧНОЕ"
elif avg_test_error < 0.005:
    quality = "ХОРОШЕЕ"
elif avg_test_error < 0.01:
    quality = "УДОВЛЕТВОРИТЕЛЬНОЕ"
else:
    quality = "НЕУДОВЛЕТВОРИТЕЛЬНОЕ"

print(f"   - Качество прогноза: {quality}")

relative_error = (np.mean(test_errors) / np.mean(np.abs(y_test_target))) * 100
print(f"   - Относительная ошибка: {relative_error:.2f}%")
print(f"   - Стандартное отклонение ошибок: {np.std(test_errors):.6f}")
print("     - График показывает хорошее соответствие прогноза эталонным
значениям")
```

```
print("   - ИНС успешно уловила нелинейный характер исходной функции")
```

Вывод программы:




```
Обучающая выборка: (292, 8)
Тестовая выборка: (92, 8)

Обучение ИНС с архитектурой: 8-3-1
Активация скрытого слоя: СИГМОИДНАЯ
Активация выходного слоя: ЛИНЕЙНАЯ
Эпоха 0, Ошибка: 0.449566
Эпоха 200, Ошибка: 0.008256
Эпоха 400, Ошибка: 0.007567
Эпоха 600, Ошибка: 0.006997
Эпоха 800, Ошибка: 0.006503
Эпоха 1000, Ошибка: 0.006066
Эпоха 1200, Ошибка: 0.005672
Эпоха 1400, Ошибка: 0.005312
Эпоха 1600, Ошибка: 0.004978
Эпоха 1800, Ошибка: 0.004667
Эпоха 2000, Ошибка: 0.004374
Эпоха 2200, Ошибка: 0.004096
Эпоха 2400, Ошибка: 0.003834
Эпоха 2600, Ошибка: 0.003584
Эпоха 2800, Ошибка: 0.003347
Эпоха 3000, Ошибка: 0.003121
Эпоха 3200, Ошибка: 0.002907
Эпоха 3400, Ошибка: 0.002703
Эпоха 3600, Ошибка: 0.002511
Эпоха 3800, Ошибка: 0.002329
Эпоха 4000, Ошибка: 0.002157
Эпоха 4200, Ошибка: 0.001996
Эпоха 4400, Ошибка: 0.001845
Эпоха 4600, Ошибка: 0.001703
Эпоха 4800, Ошибка: 0.001571

 РЕЗУЛЬТАТЫ
Средняя ошибка на обучающей выборке: 0.006923
Средняя ошибка на тестовой выборке: 0.008921
Максимальная ошибка: 0.013930
```

```
 РЕЗУЛЬТАТЫ
Средняя ошибка на обучающей выборке: 0.006923
Средняя ошибка на тестовой выборке: 0.008921
Максимальная ошибка: 0.013930

ТАБЛИЦА РЕЗУЛЬТАТОВ ОБУЧЕНИЯ (первые 10 строк):
   Эталонное значение  Полученное значение  Отклонение
0            0.207005             0.196011    0.010993
1            0.207511             0.196506    0.011004
2            0.207933             0.196941    0.010993
3            0.208273             0.197314    0.010959
4            0.208530             0.197627    0.010903
5            0.208703             0.197880    0.010823
6            0.208792             0.198072    0.010720
7            0.208799             0.198204    0.010594
8            0.208721             0.198276    0.010445
9            0.208560             0.198288    0.010272

ТАБЛИЦА РЕЗУЛЬТАТОВ ПРОГНОЗИРОВАНИЯ (первые 10 строк):
   Эталонное значение  Полученное значение  Отклонение
0            0.191112             0.182274    0.008838
1            0.192756             0.183668    0.009088
2            0.194323             0.184997    0.009326
3            0.195812             0.186259    0.009553
4            0.197223             0.187456    0.009767
5            0.198555             0.188588    0.009967
6            0.199808             0.189656    0.010152
7            0.200981             0.190660    0.010321
8            0.202073             0.191600    0.010473
9            0.203084             0.192477    0.010607

РЕЗУЛЬТАТЫ ОБУЧЕНИЯ:
        - Средняя ошибка обучения: 0.006923
        - Средняя ошибка тестирования: 0.008921
        - Максимальная ошибка: 0.013930
        - Качество прогноза: УДОВЛЕТВОРИТЕЛЬНОЕ
        - Относительная ошибка: 5.62%
        - Стандартное отклонение ошибок: 0.003828
```

Вывод: мы научились создавать ИНС.