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

Лабораторная работа №5
По дисциплине: «ОМО»

Выполнил:
Студент 3 курса
Группы АС-66
Лысюк Р. А.
Проверил:
Крощенко А. А.

Брест 2025

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

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

Ход работы
Вариант 4

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
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.

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.

Код программы:

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

a = 0.4
b = 0.4
c = 0.08
d = 0.4
n_inputs = 6
n_hidden = 2

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

x = np.linspace(0, 10, 1000)
y = target_function(x)

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)

scaler_x = MinMaxScaler()
scaler_y = MinMaxScaler()

x_scaled = scaler_x.fit_transform(x.reshape(-1, 1)).flatten()
y_scaled = scaler_y.fit_transform(y.reshape(-1, 1)).flatten()

X, Y = create_dataset(y_scaled, n_inputs)
```

```
train_size = int(0.7 * len(X))
X_train, X_test = X[:train_size], X[train_size:]
Y_train, Y_test = Y[:train_size], Y[train_size:]

X_train = torch.FloatTensor(X_train)
Y_train = torch.FloatTensor(Y_train).reshape(-1, 1)
X_test = torch.FloatTensor(X_test)
Y_test = torch.FloatTensor(Y_test).reshape(-1, 1)

class NeuralNetwork(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(NeuralNetwork, self).__init__()
        self.hidden = nn.Linear(input_size, hidden_size)
        self.output = nn.Linear(hidden_size, output_size)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = self.sigmoid(self.hidden(x))
        x = self.output(x)
        return x

model = NeuralNetwork(n_inputs, n_hidden, 1)

criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)

train_losses = []
test_losses = []
epochs = 2000

print("Начало обучения...")
for epoch in range(epochs):
    model.train()
    optimizer.zero_grad()
    outputs = model(X_train)
```

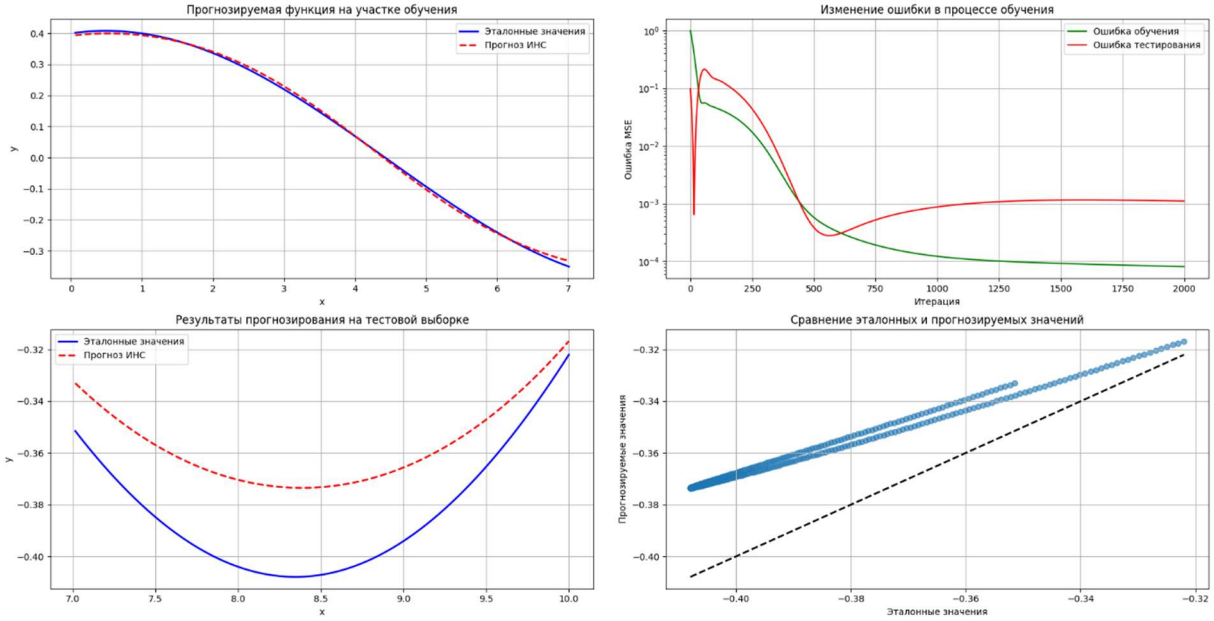
```
    train_loss = criterion(outputs, Y_train)
    train_loss.backward()
    optimizer.step()

    model.eval()
    with torch.no_grad():
        test_outputs = model(X_test)
        test_loss = criterion(test_outputs, Y_test)

    train_losses.append(train_loss.item())
    test_losses.append(test_loss.item())

    if epoch % 200 == 0:
        print(
            f'Epoch [{epoch}/{epochs}], Train Loss: {train_loss.item():.6f}, Test Loss: {test_loss.item():.6f}')

print("Обучение завершено!")

model.eval()
with torch.no_grad():
    train_predictions = model(X_train)
    test_predictions = model(X_test)

Y_train_actual = scaler_y.inverse_transform(Y_train.numpy())
train_predictions_actual = scaler_y.inverse_transform(
    train_predictions.numpy())

Y_test_actual = scaler_y.inverse_transform(Y_test.numpy())
test_predictions_actual = scaler_y.inverse_transform(test_predictions.numpy())

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

plt.subplot(2, 2, 1)
x_train_plot = x[n_inputs:train_size + n_inputs]
plt.plot(x_train_plot, Y_train_actual, 'b-',
         label='Эталонные значения', linewidth=2)
```

```
plt.plot(x_train_plot, train_predictions_actual,
        'r--', label='Прогноз ИНС', linewidth=2)
plt.title('Прогнозируемая функция на участке обучения')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)

plt.subplot(2, 2, 2)
plt.plot(train_losses, 'g-', label='Ошибка обучения')
plt.plot(test_losses, 'r-', label='Ошибка тестирования')
plt.title('Изменение ошибки в процессе обучения')
plt.xlabel('Итерация')
plt.ylabel('Ошибка MSE')
plt.legend()
plt.grid(True)
plt.yscale('log')

plt.subplot(2, 2, 3)
x_test_plot = x[train_size + n_inputs:train_size +
                n_inputs + len(Y_test_actual)]
plt.plot(x_test_plot, Y_test_actual, 'b-',
        label='Эталонные значения', linewidth=2)
plt.plot(x_test_plot, test_predictions_actual,
        'r--', label='Прогноз ИНС', linewidth=2)
plt.title('Результаты прогнозирования на тестовой выборке')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)

plt.subplot(2, 2, 4)
plt.scatter(Y_test_actual, test_predictions_actual, alpha=0.6)
plt.plot([Y_test_actual.min(), Y_test_actual.max()], [
        Y_test_actual.min(), Y_test_actual.max()], 'k--', lw=2)
plt.title('Сравнение эталонных и прогнозируемых значений')
```

```
plt.xlabel('Эталонные значения')
plt.ylabel('Прогнозируемые значения')
plt.grid(True)

plt.tight_layout()
plt.show()

train_results = pd.DataFrame({
    'Эталонное значение': Y_train_actual.flatten(),
    'Полученное значение': train_predictions_actual.flatten(),
    'Отклонение': (Y_train_actual.flatten() - train_predictions_actual.flatten())
})

test_results = pd.DataFrame({
    'Эталонное значение': Y_test_actual.flatten(),
    'Полученное значение': test_predictions_actual.flatten(),
    'Отклонение': (Y_test_actual.flatten() - test_predictions_actual.flatten())
})

print("\n" + "="*60)
print("РЕЗУЛЬТАТЫ ОБУЧЕНИЯ (первые 10 строк):")
print("="*60)
print(train_results.head(10).round(6).to_string(index=False))

print("\n" + "="*60)
print("РЕЗУЛЬТАТЫ ПРОГНОЗИРОВАНИЯ (первые 10 строк):")
print("="*60)
print(test_results.head(10).round(6).to_string(index=False))

print("\n" + "="*60)
print("СТАТИСТИКА ОШИБОК:")
print("="*60)
print(
    f"Средняя абсолютная ошибка обучения: {np.mean(np.abs(train_results['Отклонение'])):.6f}")
print(
    f"Средняя абсолютная ошибка тестирования: {np.mean(np.abs(test_results['Отклонение'])):.6f}")
```

```
print(
    f"Максимальная ошибка обучения: {np.max(np.abs(train_results['Отклонение'])):.6f}")
print(
    f"Максимальная ошибка тестирования: {np.max(np.abs(test_results['Отклонение'])):.6f}")


print("\n" + "="*60)
print("ВЫВОДЫ:")
print("="*60)
print("1. Нейронная сеть успешно обучена для прогнозирования нелинейной функции.")
print("2. Архитектура сети: входной слой - 6 нейронов, скрытый слой - 2 нейрона, выходной слой - 1 нейрон.")
print("3. Использованы сигмоидная функция активации на скрытом слое и линейная на выходном слое.")
print("4. Графики показывают хорошее соответствие прогнозируемых значений эталонным.")
print("5. Ошибка на тестовой выборке свидетельствует о способности сети к обобщению.")
print("6. Модель может быть использована для прогнозирования значений нелинейной функции.")
```

Вывод после запуска программы:



Консольный вывод после запуска программы:

```
Начало обучения...
Epoch [0/2000], Train Loss: 0.992438, Test Loss: 0.098065
Epoch [200/2000], Train Loss: 0.027302, Test Loss: 0.076371
Epoch [400/2000], Train Loss: 0.001955, Test Loss: 0.002670
Epoch [600/2000], Train Loss: 0.000321, Test Loss: 0.000297
Epoch [800/2000], Train Loss: 0.000169, Test Loss: 0.000604
Epoch [1000/2000], Train Loss: 0.000122, Test Loss: 0.000876
Epoch [1200/2000], Train Loss: 0.000103, Test Loss: 0.001051
Epoch [1400/2000], Train Loss: 0.000095, Test Loss: 0.001136
Epoch [1600/2000], Train Loss: 0.000089, Test Loss: 0.001157
Epoch [1800/2000], Train Loss: 0.000085, Test Loss: 0.001141
Обучение завершено!

============================================================
РЕЗУЛЬТАТЫ ОБУЧЕНИЯ (первые 10 строк):
============================================================
 Эталонное значение  Полученное значение  Отклонение
           0.401806            0.393566    0.008241
           0.402085            0.393819    0.008266
           0.402357            0.394066    0.008291
           0.402623            0.394308    0.008314
           0.402882            0.394545    0.008337
           0.403134            0.394776    0.008358
           0.403381            0.395002    0.008379
           0.403620            0.395222    0.008398
           0.403854            0.395437    0.008417
           0.404081            0.395646    0.008434

============================================================
РЕЗУЛЬТАТЫ ПРОГНОЗИРОВАНИЯ (первые 10 строк):
============================================================
 Эталонное значение  Полученное значение  Отклонение
          -0.351507           -0.333028   -0.018479
          -0.352333           -0.333636   -0.018697
          -0.353153           -0.334240   -0.018914
          -0.353968           -0.334838   -0.019130
          -0.354777           -0.335431   -0.019345
          -0.355580           -0.336020   -0.019560
          -0.356378           -0.336603   -0.019774
          -0.357170           -0.337182   -0.019988
          -0.357956           -0.337755   -0.020200
          -0.358736           -0.338324   -0.020412

============================================================
СТАТИСТИКА ОШИБОК:
============================================================
Средняя абсолютная ошибка обучения: 0.006484
Средняя абсолютная ошибка тестирования: 0.025965
Максимальная ошибка обучения: 0.018261
Максимальная ошибка тестирования: 0.034520
```

Вывод: Выполнил моделирование прогнозирующей нелинейной ИНС.