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(-100, 300, 4000)
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)
```

```
# Определяем индексы для обучения (50-100) и теста (100-150)
start_train = np.where(x >= 50)[0][0] - n_inputs
end_train = np.where(x <= 100)[0][-1] - n_inputs
start_test = np.where(x >= 100)[0][0] - n_inputs
end_test = np.where(x <= 150)[0][-1] - n_inputs

X_train, X_test = X[start_train:end_train], X[start_test:end_test]
Y_train, Y_test = Y[start_train:end_train], Y[start_test:end_test]

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 = 500
```

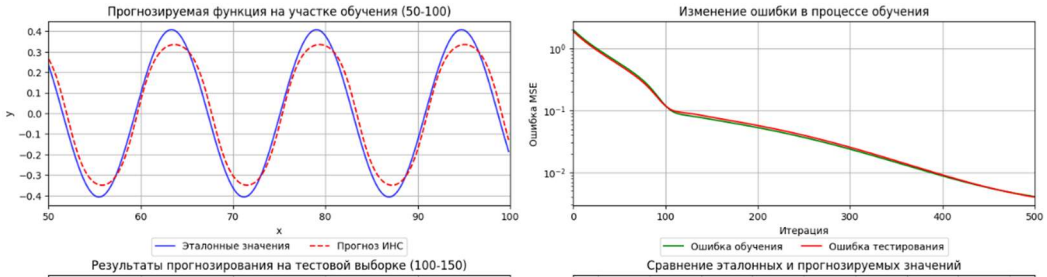
```
for epoch in range(epochs):
    model.train()
    optimizer.zero_grad()
    outputs = model(X_train)
    train_loss = criterion(outputs, Y_train)
    train_loss.backward()
    optimizer.step()

    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 % 100 == 0:
        print(
            f'Epoch [{epoch}/{epochs}], Train Loss: {train_loss.item():.6f}, Test Loss: {test_loss.item():.6f}')

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=(20, 12))

plt.subplot(2, 2, 1)
x_train_plot = x[start_train + n_inputs:end_train + n_inputs]
```

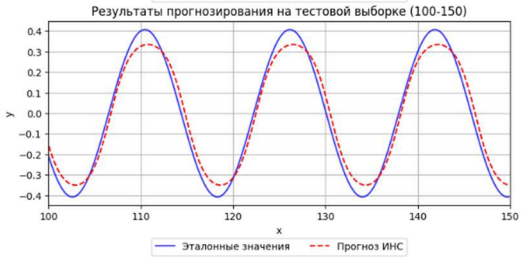
```
plt.plot(x_train_plot, Y_train_actual, 'b-',
        label='Эталонные значения', linewidth=1.5, alpha=0.8)
plt.plot(x_train_plot, train_predictions_actual,
        'r--', label='Прогноз ИНС', linewidth=1.5)
plt.title('Прогнозируемая функция на участке обучения (50-100)', fontsize=12)
plt.xlabel('x')
plt.ylabel('y')
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15), ncol=2)
plt.grid(True)
plt.xlim(50, 100)

plt.subplot(2, 2, 2)
plt.plot(train_losses, 'g-', label='Ошибка обучения', linewidth=1.5)
plt.plot(test_losses, 'r-', label='Ошибка тестирования', linewidth=1.5)
plt.title('Изменение ошибки в процессе обучения', fontsize=12)
plt.xlabel('Итерация')
plt.ylabel('Ошибка MSE')
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15), ncol=2)
plt.grid(True)
plt.yscale('log')
plt.xlim(0, 500)

plt.subplot(2, 2, 3)
x_test_plot = x[start_test + n_inputs:end_test + n_inputs]
plt.plot(x_test_plot, Y_test_actual, 'b-',
        label='Эталонные значения', linewidth=1.5, alpha=0.8)
plt.plot(x_test_plot, test_predictions_actual,
        'r--', label='Прогноз ИНС', linewidth=1.5)
plt.title('Результаты прогнозирования на тестовой выборке (100-150)', fontsize=12)
plt.xlabel('x')
plt.ylabel('y')
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15), ncol=2)
plt.grid(True)
plt.xlim(100, 150)

plt.subplot(2, 2, 4)
```

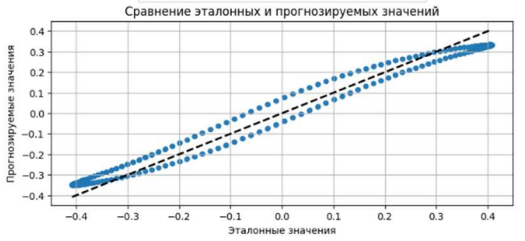
```
plt.scatter(Y_test_actual, test_predictions_actual, alpha=0.6, s=20)
y_min = min(Y_test_actual.min(), test_predictions_actual.min())
y_max = max(Y_test_actual.max(), test_predictions_actual.max())
plt.plot([y_min, y_max], [y_min, y_max], 'k--', lw=2)
plt.title('Сравнение эталонных и прогнозируемых значений', fontsize=12)
plt.xlabel('Эталонные значения')
plt.ylabel('Прогнозируемые значения')
plt.grid(True)

plt.tight_layout(pad=3.0)
plt.show()

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

plt.subplot(1, 2, 1)
plt.plot(train_losses[:500], 'g-', label='Ошибка обучения', linewidth=1.5)
plt.plot(test_losses[:500], 'r-', label='Ошибка тестирования', linewidth=1.5)
plt.title('Детальный просмотр ошибок', fontsize=12)
plt.xlabel('Итерация')
plt.ylabel('Ошибка MSE')
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15), ncol=2)
plt.grid(True)
plt.yscale('log')

plt.subplot(1, 2, 2)
plt.plot(train_losses[400:], 'g-', label='Ошибка обучения', linewidth=1.5)
plt.plot(test_losses[400:], 'r-', label='Ошибка тестирования', linewidth=1.5)
plt.title('Ошибки на последних 100 итерациях', fontsize=12)
plt.xlabel('Итерация')
plt.ylabel('Ошибка MSE')
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15), ncol=2)
plt.grid(True)
plt.yscale('log')

plt.tight_layout(pad=3.0)
plt.show()
```

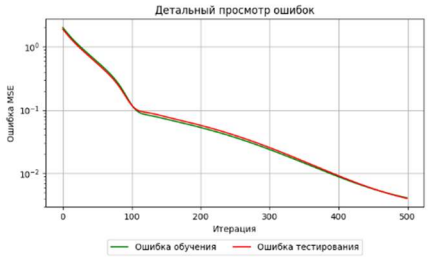
```
train_results = pd.DataFrame({
    'Эталонное значение': Y_train_actual.flatten(),
    'Полученное значение': train_predictions_actual.flatten(),
    'Отклонение': (Y_train_actual.flatten() - train_predictions_actual.flatten())
})

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}")
```

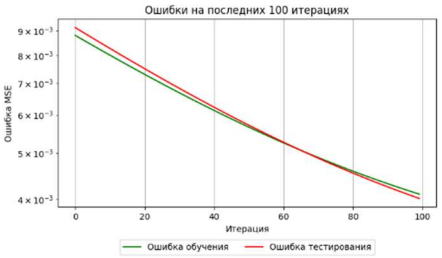
Вывод после запуска программы:











Консольный вывод после запуска программы:

```
Epoch [0/500], Train Loss: 2.015318, Test Loss: 1.895294
Epoch [100/500], Train Loss: 0.119846, Test Loss: 0.118819
Epoch [200/500], Train Loss: 0.052953, Test Loss: 0.057293
Epoch [300/500], Train Loss: 0.023769, Test Loss: 0.025474
Epoch [400/500], Train Loss: 0.008802, Test Loss: 0.009139

============================================================
РЕЗУЛЬТАТЫ ОБУЧЕНИЯ (первые 10 строк):
============================================================
 Эталонное значение  Полученное значение  Отклонение
           0.231253             0.263899   -0.032646
           0.217626             0.255667   -0.038041
           0.203652             0.246840   -0.043188
           0.189351             0.237398   -0.048047
           0.174747             0.227326   -0.052579
           0.159864             0.216611   -0.056747
           0.144724             0.205244   -0.060520
           0.129353             0.193222   -0.063869
           0.113775             0.180547   -0.066771
           0.098015             0.167226   -0.069211

============================================================
РЕЗУЛЬТАТЫ ПРОГНОЗИРОВАНИЯ (первые 10 строк):
============================================================
 Эталонное значение  Полученное значение  Отклонение
          -0.214154            -0.160396   -0.053758
          -0.227870            -0.174932   -0.052937
          -0.241221            -0.188868   -0.052353
          -0.254186            -0.202180   -0.052006
          -0.266744            -0.214854   -0.051890
          -0.278875            -0.226882   -0.051993
          -0.290560            -0.238260   -0.052300
          -0.301780            -0.248990   -0.052790
          -0.312517            -0.259080   -0.053437
          -0.322753            -0.268539   -0.054214

============================================================
СТАТИСТИКА ОШИБОК:
============================================================
Средняя абсолютная ошибка обучения: 0.048588
Средняя абсолютная ошибка тестирования: 0.048298
Максимальная ошибка обучения: 0.075422
Максимальная ошибка тестирования: 0.075407
```

Вывод: Выполнил моделирование прогнозирующей нелинейной ИНС.