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
Савинец М.Д.
Проверил:
Крощенко А.А.

Брест 2025

Цель: Выполнить моделирование прогнозирующей нелинейной ИНС.

Вариант 11

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

```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
import pandas as pd

# Параметры ряда
b = 0.5
c = 0.05
d = 0.5

n_inputs = 8
n_hidden = 3

# Генерация временного ряда
def generate_series(a, N=2000):
    i = np.arange(N)
    y = a * np.cos(b * i) + c * np.sin(d * i)
    return y

# Формирование выборки
def create_dataset(series, look_back=8):
    X, Y = [], []
    for i in range(look_back, len(series)):
        X.append(series[i - look_back:i])
        Y.append(series[i])
```

```
    return np.array(X, dtype=np.float32), np.array(Y, dtype=np.float32)

# Модель MLP
class MLP(nn.Module):
    def __init__(self, input_size, hidden_size):
        super(MLP, self).__init__()
        self.hidden = nn.Linear(input_size, hidden_size)
        self.output = nn.Linear(hidden_size, 1)
        self.sigmoid = nn.Sigmoid()

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

# Подбор параметра a
a_values = np.arange(0.1, 0.51, 0.05)
best_a = None
min_test_mse = float('inf')
results = []

for a in a_values:
    y_full = generate_series(a, N=2000)
    X, Y = create_dataset(y_full, look_back=n_inputs)
    split = int(0.8 * len(X))
    X_train, X_test = X[:split], X[split:]
    Y_train, Y_test = Y[:split], Y[split:]

    X_train_t = torch.tensor(X_train)
    Y_train_t = torch.tensor(Y_train).unsqueeze(1)
    X_test_t = torch.tensor(X_test)
    Y_test_t = torch.tensor(Y_test).unsqueeze(1)

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

    losses = []
    for epoch in range(1500):
        model.train()
        optimizer.zero_grad()
        pred = model(X_train_t)
        loss = criterion(pred, Y_train_t)
        loss.backward()
        optimizer.step()
        losses.append(loss.item())

    model.eval()
    with torch.no_grad():
        pred_test = model(X_test_t).numpy().flatten()

    test_mse = float(np.mean((Y_test - pred_test) ** 2))
    results.append({'a': round(a, 2), 'test_mse': test_mse})

    if test_mse < min_test_mse:
        min_test_mse = test_mse
        best_a = round(a, 2)
        best_model = model
        best_losses = losses
        best_split_data = (X_train, Y_train, X_test, Y_test, pred_test)

print(f"Оптимальное a = {best_a} (Test MSE = {min_test_mse:.8f})")

# Результаты для best_a
a = best_a
```

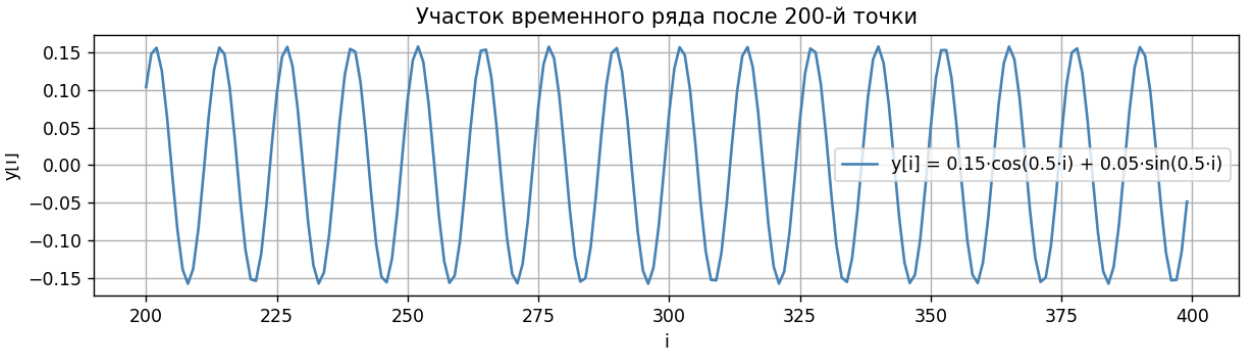
```
X_train, Y_train, X_test, Y_test, pred_test = best_split_data

X_train_t = torch.tensor(X_train)
Y_train_t = torch.tensor(Y_train).unsqueeze(1)
X_test_t = torch.tensor(X_test)
Y_test_t = torch.tensor(Y_test).unsqueeze(1)

model = best_model
with torch.no_grad():
    pred_train = model(X_train_t).numpy().flatten()

# График временного ряда после 200-й точки
y_full = generate_series(a, N=2000)
y_plot = y_full[200:400]
plt.figure(figsize=(10, 3))
plt.plot(np.arange(200, 400), y_plot,
         label=f'y[i] = {a}·cos({b}·i) + {c}·sin({d}·i)',
         color='steelblue')
plt.title('Участок временного ряда после 200-й точки')
plt.xlabel('i')
plt.ylabel('y[i]')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()

# График ошибки обучения
plt.figure(figsize=(8, 4))
plt.plot(best_losses, color='darkorange')
plt.title(f'Ошибка (MSE) при обучении (a = {best_a})')
plt.xlabel('Эпоха')
plt.ylabel('MSE')
plt.yscale('log')
plt.grid(True)
plt.tight_layout()
plt.show()

# Таблица обучения
train_df = pd.DataFrame({
    'Эталонное значение': Y_train[:10],
    'Полученное значение': pred_train[:10],
    'Отклонение': Y_train[:10] - pred_train[:10]
})
print("\n=== РЕЗУЛЬТАТЫ ОБУЧЕНИЯ (первые 10) ===")
print(train_df.round(6).to_string(index=False))

# Таблица прогнозирования
test_df = pd.DataFrame({
    'Эталонное значение': Y_test[:10],
    'Полученное значение': pred_test[:10],
    'Отклонение': Y_test[:10] - pred_test[:10]
})
print("\n=== РЕЗУЛЬТАТЫ ПРОГНОЗИРОВАНИЯ (первые 10) ===")
print(test_df.round(6).to_string(index=False))

# Метрики
train_mse = np.mean((Y_train - pred_train) ** 2)
test_mse = np.mean((Y_test - pred_test) ** 2)
train_mae = np.mean(np.abs(Y_train - pred_train))
test_mae = np.mean(np.abs(Y_test - pred_test))

print(f"\nОценка при a = {best_a}:")
print(f"Train → MSE: {train_mse:.8f}, MAE: {train_mae:.8f}")
print(f"Test  → MSE: {test_mse:.8f}, MAE: {test_mae:.8f}")
```

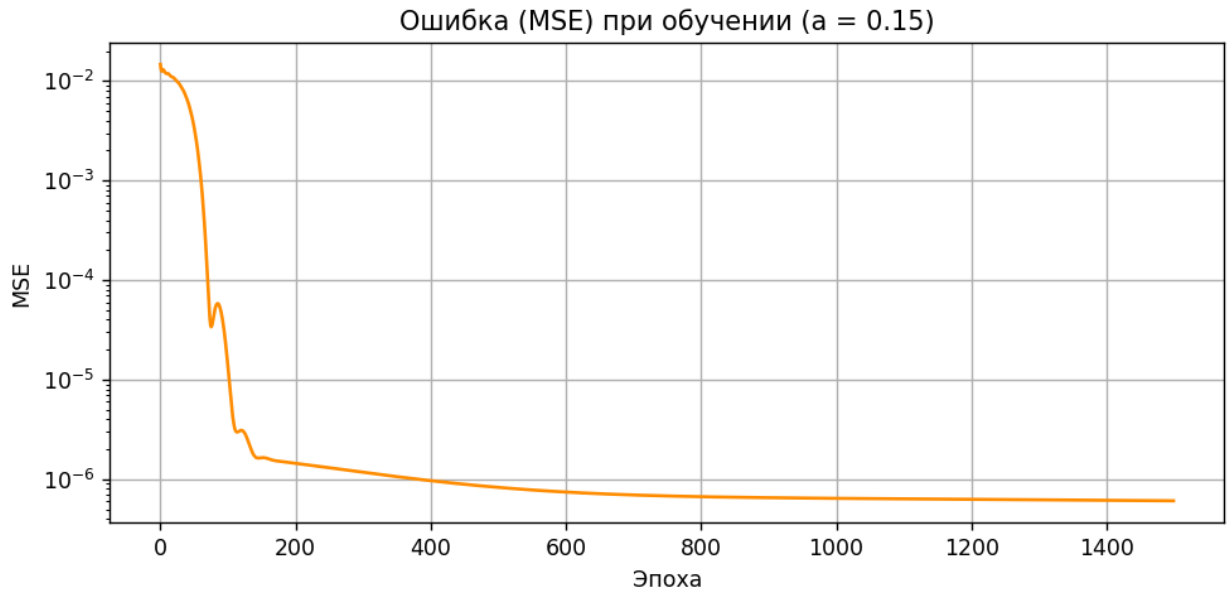
```
# Сравнение эталона и прогноза
plt.figure(figsize=(10, 4))
plt.plot(Y_test[:100], label='Эталон (тест)', color='blue')
plt.plot(pred_test[:100], label='Прогноз (тест)', color='red', linestyle='--
')
plt.title('Сравнение эталона и прогноза (первые 100 точек теста)')
plt.xlabel('Номер точки в тесте')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

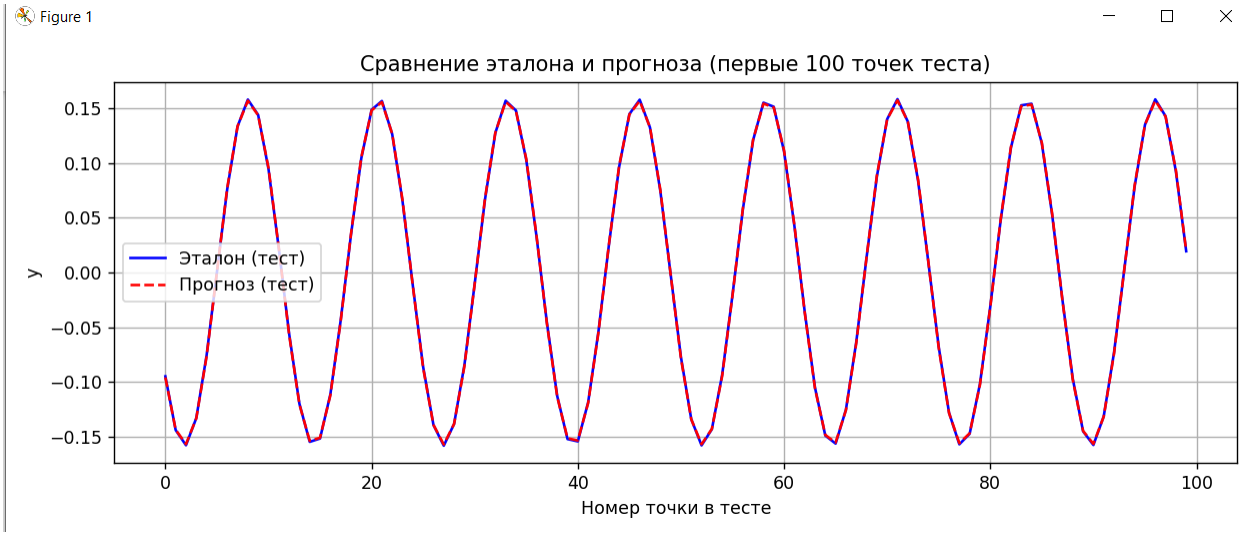
Результаты:

```
Оптимальное a = 0.4 (Test MSE = 0.00000046)

=== РЕЗУЛЬТАТЫ ОБУЧЕНИЯ (первые 10) ===
 Эталонное значение  Полученное значение  Отклонение
          -0.299298           -0.299587    0.000289
          -0.133195           -0.132750   -0.000445
           0.065519            0.064399    0.001119
           0.248191            0.249053   -0.000862
           0.370097            0.370064    0.000034
           0.401391            0.401022    0.000369
           0.334410            0.335174   -0.000764
           0.185554            0.184652    0.000902
          -0.008732           -0.008035   -0.000697
          -0.200880           -0.201390    0.000510

=== РЕЗУЛЬТАТЫ ПРОГНОЗИРОВАНИЯ (первые 10) ===
 Эталонное значение  Полученное значение  Отклонение
          -0.300260           -0.301281    0.001021
          -0.392453           -0.391446   -0.001007
          -0.388560           -0.388568    0.000008
          -0.289534           -0.289691    0.000157
          -0.119620           -0.119302   -0.000318
           0.079581            0.078512    0.001069
           0.259298            0.260249   -0.000951
           0.375530            0.375348    0.000182
           0.399819            0.399582    0.000236
           0.326218            0.326896   -0.000678
```

```
Оценка при a = 0.4:
Train → MSE: 0.00000045, MAE: 0.00059598
Test  → MSE: 0.00000046, MAE: 0.00059769
```







Вывод: На практике выполнили моделирование прогнозирующей нелинейной ИНС.