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

Лабораторная работа №6
По дисциплине: «ОМО»

Выполнил:
Студент 3-го курса
Группы АС-66
Езепчук А.С.
Проверил:
Крощенко А.А.

Брест 2025

Цель работы: По вариантам предыдущей лабораторной работы реализовать предложенный вариант рекуррентной нейронной сети.

Ход работы

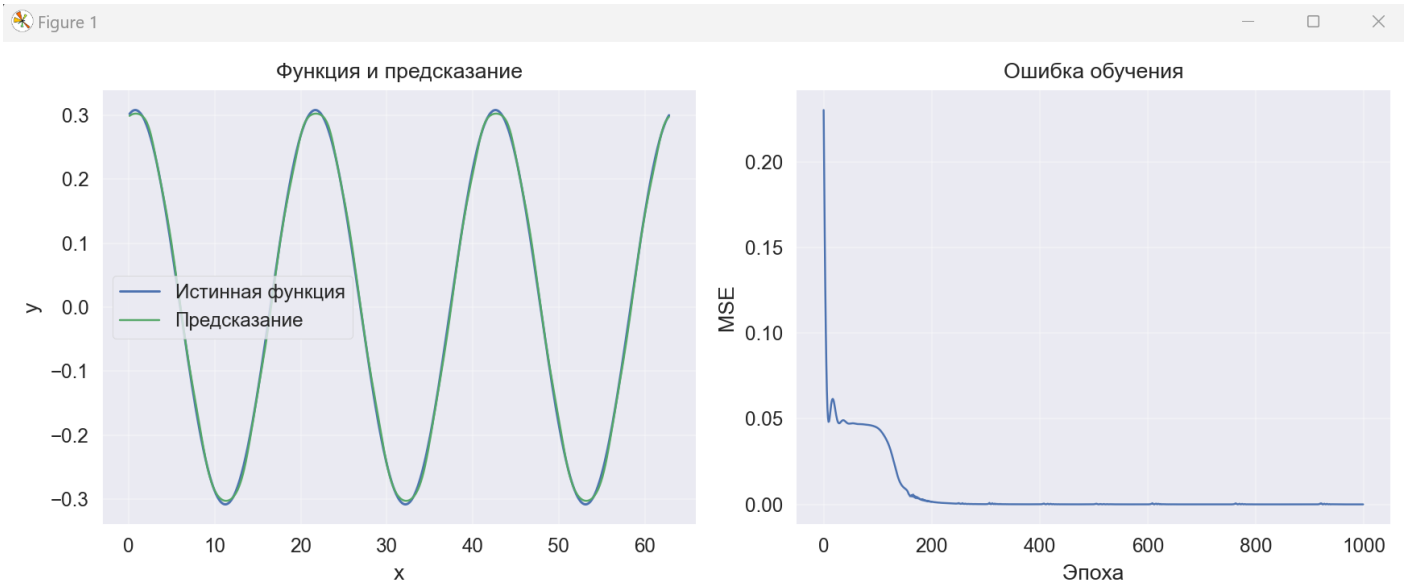
Вариант 3

a	b	c	d	Кол-во входов ИНС	Кол-во НЭ в скрытом слое	Тип РНС
0.3	0.3	0.07	0.3	10	4	Мультирекуррентная

```python
import torch
import torch.nn as nn
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0, 20*np.pi, 0.01)
y = 0.3 * np.cos(0.3 * x) + 0.07 * np.sin(0.3 * x)

window = 10
X, Y = [], []

for i in range(len(x) - window):
    X.append(y[i:i+window])
    Y.append(y[i+window])

X = np.array(X)
Y = np.array(Y).reshape(-1, 1)

X_tensor = torch.tensor(X, dtype=torch.float32)
Y_tensor = torch.tensor(Y, dtype=torch.float32)

class MultiRecurrentNet(nn.Module):
    def __init__(self, input_size, hidden_size):
        super().__init__()
        self.rnn1 = nn.RNN(input_size, hidden_size, batch_first=True)
        self.rnn2 = nn.RNN(hidden_size, hidden_size, batch_first=True)
        self.fc = nn.Linear(hidden_size, 1)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = x.unsqueeze(2)
        out1, _ = self.rnn1(x)
        out1 = self.sigmoid(out1)

        out2, _ = self.rnn2(out1)
        out2 = self.sigmoid(out2)

        out = self.fc(out2[:, -1, :])
        return out

model = MultiRecurrentNet(input_size=1, hidden_size=4)
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
```

```python
num_epochs = 1000
losses = []

print("Начало обучения...\n")

for epoch in range(num_epochs):
    output = model(X_tensor)
    loss = criterion(output, Y_tensor)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    losses.append(loss.item())
    if (epoch + 1) % 50 == 0:
        print(f"Эпоха {epoch+1:3d}/{num_epochs} | Loss: {loss.item():.6f}")

print("\nОбучение завершено!\n")

with torch.no_grad():
    prediction = model(X_tensor).numpy()

mae = np.mean(np.abs(Y - prediction))
rmse = np.sqrt(np.mean((Y - prediction)**2))

print(f"Средняя абсолютная ошибка (MAE): {mae:.6f}")
print(f"Корень из среднеквадратичной ошибки (RMSE): {rmse:.6f}")

plt.style.use('seaborn-v0_8')

fig, axes = plt.subplots(1, 2, figsize=(10, 4))

axes[0].plot(x[window:], Y, label='Истинная функция', linewidth=1.2)
axes[0].plot(x[window:], prediction, label='Предсказание', linewidth=1)
axes[0].set_title("Функция и предсказание", fontsize=11)
axes[0].set_xlabel("x")
axes[0].set_ylabel("y")
axes[0].legend(frameon=True)
axes[0].grid(alpha=0.3)

axes[1].plot(losses, linewidth=1)
axes[1].set_title("Ошибка обучения", fontsize=11)
axes[1].set_xlabel("Эпоха")
axes[1].set_ylabel("MSE")
axes[1].grid(alpha=0.3)

plt.tight_layout()
plt.show()
```

Графики при выполнении:



Эпохи 5 ЛР:
Epoch 1 | MSE=0.12569170
Epoch 250 | MSE=0.02648347
Epoch 500 | MSE=0.00404351
Epoch 750 | MSE=0.00023834
Epoch 1000 | MSE=0.00002726

Эпохи 6 ЛР:
Эпоха 50/1000 | Loss: 0.047365
Эпоха 100/1000 | Loss: 0.046872
Эпоха 150/1000 | Loss: 0.009750
Эпоха 200/1000 | Loss: 0.001491
Эпоха 250/1000 | Loss: 0.000514
Эпоха 300/1000 | Loss: 0.000250
Эпоха 350/1000 | Loss: 0.000156
Эпоха 400/1000 | Loss: 0.000110
Эпоха 450/1000 | Loss: 0.000083
Эпоха 500/1000 | Loss: 0.000066
Эпоха 550/1000 | Loss: 0.000053
Эпоха 600/1000 | Loss: 0.000043
Эпоха 650/1000 | Loss: 0.000035
Эпоха 700/1000 | Loss: 0.000029
Эпоха 750/1000 | Loss: 0.000027
Эпоха 800/1000 | Loss: 0.000022
Эпоха 850/1000 | Loss: 0.000025
Эпоха 900/1000 | Loss: 0.000020
Эпоха 950/1000 | Loss: 0.000018
Эпоха 1000/1000 | Loss: 0.000017

Значения из ЛР6:

Средняя абсолютная ошибка (MAE): 0.003520
Корень из среднеквадратичной ошибки (RMSE): 0.004129

Вывод: я реализовал предложенный вариант рекуррентной нейронной сети.