Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ
Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №6

Выполнил
В.Д.Головкина,
студент группы АС66
Проверил
А. А. Крощенко,
доц. кафедры ИИТ,
« __ » _____2025 г.

Брест 2025

Цель работы: изучить и выполнить моделирование прогнозирующей рекуррентной нейронной сети.
Задание:
1. По вариантам предыдущей лабораторной работы реализовать предложенный вариант рекуррентной нейронной сети. Сравнить полученные результаты с ЛР 5. В качестве функций активации для скрытого слоя использовать сигмоидную функцию, для выходного - линейную.

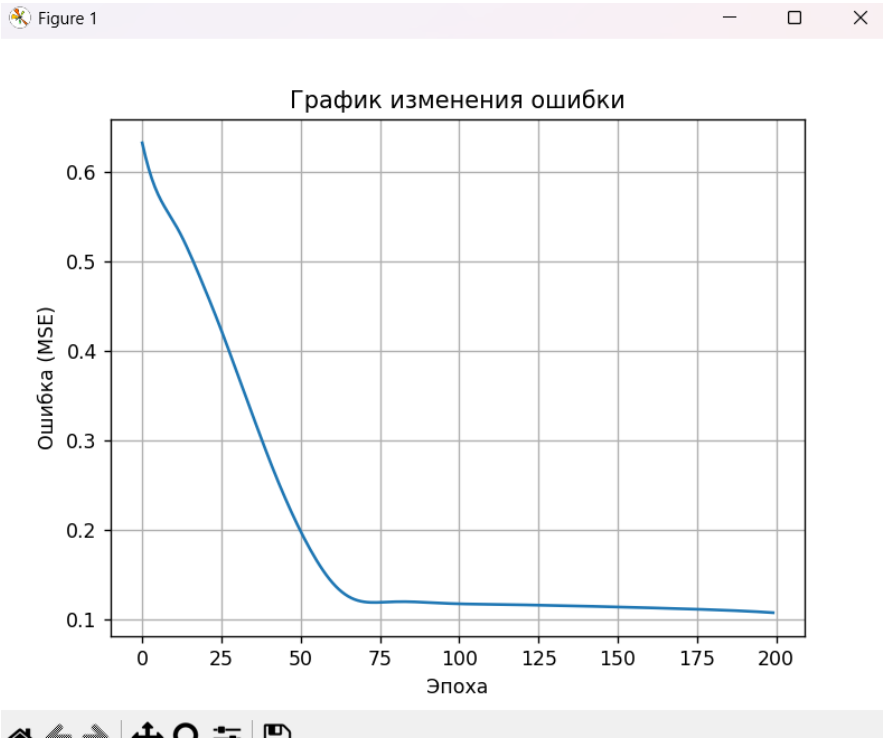
Варианты заданий приведены в следующей таблице:

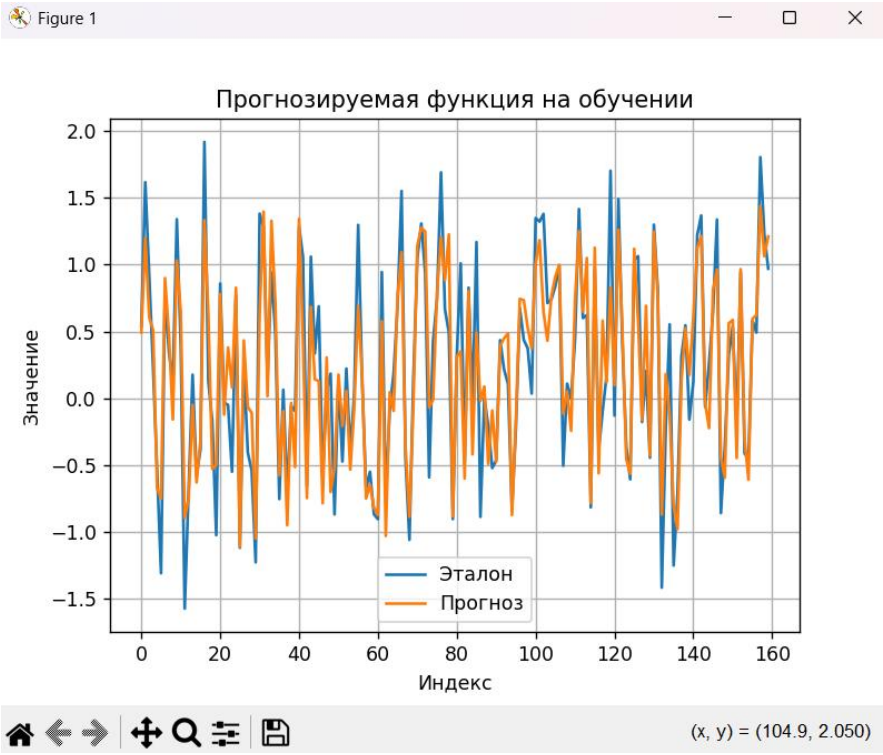
№	a	b	c	d	Кол-во входов ИНС	Кол-во НЭ в скрытом слое	Тип РНС
1	0.1	0.1	0.05	0.1	6	2	Элмана

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

a, b, c, d = 0.1, 0.1, 0.05, 0.1
n_inputs = 6
n_hidden = 2
n_samples = 200

def target_function(X):
    x1, x2, x3, x4, x5, x6 = X[:, 0], X[:, 1], X[:, 2], X[:, 3], X[:, 4], X[:, 5]
    return a * np.cos(b * x1) + c * np.sin(d * x2) + x3 * x4 - x5 + 0.5 * x6

np.random.seed(42)
X = np.random.uniform(-1, 1, (n_samples, n_inputs))
y = target_function(X)

split = int(0.8 * n_samples)
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]

X_train_tensor = torch.tensor(X_train, dtype=torch.float32).unsqueeze(1)  # [samples,
seq_len=1, features] тк динамич
y_train_tensor = torch.tensor(y_train, dtype=torch.float32).view(-1, 1)
X_test_tensor = torch.tensor(X_test, dtype=torch.float32).unsqueeze(1)
y_test_tensor = torch.tensor(y_test, dtype=torch.float32).view(-1, 1)

class ElmanRNN(nn.Module):
    def __init__(self):
        super(ElmanRNN, self).__init__()
        self.rnn = nn.RNN(input_size=n_inputs, hidden_size=n_hidden, batch_first=True,
nonlinearity='tanh')
        self.output = nn.Linear(n_hidden, 1)
```

```
    def forward(self, x):
        out, _ = self.rnn(x)
        return self.output(out[:, -1, :])  # берём последний выход

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

n_epochs = 200
losses = []

for epoch in range(n_epochs):
    model.train()
    y_pred = model(X_train_tensor)
    loss = criterion(y_pred, y_train_tensor)
    losses.append(loss.item())
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

plt.figure()
plt.plot(losses)
plt.xlabel("Эпоха")
plt.ylabel("Ошибка (MSE)")
plt.title("График изменения ошибки")
plt.grid(True)
plt.show()

model.eval()
with torch.no_grad():
    y_train_pred = model(X_train_tensor).numpy().flatten()
    y_test_pred = model(X_test_tensor).numpy().flatten()

train_df = pd.DataFrame({
    "Эталонное значение": y_train,
    "Полученное значение": y_train_pred,
    "Отклонение": y_train_pred - y_train
})
train_df.to_csv("train_results.csv", index=False)

test_df = pd.DataFrame({
    "Эталонное значение": y_test,
    "Полученное значение": y_test_pred,
    "Отклонение": y_test_pred - y_test
})
test_df.to_csv("test_results.csv", index=False)

plt.figure()
plt.plot(y_train, label="Эталон")
plt.plot(y_train_pred, label="Прогноз")
plt.title("Прогнозируемая функция на обучении")
plt.xlabel("Индекс")
plt.ylabel("Значение")
```

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

print("\n Результаты обучения (первые 5 строк):")
print(train_df.head())
print("\n Результаты прогнозирования (первые 5 строк):")
print(test_df.head())
print("\n Ошибка на последней эпохе:", losses[-1])
```





```
 Результаты обучения (первые 5 строк):
   Эталонное значение  Полученное значение  Отклонение
0            0.539979             0.493258   -0.046722
1            1.616166             1.199876   -0.416290
2            0.916107             0.604530   -0.311577
3            0.218682             0.509882    0.291200
4           -0.552632            -0.671061   -0.118428

 Результаты прогнозирования (первые 5 строк):
   Эталонное значение  Полученное значение  Отклонение
0            0.491733             0.411590   -0.080143
1            0.201002             0.199272   -0.001730
2           -0.095869            -0.584598   -0.488730
3            0.283288             0.396431    0.113143
4            0.877034             0.945185    0.068151

 Ошибка на последней эпохе: 0.10785198211669922
PS D:\y--ба\3 kurs\омо\1>
```

Вывод: я изучила и выполнила моделирование прогнозирующей рекуррентной нейронной сети.