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
Лысюк Р.А.
Проверил:
Крощенко А.А.

Брест 2025

Ход работы

Вариант 4

Задание:

1. По вариантам предыдущей лабораторной работы реализовать предложенный вариант рекуррентной нейронной сети. Сравнить полученные результаты с ЛР 5.

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
2	0.2	0.2	0.06	0.2	8	3	Джордана
3	0.3	0.3	0.07	0.3	10	4	Мультирекуррентная
4	0.4	0.4	0.08	0.4	6	2	Элмана
5	0.1	0.5	0.09	0.5	8	3	Джордана
6	0.2	0.6	0.05	0.6	10	4	Мультирекуррентная
7	0.3	0.1	0.06	0.1	6	2	Элмана
8	0.4	0.2	0.07	0.2	8	3	Джордана
9	0.1	0.3	0.08	0.3	10	4	Мультирекуррентная
10	0.2	0.4	0.09	0.4	6	2	Элмана
11	0.3	0.5	0.05	0.5	8	3	Джордана

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

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

print("="*70)
print("ЛАБОРАТОРНАЯ РАБОТА 6: РЕКУРРЕНТНАЯ СЕТЬ ЭЛМАНА")
print(f"Вариант №4: a={a}, b={b}, c={c}, d={d}")
print("="*70)

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

t = np.linspace(-100, 300, 4000)
series = target_function(t)

def create_dataset(data, n_inputs):
    X, y = [], []
```

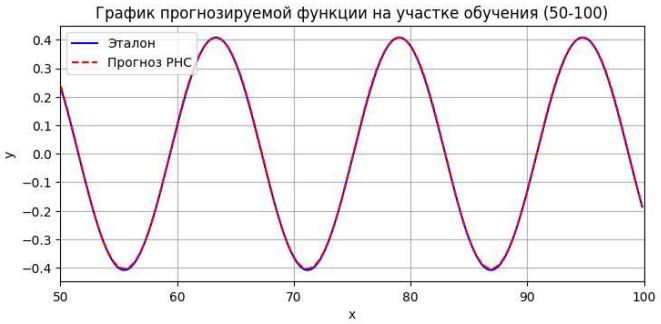
```
    for i in range(len(data) - n_inputs):
        X.append(data[i:i + n_inputs])
        y.append(data[i + n_inputs])
    return np.array(X), np.array(y)

X, y = create_dataset(series, n_inputs)

start_train_idx = np.where(t >= 50)[0][0] - n_inputs
end_train_idx = np.where(t <= 100)[0][-1] - n_inputs
start_test_idx = np.where(t >= 100)[0][0] - n_inputs
end_test_idx = np.where(t <= 150)[0][-1] - n_inputs

X_train = X[start_train_idx:end_train_idx]
y_train = y[start_train_idx:end_train_idx]
X_test = X[start_test_idx:end_test_idx]
y_test = y[start_test_idx:end_test_idx]

class ElmanRNN:
    def __init__(self, input_size, hidden_size, output_size):
        self.input_size = input_size
        self.hidden_size = hidden_size
        self.output_size = output_size

        self.W_ih = np.random.randn(hidden_size, input_size) * 0.1
        self.W_hh = np.random.randn(hidden_size, hidden_size) * 0.1
        self.W_ho = np.random.randn(output_size, hidden_size) * 0.1

        self.b_h = np.zeros((hidden_size, 1))
        self.b_o = np.zeros((output_size, 1))

        self.context = np.zeros((hidden_size, 1))

    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))

    def sigmoid_derivative(self, x):
        return x * (1 - x)

    def forward(self, inputs):
        self.inputs = inputs.reshape(-1, 1)

        hidden_input = (self.W_ih @ self.inputs +
                        self.W_hh @ self.context +
                        self.b_h)
        self.hidden = self.sigmoid(hidden_input)

        self.context = self.hidden.copy()
```

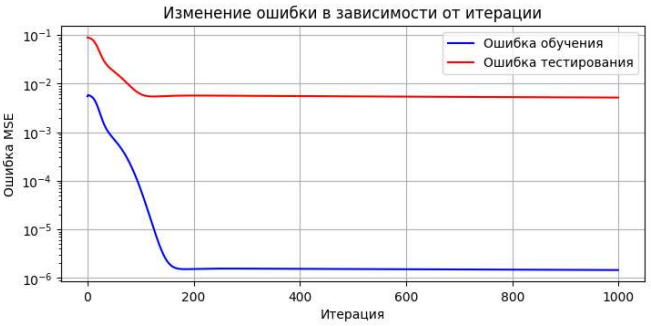
```
        output_input = self.W_ho @ self.hidden + self.b_o
        self.output = output_input

        return self.output

    def backward(self, target, learning_rate=0.01):
        output_error = self.output - target

        dW_ho = output_error @ self.hidden.T
        db_o = output_error

        hidden_error = (self.W_ho.T @ output_error) * self.sigmoid_derivative(self.hidden)

        dW_ih = hidden_error @ self.inputs.T
        dW_hh = hidden_error @ self.context.T
        db_h = hidden_error

        self.W_ho -= learning_rate * dW_ho
        self.W_ih -= learning_rate * dW_ih
        self.W_hh -= learning_rate * dW_hh
        self.b_o -= learning_rate * db_o
        self.b_h -= learning_rate * db_h

        return np.mean(output_error**2)

rnn = ElmanRNN(n_inputs, n_hidden, 1)

epochs = 1000
train_errors = []
test_errors = []

print("\nНачало обучения...")
for epoch in range(epochs):
    epoch_train_error = 0
    epoch_test_error = 0

    for i in range(len(X_train)):
        rnn.context = np.zeros((n_hidden, 1))
        output = rnn.forward(X_train[i])
        error = rnn.backward(np.array([[y_train[i]]]), 0.1)
        epoch_train_error += error

    rnn.context = np.zeros((n_hidden, 1))
    for i in range(len(X_test)):
        output = rnn.forward(X_test[i])
        error = np.mean((output - y_test[i])**2)
        epoch_test_error += error
```

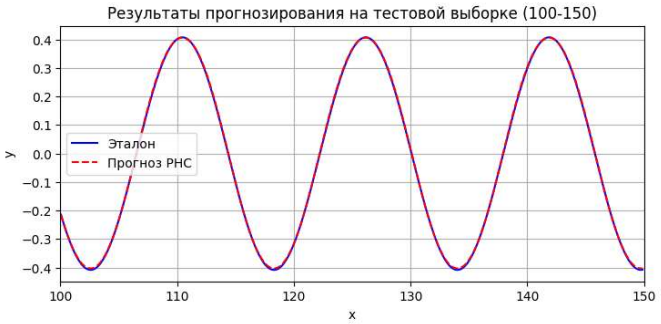
```
    avg_train_error = epoch_train_error / len(X_train)
    avg_test_error = epoch_test_error / len(X_test)
    train_errors.append(avg_train_error)
    test_errors.append(avg_test_error)

    if epoch % 100 == 0:
        print(f"Эпоха {epoch}: train error = {avg_train_error:.6f}, test error = {avg_test_error:.6f}")

print("Обучение завершено!")

t_train_plot = t[start_train_idx + n_inputs:end_train_idx + n_inputs]
t_test_plot = t[start_test_idx + n_inputs:end_test_idx + n_inputs]

train_predictions = []
for i in range(len(X_train)):
    rnn.context = np.zeros((n_hidden, 1))
    pred = rnn.forward(X_train[i])[0, 0]
    train_predictions.append(pred)

test_predictions = []
for i in range(len(X_test)):
    rnn.context = np.zeros((n_hidden, 1))
    pred = rnn.forward(X_test[i])[0, 0]
    test_predictions.append(pred)

train_predictions = np.array(train_predictions)
test_predictions = np.array(test_predictions)

plt.figure(figsize=(16, 10))

plt.subplot(2, 2, 1)
plt.plot(t_train_plot, y_train, 'b-', label='Эталон', linewidth=1.5)
plt.plot(t_train_plot, train_predictions, 'r--', label='Прогноз РНС', linewidth=1.5)
plt.title('График прогнозируемой функции на участке обучения (50-100)')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.xlim(50, 100)

plt.subplot(2, 2, 2)
plt.plot(train_errors, 'b-', label='Ошибка обучения', linewidth=1.5)
plt.plot(test_errors, 'r-', label='Ошибка тестирования', linewidth=1.5)
plt.title('Изменение ошибки в зависимости от итерации')
plt.xlabel('Итерация')
plt.ylabel('Ошибка MSE')
plt.legend()
plt.grid(True)
plt.yscale('log')
```

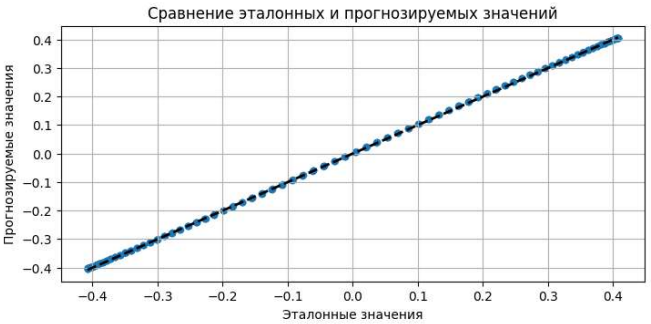
```
plt.subplot(2, 2, 3)
plt.plot(t_test_plot, y_test, 'b-', label='Эталон', linewidth=1.5)
plt.plot(t_test_plot, test_predictions, 'r--', label='Прогноз РНС', linewidth=1.5)
plt.title('Результаты прогнозирования на тестовой выборке (100-150)')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.xlim(100, 150)

plt.subplot(2, 2, 4)
plt.scatter(y_test, test_predictions, alpha=0.6, s=20)
y_min = min(y_test.min(), test_predictions.min())
y_max = max(y_test.max(), test_predictions.max())
plt.plot([y_min, y_max], [y_min, y_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(pad=3.0)
plt.show()

train_results = []
for i in range(min(20, len(X_train))):
    rnn.context = np.zeros((n_hidden, 1))
    pred = rnn.forward(X_train[i])[0, 0]
    target = y_train[i]
    deviation = target - pred
    train_results.append([target, pred, deviation])

test_results = []
for i in range(min(20, len(X_test))):
    rnn.context = np.zeros((n_hidden, 1))
    pred = rnn.forward(X_test[i])[0, 0]
    target = y_test[i]
    deviation = target - pred
    test_results.append([target, pred, deviation])

train_df = pd.DataFrame(train_results, columns=['Эталонное значение', 'Полученное значение',
'Отклонение'])
test_df = pd.DataFrame(test_results, columns=['Эталонное значение', 'Полученное значение',
'Отклонение'])

print("\n" + "="*70)
print("РЕЗУЛЬТАТЫ ОБУЧЕНИЯ (первые 20 значений)")
print("="*70)
print(train_df.round(6).to_string(index=False))
```

```
print("\n" + "="*70)
print("РЕЗУЛЬТАТЫ ПРОГНОЗИРОВАНИЯ (первые 20 значений)")
print("="*70)
print(test_df.round(6).to_string(index=False))

train_mae = np.mean(np.abs(train_df['Отклонение']))
train_mse = np.mean(train_df['Отклонение']**2)
train_rmse = np.sqrt(train_mse)
train_max_err = np.max(np.abs(train_df['Отклонение']))

test_mae = np.mean(np.abs(test_df['Отклонение']))
test_mse = np.mean(test_df['Отклонение']**2)
test_rmse = np.sqrt(test_mse)
test_max_err = np.max(np.abs(test_df['Отклонение']))

print("\n" + "="*70)
print("СТАТИСТИКА ОШИБОК")
print("="*70)
print(f"Обучение:")
print(f"  MAE:  {train_mae:.6f}")
print(f"  MSE:  {train_mse:.6f}")
print(f"  RMSE: {train_rmse:.6f}")
print(f"  Max:  {train_max_err:.6f}")

print(f"\nТестирование:")
print(f"  MAE:  {test_mae:.6f}")
print(f"  MSE:  {test_mse:.6f}")
print(f"  RMSE: {test_rmse:.6f}")
print(f"  Max:  {test_max_err:.6f}")
```










```
================================================================================
ЛАБОРАТОРНАЯ РАБОТА 6: РЕКУРРЕНТНАЯ СЕТЬ ЭЛМАНА
Вариант №4: a=0.4, b=0.4, c=0.08, d=0.4
================================================================================

Начало обучения...
Эпоха 0: train error = 0.005514, test error = 0.089197
Эпоха 100: train error = 0.000067, test error = 0.006107
Эпоха 200: train error = 0.000002, test error = 0.005692
Эпоха 300: train error = 0.000002, test error = 0.005626
Эпоха 400: train error = 0.000002, test error = 0.005545
Эпоха 500: train error = 0.000002, test error = 0.005471
Эпоха 600: train error = 0.000002, test error = 0.005403
Эпоха 700: train error = 0.000001, test error = 0.005340
Эпоха 800: train error = 0.000001, test error = 0.005282
Эпоха 900: train error = 0.000001, test error = 0.005227
Обучение завершено!
```