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

Лабораторная работа №6
По дисциплине «Основы машинного обучения»
Тема: «Рекуррентные нейронные сети»

Выполнил:
Студент 3 курса
Группы АС-65
Ракецкий П. П.
Проверил:
Крощенко А. А.

Брест 2025

Цель: научится реализовывать рекуррентную нейронную сеть и сравнить полученные значения с результатами ЛР5.

Вариант 4

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

В качестве функций активации для скрытого слоя использовать сигмоидную функцию, для выходного - линейную.

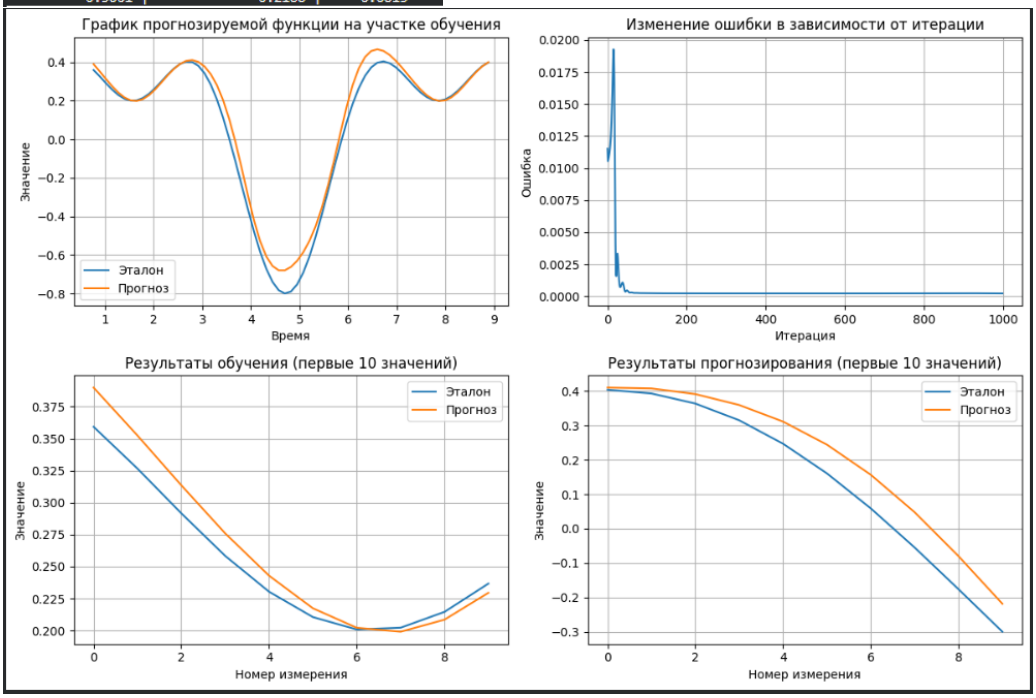
```
import numpy as np
import matplotlib.pyplot as plt

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 linear(self, x):
        return x

    def forward(self, inputs):
        self.inputs = inputs

        hidden_input = (self.W_ih @ 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 = self.linear(output_input)
```

```
        return self.output

    def backward(self, target, learning_rate=0.1):
        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_deriva-
tive(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)

def target_function(t):
    return 0.5 * np.sin(t) + 0.3 * np.cos(2*t)

t = np.linspace(0, 4*np.pi, 100)
series = target_function(t)

X = []
y = []
for i in range(len(series) - 6):
    X.append(series[i:i+6])
    y.append(series[i+6])

X = np.array(X)
y = np.array(y)

split_idx = int(0.7 * len(X))
X_train, X_test = X[:split_idx], X[split_idx:]
y_train, y_test = y[:split_idx], y[split_idx:]

rnn = ElmanRNN(6, 2, 1)

epochs = 1000
errors = []

for epoch in range(epochs):
    epoch_error = 0
    for i in range(len(X_train)):
        rnn.context = np.zeros((2, 1))

        inputs = X_train[i].reshape(-1, 1)
        target = np.array([[y_train[i]]])

        output = rnn.forward(inputs)
        error = rnn.backward(target, 0.4)
        epoch_error += error

    avg_error = epoch_error / len(X_train)
    errors.append(avg_error)

print("Результаты обучения:")
print("Эталонное значение | Полученное значение | Отклонение")
print("-" * 55)
```

```
train_results = []
for i in range(10):
    rnn.context = np.zeros((2, 1))
    inputs = X_train[i].reshape(-1, 1)
    prediction = rnn.forward(inputs)[0, 0]
    target = y_train[i]
    deviation = abs(target - prediction)
    train_results.append((target, prediction, deviation))
    print(f"{target:16.4f} | {prediction:19.4f} | {deviation:10.4f}")

print("\nРезультаты прогнозирования:")
print("Эталонное значение | Полученное значение | Отклонение")
print("-" * 55)

test_results = []
for i in range(10):
    rnn.context = np.zeros((2, 1))
    inputs = X_test[i].reshape(-1, 1)
    prediction = rnn.forward(inputs)[0, 0]
    target = y_test[i]
    deviation = abs(target - prediction)
    test_results.append((target, prediction, deviation))
    print(f"{target:16.4f} | {prediction:19.4f} | {deviation:10.4f}")

plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1)
t_train = t[6:6+len(y_train)]
plt.plot(t_train, y_train, label='Эталон')
train_preds = []
for i in range(len(X_train)):
    rnn.context = np.zeros((2, 1))
    inputs = X_train[i].reshape(-1, 1)
    pred = rnn.forward(inputs)[0, 0]
    train_preds.append(pred)
plt.plot(t_train, train_preds, label='Прогноз')
plt.title('График прогнозируемой функции на участке обучения')
plt.xlabel('Время')
plt.ylabel('Значение')
plt.legend()
plt.grid(True)

plt.subplot(2, 2, 2)
plt.plot(errors)
plt.title('Изменение ошибки в зависимости от итерации')
plt.xlabel('Итерация')
plt.ylabel('Ошибка')
plt.grid(True)

plt.subplot(2, 2, 3)
train_targets = [r[0] for r in train_results]
train_preds = [r[1] for r in train_results]
plt.plot(train_targets, label='Эталон')
plt.plot(train_preds, label='Прогноз')
plt.title('Результаты обучения (первые 10 значений)')
plt.xlabel('Номер измерения')
plt.ylabel('Значение')
plt.legend()
plt.grid(True)

plt.subplot(2, 2, 4)
test_targets = [r[0] for r in test_results]
test_preds = [r[1] for r in test_results]
plt.plot(test_targets, label='Эталон')
plt.plot(test_preds, label='Прогноз')
```

```
plt.title('Результаты прогнозирования (первые 10 значений)')
plt.xlabel('Номер измерения')
plt.ylabel('Значение')
plt.legend()
plt.grid(True)

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

```
Результаты обучения:
Эталонное значение | Полученное значение | Отклонение
-------------------------------------------------------
            0.3593 |              0.3900 |     0.0306
            0.3266 |              0.3525 |     0.0259
            0.2916 |              0.3135 |     0.0219
            0.2584 |              0.2760 |     0.0176
            0.2303 |              0.2429 |     0.0125
            0.2105 |              0.2174 |     0.0069
            0.2008 |              0.2021 |     0.0013
            0.2022 |              0.1990 |     0.0032
            0.2146 |              0.2085 |     0.0061
            0.2367 |              0.2295 |     0.0072

Результаты прогнозирования:
Эталонное значение | Полученное значение | Отклонение
-------------------------------------------------------
            0.4040 |              0.4105 |     0.0064
            0.3932 |              0.4080 |     0.0147
            0.3640 |              0.3915 |     0.0274
            0.3153 |              0.3598 |     0.0445
            0.2471 |              0.3114 |     0.0644
            0.1607 |              0.2444 |     0.0836
            0.0588 |              0.1567 |     0.0979
           -0.0551 |              0.0476 |     0.1027
           -0.1765 |             -0.0802 |     0.0963
           -0.3001 |             -0.2188 |     0.0813
```



Вывод: научился реализовывать рекуррентную нейронную сеть и сравнил полученные значения с результатами ЛР5.