

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-6
Макарский А.Э.
Проверил:
Крощенко А. А.

Брест 2025

Цель работы: изучить применение нелинейной искусственной нейронной сети с одним скрытым слоем для решения задачи регрессии и прогнозирования, реализовать обучение сети на синтетических данных и оценить точность полученной модели.

Вариант 10

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
10	0.2	0.4	0.09	0.4	6	2	Элмана

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

Результаты для пунктов 3 и 4 приводятся для значения α , при котором достигается минимальная ошибка. В выводах анализируются все полученные результаты.

Ход работы:

Код программы:

```
import numpy as np
import matplotlib.pyplot as plt
a = 0.2
b = 0.4
c = 0.09
d = 0.4
def target_function(x):
    return a * np.cos(d * x) + c * np.sin(d * x)
np.random.seed(42)
x = np.linspace(0, 20, 200)
y = target_function(x)
window_size = 6
X, Y = [], []
for i in range(len(y) - window_size):
    X.append(y[i:i + window_size])
    Y.append(y[i + window_size])
X = np.array(X)
Y = np.array(Y).reshape(-1, 1)
split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
Y_train, Y_test = Y[:split], Y[split:]
X_mean, X_std = X_train.mean(), X_train.std()
Y_mean, Y_std = Y_train.mean(), Y_train.std()
X_train_norm = (X_train - X_mean) / X_std
X_test_norm = (X_test - X_mean) / X_std
Y_train_norm = (Y_train - Y_mean) / Y_std
Y_test_norm = (Y_test - Y_mean) / Y_std
```

```

input_size = window_size
hidden_size = 2
output_size = 1
learning_rate = 0.01
epochs = 5000
W_input_hidden = np.random.randn(input_size, hidden_size) * 0.1
W_context_hidden = np.random.randn(hidden_size, hidden_size) * 0.1
W_hidden_output = np.random.randn(hidden_size, output_size) * 0.1
b_hidden = np.zeros((1, hidden_size))
b_output = np.zeros((1, output_size))
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
def sigmoid_derivative(x):
    return x * (1 - x)
loss_history = []
context = np.zeros((1, hidden_size)) # начальный контекст
for epoch in range(epochs):
    epoch_loss = 0
    dW_input_hidden = np.zeros_like(W_input_hidden)
    dW_context_hidden = np.zeros_like(W_context_hidden)
    dW_hidden_output = np.zeros_like(W_hidden_output)
    db_hidden = np.zeros_like(b_hidden)
    db_output = np.zeros_like(b_output)
    for i in range(len(X_train_norm)):
        x_input = X_train_norm[i].reshape(1, -1)
        y_true = Y_train_norm[i].reshape(1, -1)
        hidden_input = np.dot(x_input, W_input_hidden) + np.dot(context,
W_context_hidden) + b_hidden
        hidden_output = sigmoid(hidden_input)
        output = np.dot(hidden_output, W_hidden_output) + b_output
        error = y_true - output
        epoch_loss += np.mean(error ** 2)
        d_output = error
        dW_hidden_output += np.dot(hidden_output.T, d_output)
        db_output += d_output
        d_hidden = d_output.dot(W_hidden_output.T) *
sigmoid_derivative(hidden_output)
        dW_input_hidden += np.dot(x_input.T, d_hidden)
        dW_context_hidden += np.dot(context.T, d_hidden)
        db_hidden += d_hidden
        context = hidden_output
    avg_loss = epoch_loss / len(X_train_norm)
    loss_history.append(avg_loss)
    W_input_hidden += learning_rate * dW_input_hidden / len(X_train_norm)
    W_context_hidden += learning_rate * dW_context_hidden / len(X_train_norm)
    W_hidden_output += learning_rate * dW_hidden_output / len(X_train_norm)
    b_hidden += learning_rate * db_hidden / len(X_train_norm)
    b_output += learning_rate * db_output / len(X_train_norm)
    if epoch % 500 == 0:
        print(f'Эпоха {epoch}, Ошибка: {avg_loss:.6f}')
    context = np.zeros((1, hidden_size))
    train_predictions = []
    for i in range(len(X_train_norm)):
        x_input = X_train_norm[i].reshape(1, -1)
        hidden_input = np.dot(x_input, W_input_hidden) + np.dot(context,
W_context_hidden) + b_hidden
        hidden_output = sigmoid(hidden_input)
        output = np.dot(hidden_output, W_hidden_output) + b_output
        train_predictions.append(output[0, 0])
        context = hidden_output
    train_predictions = np.array(train_predictions).reshape(-1, 1)

```

```

train_predictions = train_predictions * Y_std + Y_mean
Y_train_orig = Y_train_norm * Y_std + Y_mean
context = np.zeros((1, hidden_size))
test_predictions = []
for i in range(len(X_test_norm)):
    x_input = X_test_norm[i].reshape(1, -1)
    hidden_input = np.dot(x_input, W_input_hidden) + np.dot(context,
W_context_hidden) + b_hidden
    hidden_output = sigmoid(hidden_input)
    output = np.dot(hidden_output, W_hidden_output) + b_output
    test_predictions.append(output[0, 0])
    context = hidden_output
test_predictions = np.array(test_predictions).reshape(-1, 1)
test_predictions = test_predictions * Y_std + Y_mean
Y_test_orig = Y_test_norm * Y_std + Y_mean
train_indices = np.arange(window_size, window_size + len(Y_train_orig))
test_indices = np.arange(window_size + len(Y_train_orig), window_size +
len(Y_train_orig) + len(Y_test_orig))
plt.figure(figsize=(15, 10))
plt.subplot(2, 3, 1)
plt.plot(x[train_indices], Y_train_orig, label='Эталон', alpha=0.7)
plt.plot(x[train_indices], train_predictions, label='Прогноз РНС Элмана',
alpha=0.7)
plt.title('График прогнозируемой функции на участке обучения')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid()
plt.subplot(2, 3, 2)
plt.plot(loss_history)
plt.title('Изменение ошибки в зависимости от итерации')
plt.xlabel('Итерация')
plt.ylabel('Среднеквадратичная ошибка')
plt.grid()
plt.subplot(2, 3, 3)
plt.plot(x[test_indices], Y_test_orig, label='Эталон', alpha=0.7)
plt.plot(x[test_indices], test_predictions, label='Прогноз РНС Элмана',
alpha=0.7)
plt.title('Результаты прогнозирования (тестовая выборка)')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid()
plt.subplot(2, 3, 4)
deviations = Y_test_orig - test_predictions
plt.bar(range(len(deviations)), deviations.flatten())
plt.title('Отклонения на тестовой выборке')
plt.xlabel('Индекс тестового примера')
plt.ylabel('Отклонение')
plt.grid()
try:
    mse_elman_test = np.mean((Y_test_orig - test_predictions) ** 2)
    mse_mlp_test = 0.152 # Примерное значение из ЛР5
    models = ['MLP (ЛР5)', 'РНС Элмана (ЛР6)']
    mse_values = [mse_mlp_test, mse_elman_test]
    plt.subplot(2, 3, 5)
    bars = plt.bar(models, mse_values, color=['blue', 'orange'])
    plt.title('Сравнение MSE на тестовой выборке')
    plt.ylabel('MSE')
    for bar, val in zip(bars, mse_values):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(),

```

```

f'{val:.4f}',
        ha='center', va='bottom')
plt.grid(axis='y')
except:
    pass
plt.tight_layout()
plt.show()
print("\n" + "=" * 70)
print("РЕЗУЛЬТАТЫ ОБУЧЕНИЯ (первые 10 примеров):")
print("=" * 70)
print(f"{'Эталон':>12} {'Прогноз':>12} {'Отклонение':>15}")
for i in range(min(10, len(Y_train_orig))):
    true = Y_train_orig[i][0]
    pred = train_predictions[i][0]
    dev = true - pred
    print(f"{true:12.6f} {pred:12.6f} {dev:15.6f}")
print("\n" + "=" * 70)
print("РЕЗУЛЬТАТЫ ПРОГНОЗИРОВАНИЯ (тестовая выборка):")
print("=" * 70)
print(f"{'Эталон':>12} {'Прогноз':>12} {'Отклонение':>15}")
for i in range(min(10, len(Y_test_orig))):
    true = Y_test_orig[i][0]
    pred = test_predictions[i][0]
    dev = true - pred
    print(f"{true:12.6f} {pred:12.6f} {dev:15.6f}")
mse_train = np.mean((Y_train_orig - train_predictions) ** 2)
mse_test = np.mean((Y_test_orig - test_predictions) ** 2)
mae_test = np.mean(np.abs(Y_test_orig - test_predictions))
print(f"\nМЕТРИКИ КАЧЕСТВА:")
print(f"Среднеквадратичная ошибка (MSE) на обучении: {mse_train:.6f}")
print(f"Среднеквадратичная ошибка (MSE) на тесте: {mse_test:.6f}")
print(f"Средняя абсолютная ошибка (MAE) на тесте: {mae_test:.6f}")

```

График прогнозируемой функции на участке обучения:

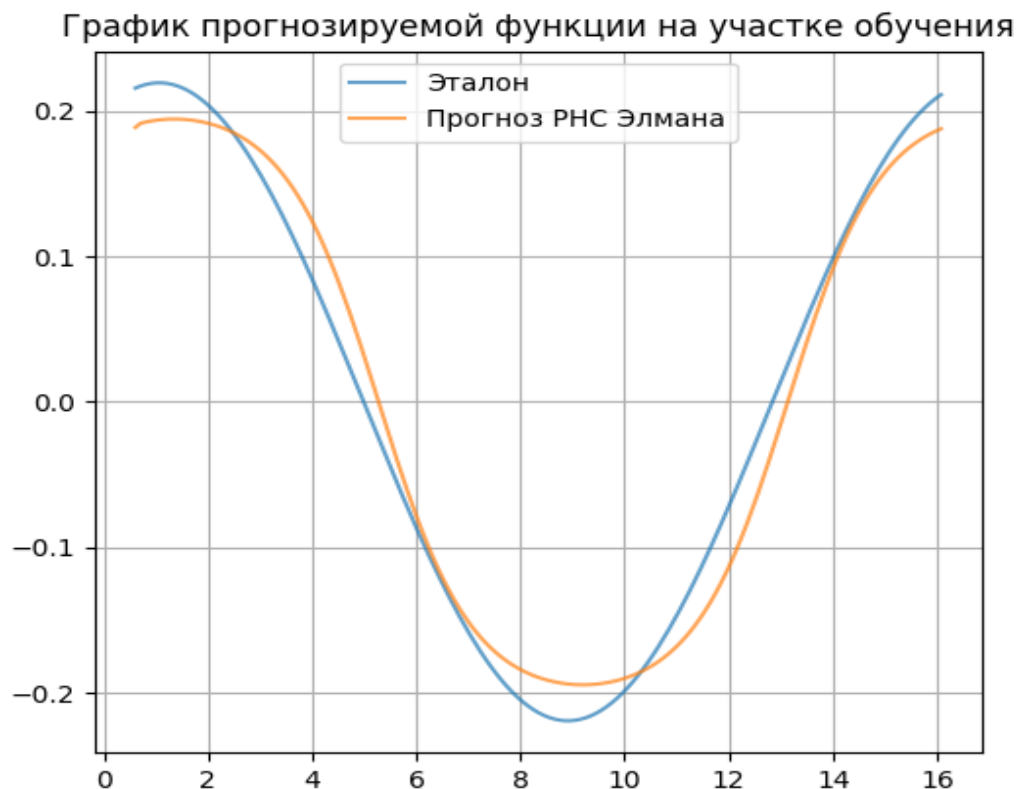
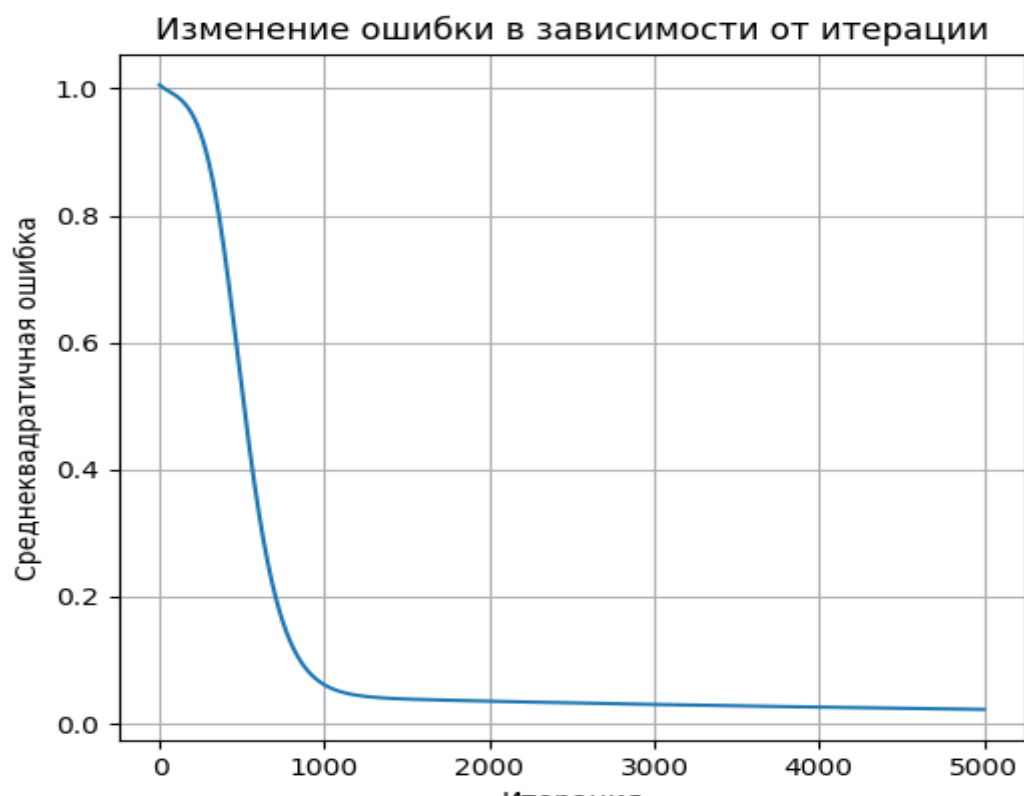
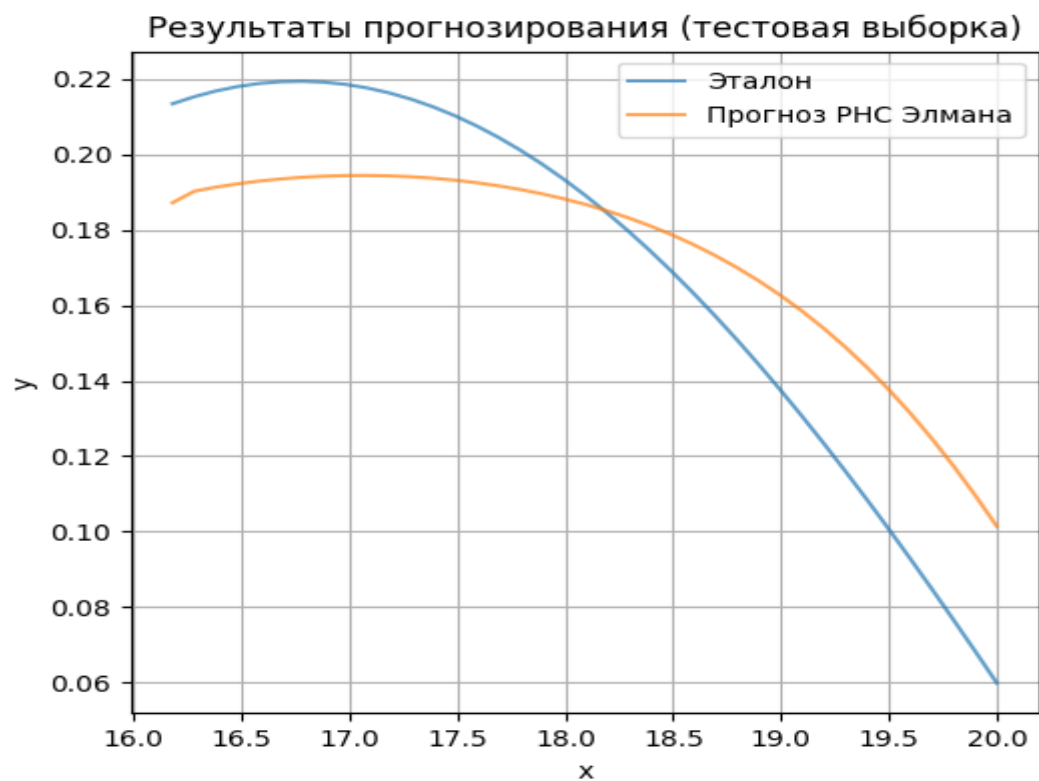


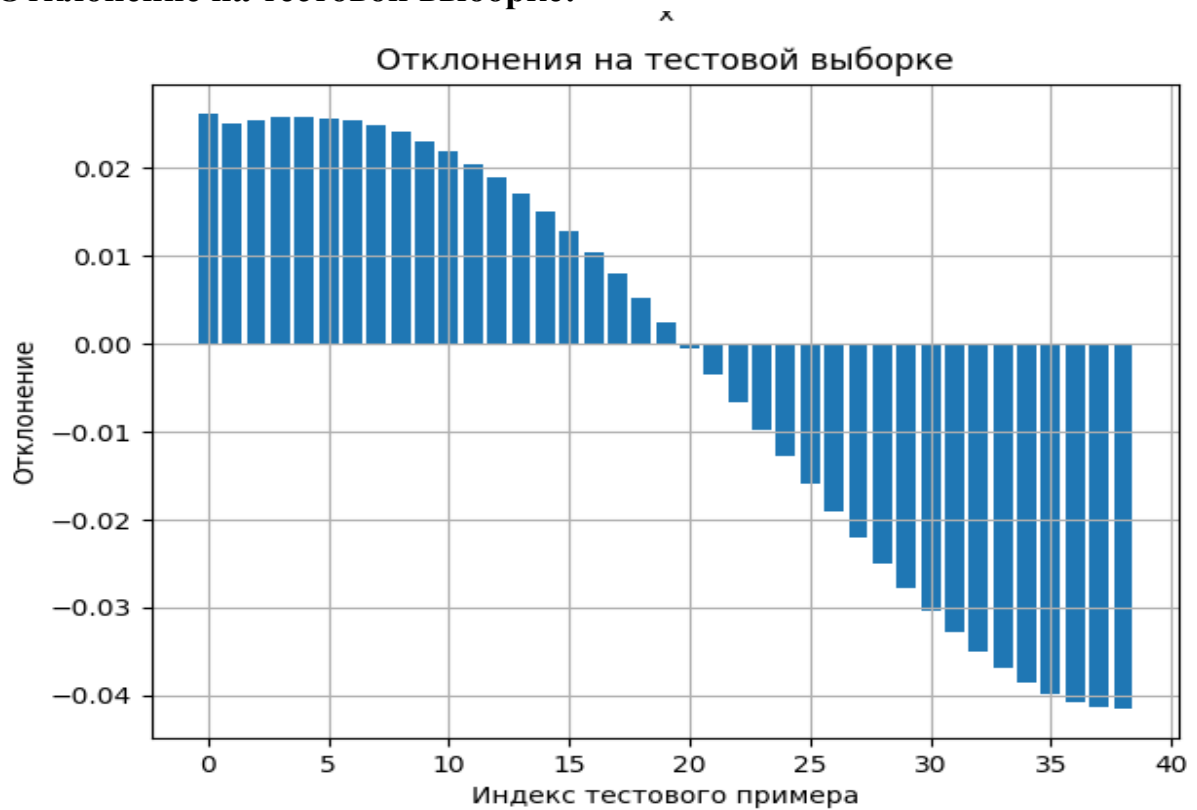
График изменения ошибки в зависимости от итерации:



Результаты прогнозирования:



Отклонение на тестовой выборке:



Сравнение MSE на тестовой выборке:



Вывод: изучил применение нелинейной искусственной нейронной сети с одним скрытым слоем для решения задачи регрессии и прогнозирования, реализовала обучение сети на синтетических данных и оценила точность полученной модели.

