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

Лабораторная работа №5
По дисциплине: «Основы машинного обучения»
Тема: «Нелинейные ИНС в задачах регрессии»

Выполнил:
Студент 3 курса
Группы АС-65
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. Выполнить моделирование прогнозирующей нелинейной ИНС. Для генерации обучающих и тестовых данных использовать функцию

$$y = a\cos(bx) + c\sin(dx)\ .$$

№ варианта	a	b	c	d	Кол-во входов ИНС	Кол-во НЭ в скрытом слое
10	0.2	0.4	0.09	0.4	6	2

Для прогнозирования использовать многослойную ИНС с одним скрытым слоем. 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 = (X_train - X_mean) / X_std
X_test = (X_test - X_mean) / X_std
```

```
Y_train = (Y_train - Y_mean) / Y_std
Y_test = (Y_test - Y_mean) / Y_std
input_size = window_size
hidden_size = 2
output_size = 1
learning_rate = 0.01
epochs = 5000
W1 = np.random.randn(input_size, hidden_size) * 0.1
b1 = np.zeros((1, hidden_size))
W2 = np.random.randn(hidden_size, output_size) * 0.1
b2 = np.zeros((1, output_size))
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
def sigmoid_derivative(x):
    return x * (1 - x)
loss_history = []
for epoch in range(epochs):
    hidden_input = np.dot(X_train, W1) + b1
    hidden_output = sigmoid(hidden_input)
    output = np.dot(hidden_output, W2) + b2
    error = Y_train - output
    loss = np.mean(error ** 2)
    loss_history.append(loss)
    d_output = error
    d_W2 = np.dot(hidden_output.T, d_output)
    d_b2 = np.sum(d_output, axis=0, keepdims=True)
    d_hidden = d_output.dot(W2.T) * sigmoid_derivative(hidden_output)
    d_W1 = np.dot(X_train.T, d_hidden)
    d_b1 = np.sum(d_hidden, axis=0, keepdims=True)
    W2 += learning_rate * d_W2
    b2 += learning_rate * d_b2
    W1 += learning_rate * d_W1
    b1 += learning_rate * d_b1
    if epoch % 500 == 0:
        print(f'Эпоха {epoch}, Ошибка: {loss:.6f}')
hidden = sigmoid(np.dot(X_train, W1) + b1)
train_predictions = np.dot(hidden, W2) + b2
train_predictions = train_predictions * Y_std + Y_mean
Y_train_orig = Y_train * Y_std + Y_mean
hidden_test = sigmoid(np.dot(X_test, W1) + b1)
test_predictions = np.dot(hidden_test, W2) + b2
test_predictions = test_predictions * Y_std + Y_mean
Y_test_orig = Y_test * 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=(12, 8))
plt.subplot(2, 2, 1)
plt.plot(x[train_indices], Y_train_orig, label='Эталон', alpha=0.7)
plt.plot(x[train_indices], train_predictions, label='Прогноз ИНС', alpha=0.7)
plt.title('График прогнозируемой функции на участке обучения')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid()
plt.subplot(2, 2, 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, 2, 3)
plt.plot(x[test_indices], Y_test_orig, label='Эталон', alpha=0.7)
plt.plot(x[test_indices], test_predictions, label='Прогноз ИНС', alpha=0.7)
plt.title('Результаты прогнозирования (тестовая выборка)')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid()
plt.subplot(2, 2, 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()
plt.tight_layout()
plt.show()
print("\n" + "=" * 60)
print("РЕЗУЛЬТАТЫ ОБУЧЕНИЯ (первые 10 примеров):")
print("=" * 60)
print(f"{'Эталон':>10} {'Прогноз':>10} {'Отклонение':>12}")
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:10.5f} {pred:10.5f} {dev:12.5f}")
print("\n" + "=" * 60)
print("РЕЗУЛЬТАТЫ ПРОГНОЗИРОВАНИЯ (тестовая выборка):")
print("=" * 60)
print(f"{'Эталон':>10} {'Прогноз':>10} {'Отклонение':>12}")
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:10.5f} {pred:10.5f} {dev:12.5f}")
mse_train = np.mean((Y_train_orig - train_predictions) ** 2)
mse_test = np.mean((Y_test_orig - test_predictions) ** 2)
print(f"\nСреднеквадратичная ошибка на обучении: {mse_train:.6f}")
print(f"Среднеквадратичная ошибка на тесте: {mse_test:.6f}")
```

График прогнозируемой функции на участке обучения:

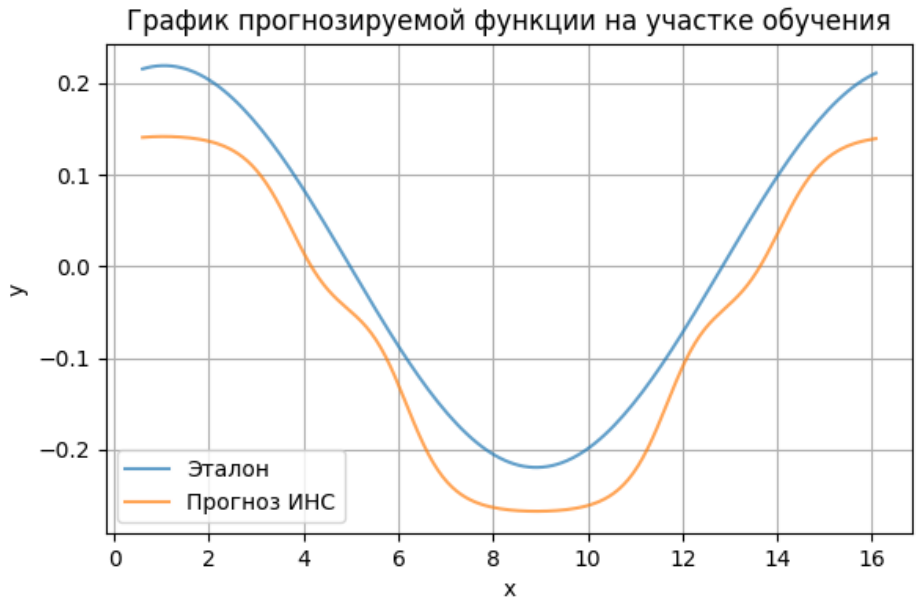
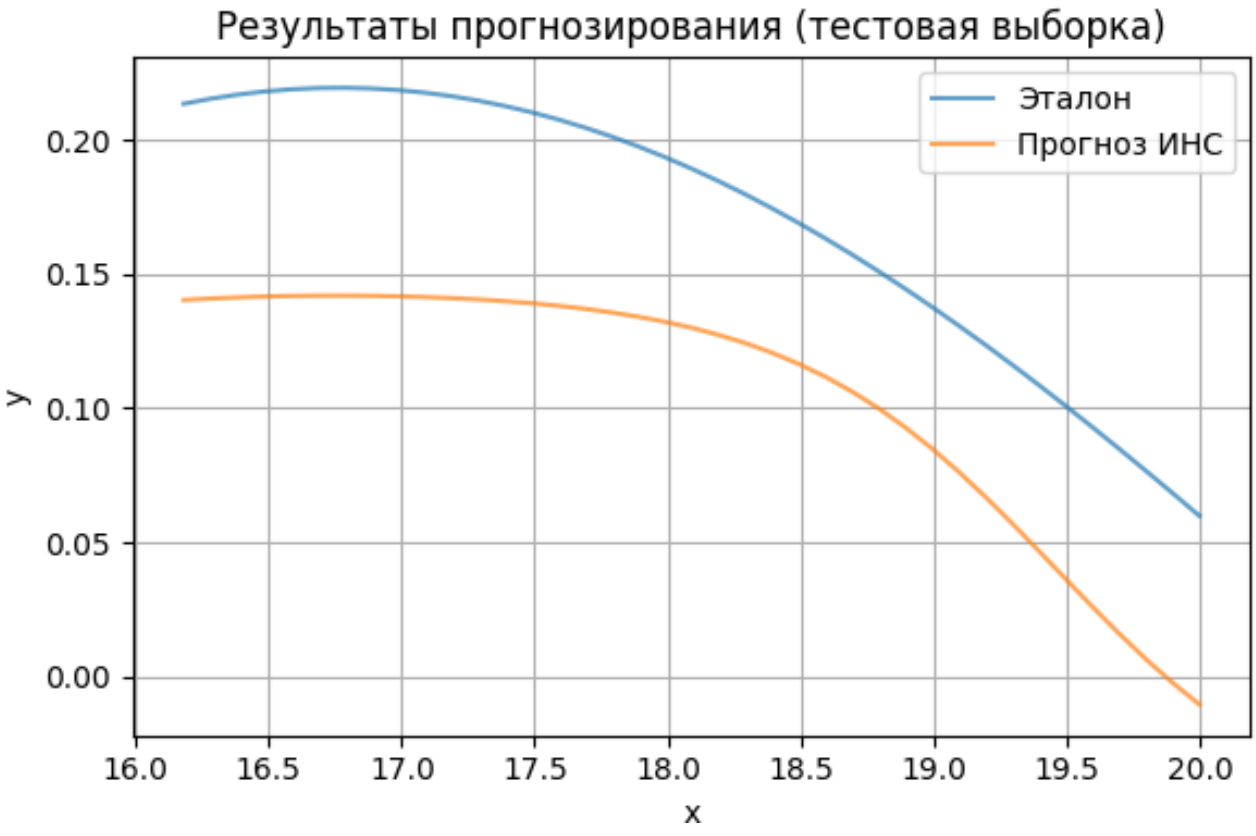
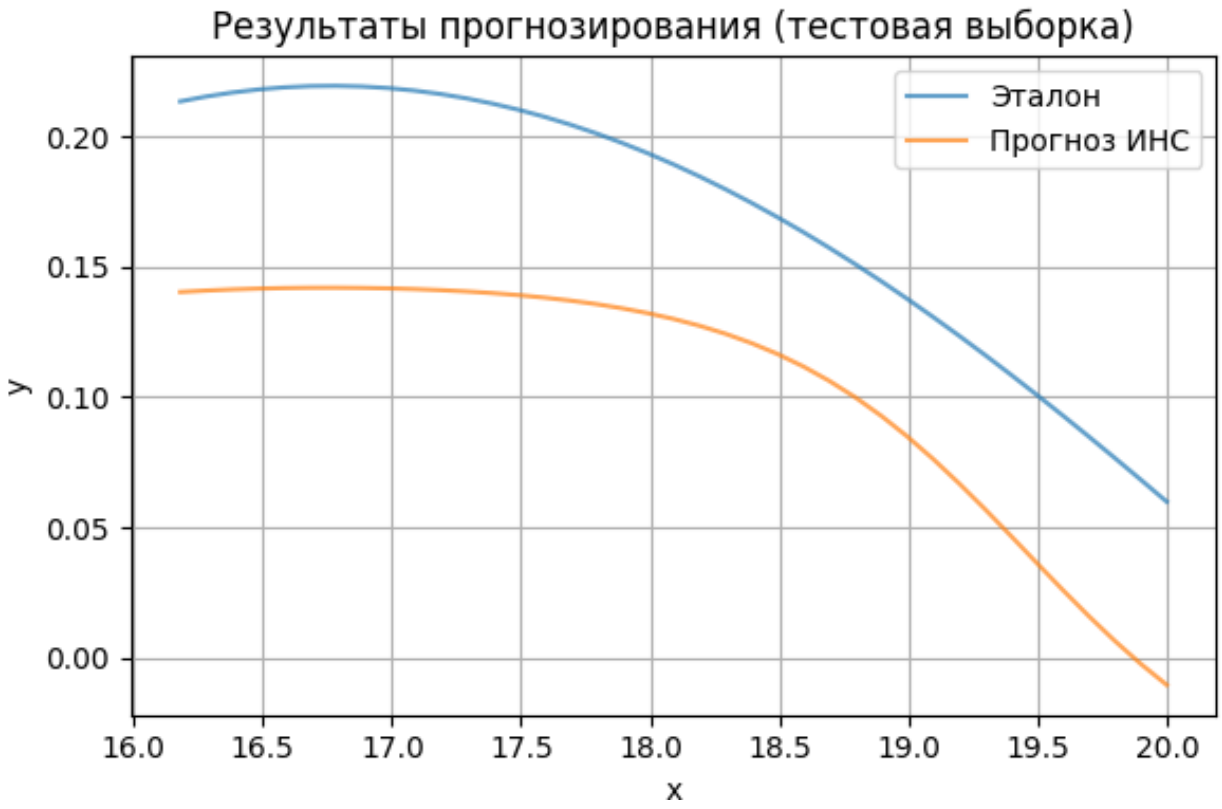


График изменения ошибки в зависимости от итерации:

Результаты прогнозирования:



Отклонения 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.