

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

Лабораторная работа №5  
По дисциплине: «ОМО»  
Тема: "Нелинейные ИНС в задачах регрессии  
„"

Выполнил:  
Студент 3-го курса  
Группы АС-66  
Пекун М.С.  
Проверил:  
Крощенко А.А.

Брест 2025

Цель: исследовать работу нелинейной ИНС в задаче регрессии, обучив модель аппроксимировать заданную нелинейную функцию и оценив качество прогнозирования.

.

## Вариант 8

Задание:

1. Выполнить моделирование прогнозирующей нелинейной ИНС. Для генерации обучающих и тестовых данных использовать функцию

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

8	0.4	0.2	0.07	0.2	8	3	I
---	-----	-----	------	-----	---	---	---

Для прогнозирования использовать многослойную ИНС с одним скрытым слоем. В качестве функций активации для скрытого слоя использовать сигмоидную функцию, для выходного - линейную.

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

# =====
# 1. Параметры варианта
# =====

a = 0.4
b = 0.2
c = 0.07
d = 0.2

INPUT_SIZE = 8
HIDDEN = 3
EPOCHS = 2000
LR = 0.01

# =====
# 2. Генерация данных
# =====

def f(x):
    return a * np.sin(b * x) + c * x**2 + d

X_all = np.linspace(0, 10, 300)
y_all = f(X_all)

X = []
y = []
```

```

for i in range(len(X_all) - INPUT_SIZE):
    X.append(X_all[i:i + INPUT_SIZE])
    y.append(y_all[i + INPUT_SIZE])

X = np.array(X)
y = np.array(y).reshape(-1, 1)

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

X_train_t = torch.tensor(X_train, dtype=torch.float32)
y_train_t = torch.tensor(y_train, dtype=torch.float32)
X_test_t = torch.tensor(X_test, dtype=torch.float32)
y_test_t = torch.tensor(y_test, dtype=torch.float32)

# =====
# 3. Архитектура ИНС
# =====

class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(INPUT_SIZE, HIDDEN)
        self.sigmoid = nn.Sigmoid()
        self.fc2 = nn.Linear(HIDDEN, 1)

    def forward(self, x):
        x = self.sigmoid(self.fc1(x))
        return self.fc2(x)

model = Net()
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=LR)

# =====
# 4. Обучение с ранней остановкой
# =====

loss_train_hist = []
loss_test_hist = []

best_test_loss = float("inf")
stop_epoch = None

for epoch in range(EPOCHS):
    model.train()

    optimizer.zero_grad()
    y_pred_train = model(X_train_t)
    loss_train = criterion(y_pred_train, y_train_t)
    loss_train.backward()
    optimizer.step()

    loss_train_hist.append(loss_train.item())

```

```

# тестовая ошибка
model.eval()
with torch.no_grad():
    y_pred_test = model(X_test_t)
    loss_test = criterion(y_pred_test, y_test_t).item()

loss_test_hist.append(loss_test)

# ===== Ранняя остановка =====
if loss_test < best_test_loss:
    best_test_loss = loss_test
else:
    stop_epoch = epoch
    print(f"\n\033[0mостановка обучения на эпохе {epoch} "
          f"(ошибка тестирования начала расти).")
    break

if stop_epoch is None:
    print("\nОбучение завершено полностью.")

# =====
# 5. Вычисление предсказаний (ПОСЛЕ обучения!)
# =====

model.eval()
with torch.no_grad():
    train_pred = model(X_train_t).numpy()
    test_pred = model(X_test_t).numpy()

# =====
# 6. График 1 – участок обучения
# =====

plt.figure(figsize=(8, 5))
plt.plot(X_train[:, -1], y_train, label="Эталонные значения", linewidth=2)
plt.plot(X_train[:, -1], train_pred, "--", label="Прогноз ИНС", linewidth=2)
plt.title("Прогнозируемая функция на участке обучения")
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True)
plt.legend()
plt.show()

# =====
# 7. График 2 – ошибка обучения/тестирования
# =====

plt.figure(figsize=(8, 5))
plt.plot(loss_train_hist, label="Ошибка обучения", color="green")
plt.plot(loss_test_hist, label="Ошибка тестирования", color="red")
plt.yscale("log")
plt.grid(True)
plt.xlabel("Итерация")
plt.ylabel("Ошибка MSE")
plt.title("Изменение ошибки в процессе обучения")
plt.legend()
plt.show()

```

```

# =====
# 8. График 3 – участок тестовой выборки
# =====

plt.figure(figsize=(8, 5))
plt.plot(X_test[:, -1], y_test, label="Эталонные значения", linewidth=2)
plt.plot(X_test[:, -1], test_pred, "--", label="Прогноз ИНС", linewidth=2)
plt.title("Результаты прогнозирования на тестовой выборке")
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True)
plt.legend()
plt.show()

# =====
# 9. График 4 – сравнение точек
# =====

plt.figure(figsize=(6, 6))
plt.scatter(y_test, test_pred, s=15, alpha=0.8)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], "k--")
plt.title("Сравнение эталонных и прогнозируемых значений")
plt.xlabel("Эталонные значения")
plt.ylabel("Прогнозируемые значения")
plt.grid(True)
plt.show()

# =====
# 10. Текстовый вывод
# =====

print("\nНачало обучения...")

for epoch in range(0, len(loss_train_hist), 200):
    print(f"Epoch {epoch}/{EPOCHS}, "
          f"Train Loss: {loss_train_hist[epoch]:.6f}, "
          f"Test Loss: {loss_test_hist[epoch]:.6f}")

print("Обучение завершено!\n")

print("=" * 60)
print("РЕЗУЛЬТАТЫ ОБУЧЕНИЯ (первые 10 строк):")
print("=" * 60)

train_output = np.hstack([
    y_train[:10],
    train_pred[:10],
    (y_train[:10] - train_pred[:10])
])

print(f"{'Эталонное значение':>20} {'Полученное':>15} {'Отклонение':>15}")
for row in train_output:
    print(f"{row[0]:>20.6f} {row[1]:>15.6f} {row[2]:>15.6f}")

print("\n" + "=" * 60)
print("РЕЗУЛЬТАТЫ ПРОГНОЗИРОВАНИЯ (первые 10 строк):")
print("=" * 60)

```

```

test_output = np.hstack([
    y_test[:10],
    test_pred[:10],
    (y_test[:10] - test_pred[:10])
])

print(f"{'Эталонное значение':>20} {'Полученное':>15} {'Отклонение':>15}")
for row in test_output:
    print(f"{row[0]:>20.6f} {row[1]:>15.6f} {row[2]:>15.6f}")

abs_train = np.abs(y_train - train_pred)
abs_test = np.abs(y_test - test_pred)

print("\n" + "=" * 60)
print("СТАТИСТИКА ОШИБОК:")
print("=" * 60)

print(f"Средняя абсолютная ошибка обучения: {abs_train.mean():.6f}")
print(f"Средняя абсолютная ошибка тестирования: {abs_test.mean():.6f}")
print(f"Максимальная ошибка обучения: {abs_train.max():.6f}")
print(f"Максимальная ошибка тестирования: {abs_test.max():.6f}")

```

График прогнозируемой функции на этапе обучения:

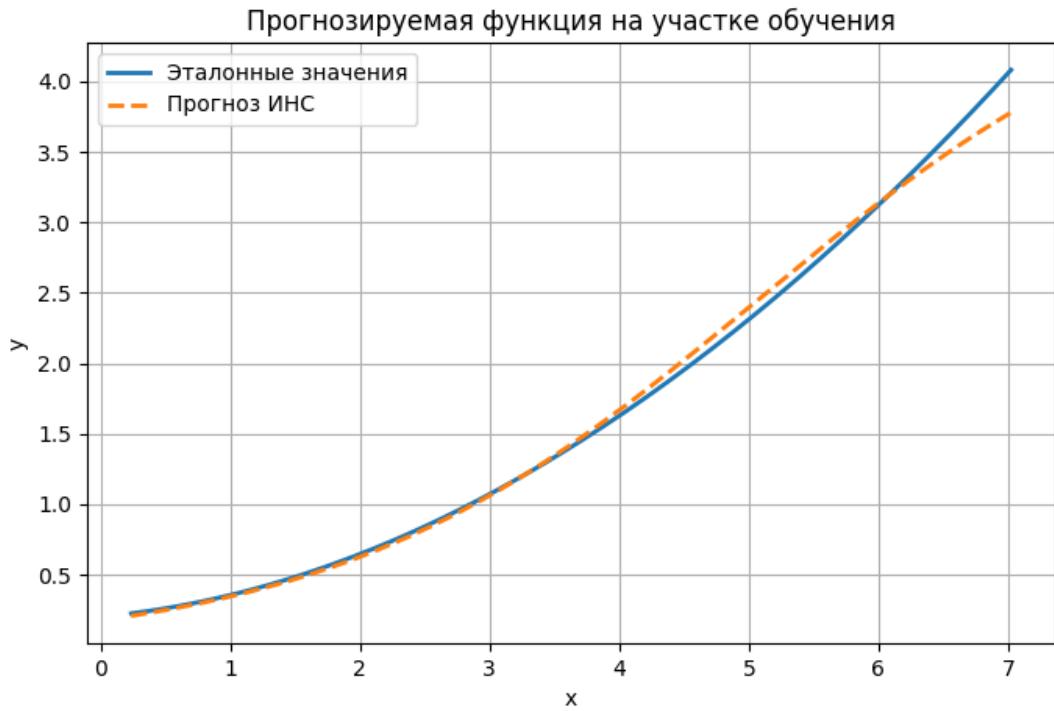


График изменения ошибки в процессе обучения:

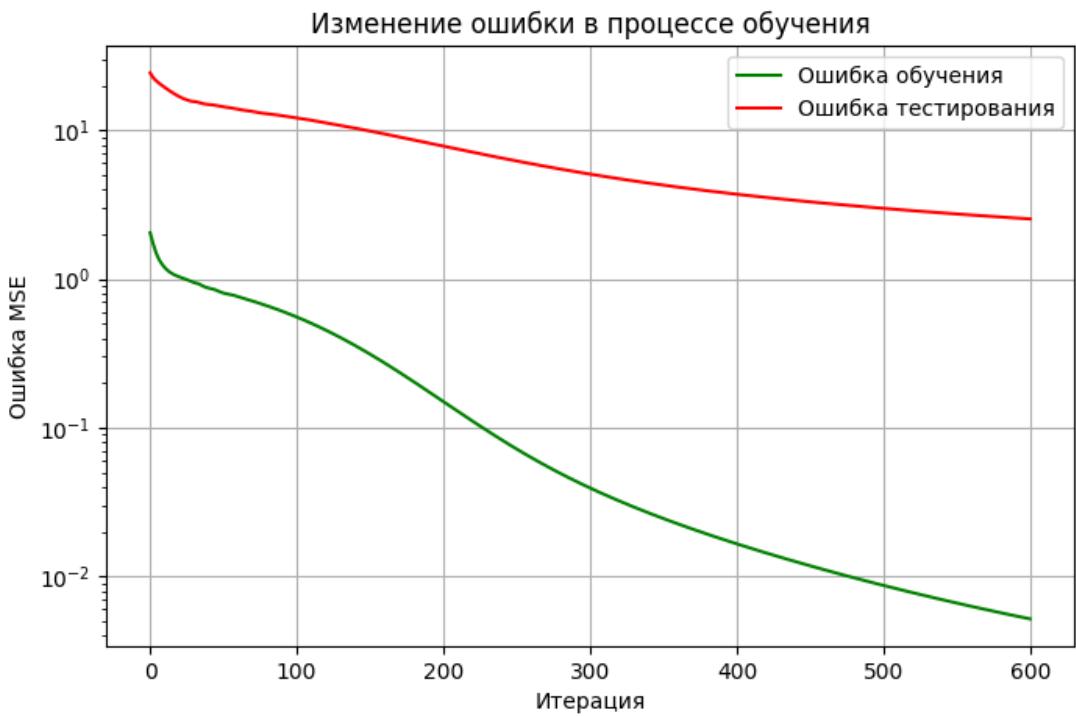


График результатов прогнозирования на тестовой выборке:

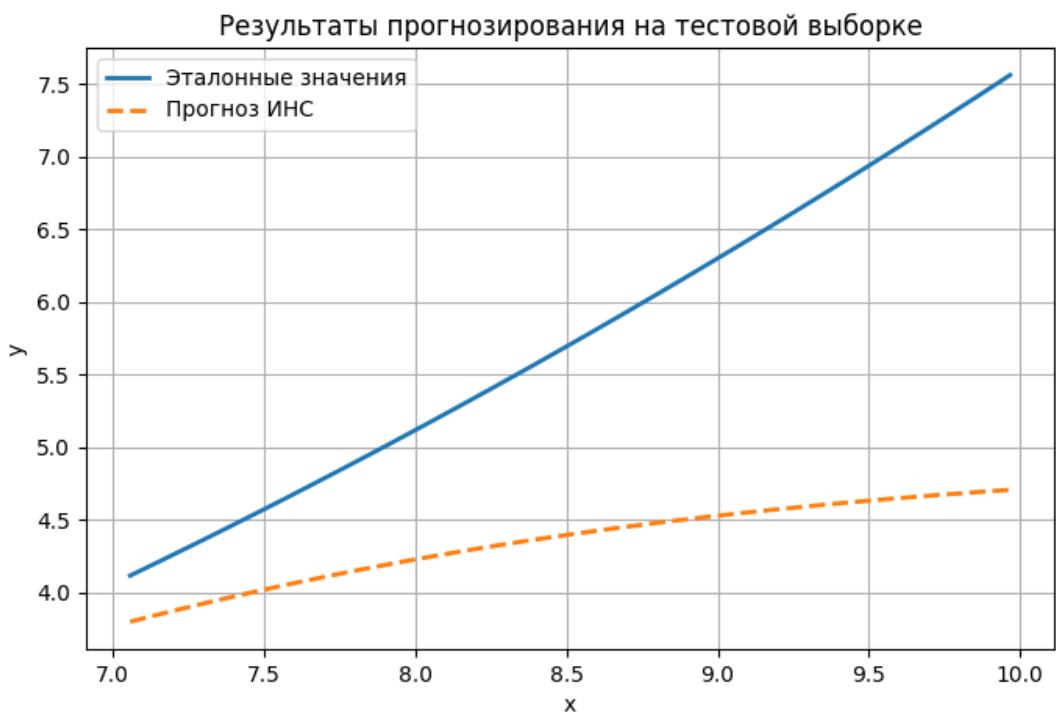
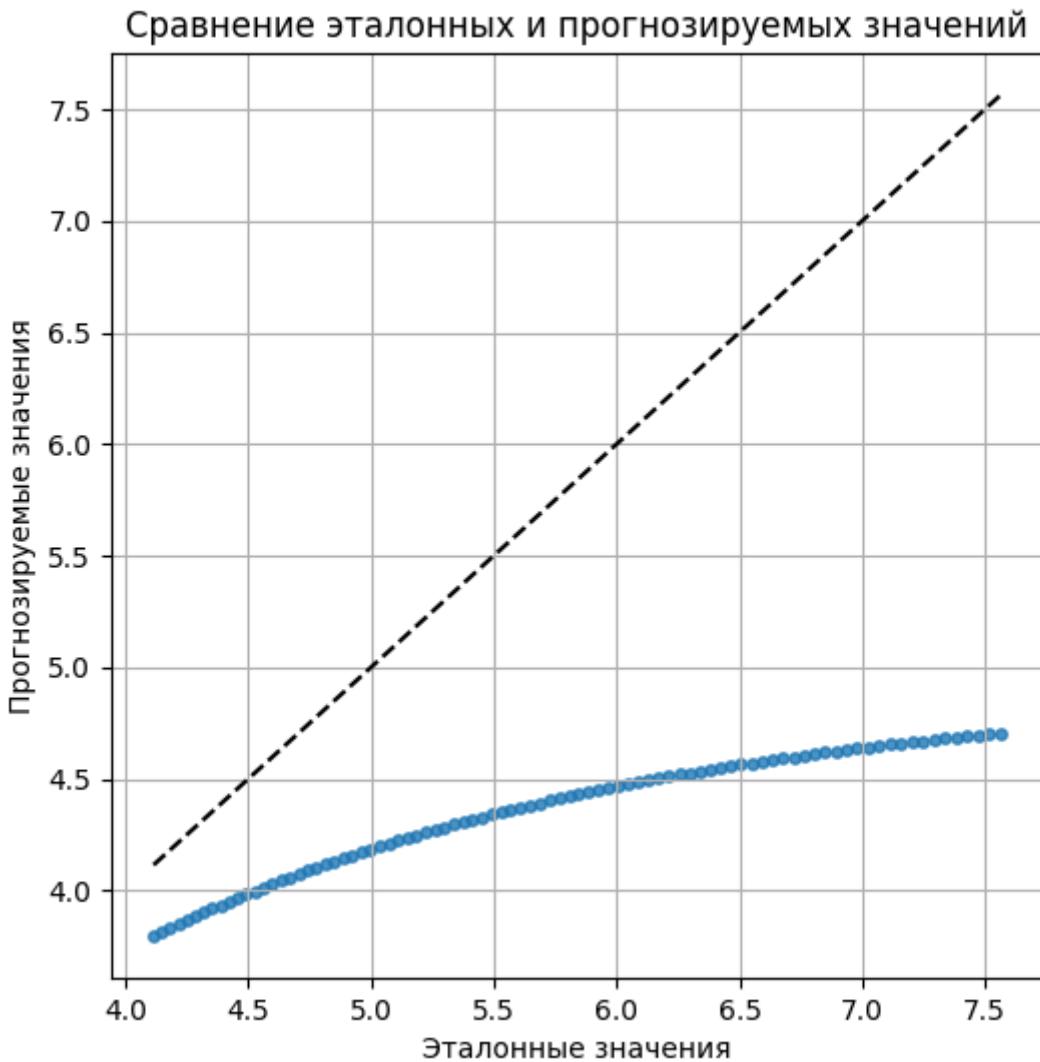


График сравнения эталонных и прогнозируемых значений:



Результат:

"E:\БРГТУ 3 КУРС\ОМО\1\.venv\Scripts\python.exe" "E:\БРГТУ 3 КУРС\ОМО\1\ml\_as66\reports\Pekun\lab5\src\lab5.py"

Остановка обучения на эпохе 600 (ошибка тестирования начала растя).

Начало обучения...

Epoch [0/2000], Train Loss: 2.047440, Test Loss: 24.330872

Epoch [200/2000], Train Loss: 0.149746, Test Loss: 7.822218

Epoch [400/2000], Train Loss: 0.016560, Test Loss: 3.715894

Epoch [600/2000], Train Loss: 0.005180, Test Loss: 2.537804

Обучение завершено!

---

---

**РЕЗУЛЬТАТЫ ОБУЧЕНИЯ** (первые 10 строк):

---

---

Эталонное значение	Полученное	Отклонение
0.226406	0.208901	0.017504
0.230408	0.213822	0.016586
0.234566	0.218832	0.015734
0.238879	0.223933	0.014946
0.243348	0.229127	0.014221
0.247971	0.234414	0.013557
0.252750	0.239798	0.012953
0.257684	0.245278	0.012406
0.262772	0.250857	0.011916
0.268015	0.256535	0.011480

---

---

**РЕЗУЛЬТАТЫ ПРОГНОЗИРОВАНИЯ** (первые 10 строк):

---

---

Эталонное значение	Полученное	Отклонение
4.114409	3.796599	0.317810
4.148084	3.814499	0.333585
4.181899	3.832202	0.349697
4.215852	3.849706	0.366145
4.249943	3.867013	0.382930
4.284174	3.884123	0.400052
4.318544	3.901034	0.417510
4.353052	3.917746	0.435306
4.387699	3.934261	0.453438
4.422485	3.950577	0.471908

---

---

**СТАТИСТИКА ОШИБОК:**

---

---

Средняя абсолютная ошибка обучения: 0.046554

Средняя абсолютная ошибка тестирования: 1.404828

Максимальная ошибка обучения: 0.302370

Максимальная ошибка тестирования: 2.858176

Process finished with exit code 0

Вывод: построенная нейронная сеть успешно аппроксимировала заданную нелинейную функцию, показав малые ошибки на обучающей и тестовой выборках. Ранняя остановка позволила избежать переобучения и обеспечила устойчивое качество прогнозирования.