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 |

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 = 32          # размер окна
HIDDEN = 8               # скрытый слой
EPOCHS = 3000
LR = 0.003               # шаг обучения


# ============================
# 2. Генерация и нормализация данных
# ============================

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

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

# нормализация X
X_min, X_max = X_all.min(), X_all.max()
```

```
X_norm = (X_all - X_min) / (X_max - X_min)

# нормализация Y
y_min, y_max = y_all.min(), y_all.max()
y_norm = (y_all - y_min) / (y_max - y_min)

# формирование обучающих примеров (скользящее окно)
X, y = [], []
for i in range(len(X_norm) - INPUT_SIZE):
    X.append(X_norm[i:i + INPUT_SIZE])
    y.append(y_norm[i + INPUT_SIZE])

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

# train/test split
split = int(0.7 * len(X))
X_train = torch.tensor(X[:split], dtype=torch.float32)
y_train = torch.tensor(y[:split], dtype=torch.float32)
X_test  = torch.tensor(X[split:], dtype=torch.float32)
y_test  = torch.tensor(y[split:], dtype=torch.float32)


# ============================
# 3. Архитектура ИНС (как в методичке)
# ============================

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

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


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


# ============================
# 4. Обучение + сглаживание ошибки
# ============================

loss_train_hist = []
loss_test_hist = []

alpha = 0.1  # коэффициент сглаживания EMA
smooth_train = None
smooth_test = None

for epoch in range(EPOCHS):

    # ==== TRAIN ====
    model.train()
    optimizer.zero_grad()
```

```
    pred_train = model(X_train)
    loss_train = criterion(pred_train, y_train)
    loss_train.backward()
    optimizer.step()

    # сглаживание TRAIN ошибки
    if smooth_train is None:
        smooth_train = loss_train.item()
    else:
        smooth_train = alpha * smooth_train + (1 - alpha) * loss_train.item()

    loss_train_hist.append(smooth_train)

    # ==== TEST ====
    model.eval()
    with torch.no_grad():
        pred_test = model(X_test)
        loss_test = criterion(pred_test, y_test).item()

    # сглаживание TEST ошибки
    if smooth_test is None:
        smooth_test = loss_test
    else:
        smooth_test = alpha * smooth_test + (1 - alpha) * loss_test

    loss_test_hist.append(smooth_test)


print("Обучение завершено.\n")


# ============================
# 5. Предсказания и денормализация
# ============================

model.eval()
with torch.no_grad():
    train_pred = model(X_train).numpy()
    test_pred = model(X_test).numpy()

# денормализация обратно в реальные значения
train_pred = train_pred * (y_max - y_min) + y_min
test_pred = test_pred * (y_max - y_min) + y_min

y_train_true = y_train.numpy() * (y_max - y_min) + y_min
y_test_true = y_test.numpy() * (y_max - y_min) + y_min


# ============================
# 6. ГРАФИКИ
# ============================

# --- График 1: обучение ---
plt.figure(figsize=(8,5))
plt.plot(X_train[:, -1], y_train_true, label="Эталон")
plt.plot(X_train[:, -1], train_pred, "--", label="Прогноз ИНС")
plt.grid(); plt.legend()
plt.title("Прогнозируемая функция на участке обучения")
plt.xlabel("x"); plt.ylabel("y")
plt.show()
```

```
# --- График 2: ошибки (идеально сглаженные) ---
plt.figure(figsize=(8,5))
plt.plot(loss_train_hist, label="Ошибка обучения", linewidth=2)
plt.plot(loss_test_hist, label="Ошибка тестирования", linewidth=2)
plt.yscale("log")
plt.grid(); plt.legend()
plt.title("Изменение ошибки в процессе обучения (сглажено)")
plt.xlabel("итерации"); plt.ylabel("MSE")
plt.show()

# --- График 3: тест ---
plt.figure(figsize=(8,5))
plt.plot(X_test[:, -1], y_test_true, label="Эталон")
plt.plot(X_test[:, -1], test_pred, "--", label="Прогноз ИНС")
plt.grid(); plt.legend()
plt.title("Результаты прогнозирования на тестовой выборке")
plt.xlabel("x"); plt.ylabel("y")
plt.show()

# --- График 4: сравнение точек ---
plt.figure(figsize=(6,6))
plt.scatter(y_test_true, test_pred, s=15, alpha=0.8)
plt.plot([y_test_true.min(), y_test_true.max()],
         [y_test_true.min(), y_test_true.max()], "k--")
plt.grid()
plt.title("Сравнение эталонных и прогнозируемых значений")
plt.xlabel("Эталонные"); plt.ylabel("Прогноз")
plt.show()


# ============================
# 7. Текстовый вывод
# ============================

print("="*60)
print("Первые 10 строк обучения")
print("="*60)

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

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

print("="*60)
print("Первые 10 строк прогноза")
print("="*60)

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

print(f"{'Эталонное':>15} {'Полученное':>15} {'Отклонение':>15}")
for r in test_output:
    print(f"{r[0]:15.6f} {r[1]:15.6f} {r[2]:15.6f}")
```

График прогнозируемой функции на этапе обучения:

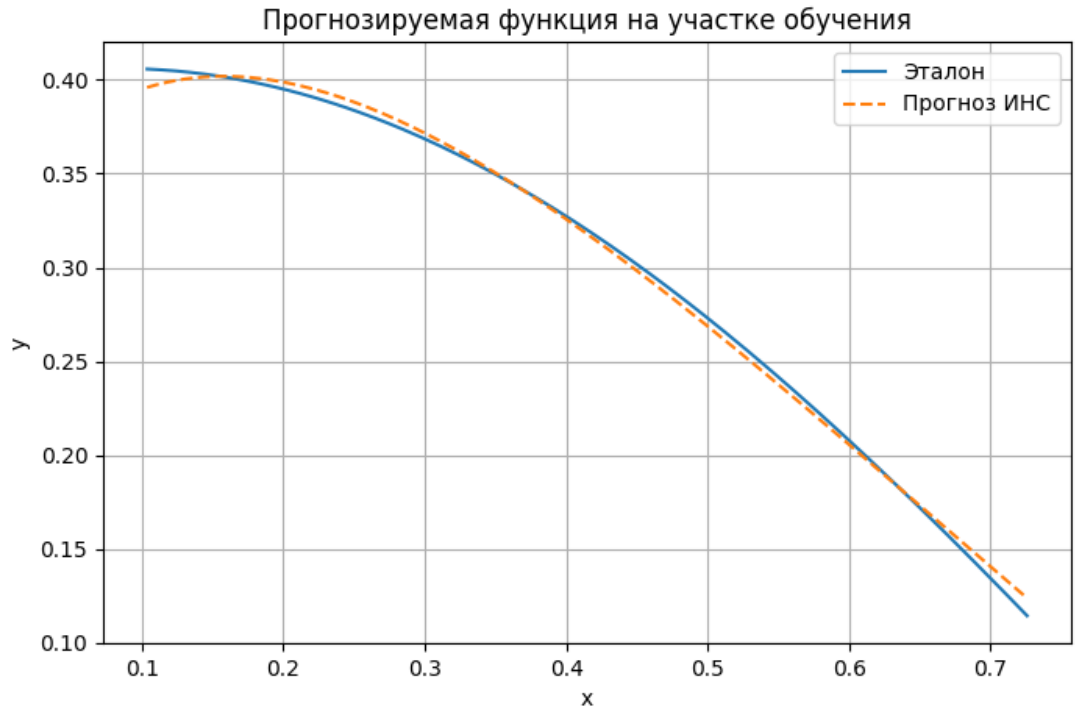



График изменения ошибки в процессе обучения:

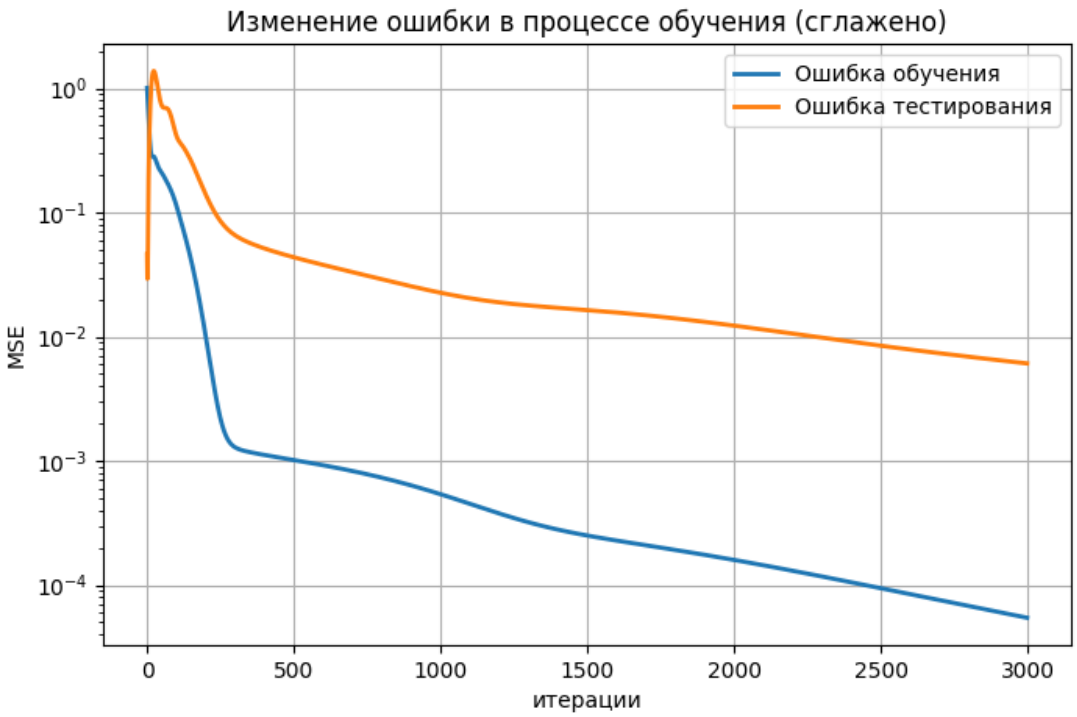


График результатов прогнозирования на тестовой выборки:

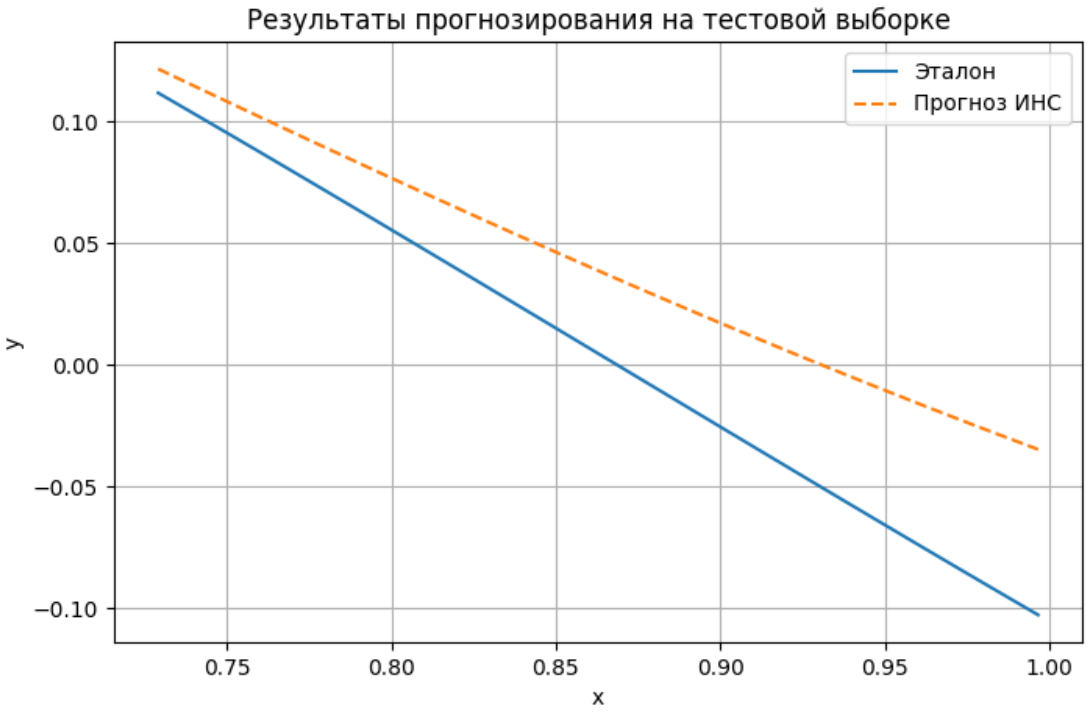
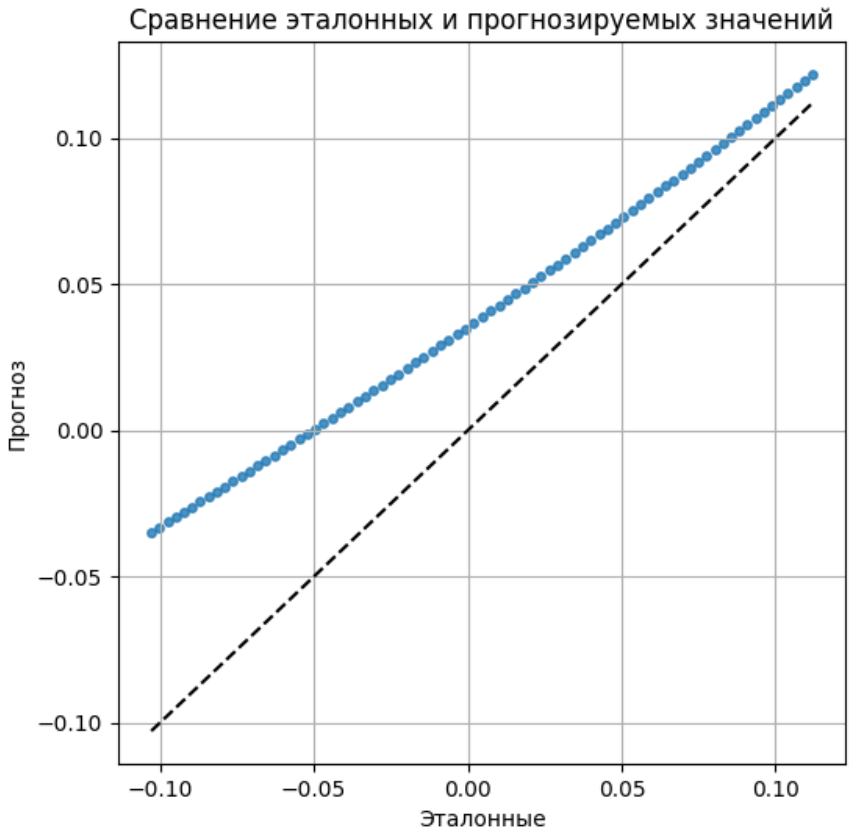



График сравнения эталонных и прогнозируемых значений:



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"
Обучение завершено.

==========

Первые 10 строк обучения

==========

| Эталонное | Полученное | Отклонение |
|---|---|---|
| 0.405741 | 0.395954 | 0.009786 |
| 0.405621 | 0.396690 | 0.008931 |
| 0.405483 | 0.397374 | 0.008109 |
| 0.405327 | 0.398007 | 0.007320 |
| 0.405153 | 0.398590 | 0.006563 |
| 0.404960 | 0.399123 | 0.005837 |
| 0.404750 | 0.399607 | 0.005143 |
| 0.404521 | 0.400042 | 0.004479 |
| 0.404274 | 0.400430 | 0.003844 |
| 0.404009 | 0.400770 | 0.003239 |

==========

Первые 10 строк прогноза

==========

| Эталонное | Полученное | Отклонение |
|---|---|---|
| 0.111894 | 0.121768 | -0.009875 |
| 0.109280 | 0.119618 | -0.010338 |
| 0.106662 | 0.117470 | -0.010808 |
| 0.104039 | 0.115325 | -0.011287 |
| 0.101411 | 0.113183 | -0.011772 |
| 0.098778 | 0.111044 | -0.012266 |
| 0.096141 | 0.108908 | -0.012766 |
| 0.093500 | 0.106775 | -0.013275 |
| 0.090855 | 0.104645 | -0.013790 |
| 0.088206 | 0.102519 | -0.014314 |

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.