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»

**Выполнила:**
Студентка 3 курса
Группы АС-66
Прокурат В. Д.
**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 9

**Задание:**

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

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

Варианты заданий приведены в следующей таблице:

| № варианта | a | b | c | d | Кол-во входов ИНС | Кол-во НЭ в скрытом слое |
|---|---|---|---|---|---|---|
| 9 | 0.1 | 0.3 | 0.08 | 0.3 | 10 | 4 |

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 pandas as pd
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt


a, b, c, d = 0.1, 0.3, 0.08, 0.3

n_inputs = 10
n_hidden = 4
n_epochs = 2000


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


x = np.linspace(0, 30, 500)
```

```
y = func(x)

X, Y = [], []
for i in range(len(y) - n_inputs):
    X.append(y[i:i + n_inputs])
    Y.append(y[i + n_inputs])

X, Y = np.array(X), np.array(Y)

split = int(len(X) * 0.8)
X_train, X_test = X[:split], X[split:]
Y_train, Y_test = Y[:split], Y[split:]

torch.manual_seed(42)
X_train, Y_train = torch.tensor(X_train, dtype=torch.float32), torch.tensor(Y_train,
dtype=torch.float32).view(-1, 1)
X_test, Y_test = torch.tensor(X_test, dtype=torch.float32), torch.tensor(Y_test,
dtype=torch.float32).view(-1, 1)


class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.hidden = nn.Linear(n_inputs, n_hidden)
        self.sigmoid = nn.Sigmoid()
        self.out = nn.Linear(n_hidden, 1)

    def forward(self, x):
        x = self.sigmoid(self.hidden(x))
        return self.out(x)


learning_rates = [0.001, 0.005, 0.01, 0.05, 0.1]
best_lr = None
best_loss = float("inf")

criterion = nn.MSELoss()

print("Подбор оптимального α:")
for lr in learning_rates:
    model = Net()
    optimizer = optim.Adam(model.parameters(), lr=lr)

    for epoch in range(700):
        optimizer.zero_grad()
        loss = criterion(model(X_train), Y_train)
        loss.backward()
        optimizer.step()

    with torch.no_grad():
        mse = criterion(model(X_test), Y_test).item()
```

```
    print(f"  α={lr:.3f}\tMSE={mse:.6f}")

    if mse < best_loss:
        best_loss = mse
        best_lr = lr

print(f"\nОптимальное значение α: {best_lr:.3f}, минимальная ошибка:
{best_loss:.6f}\n")

model = Net()
optimizer = optim.Adam(model.parameters(), lr=best_lr)
losses = []

for epoch in range(n_epochs):
    optimizer.zero_grad()
    loss = criterion(model(X_train), Y_train)
    loss.backward()
    optimizer.step()
    losses.append(loss.item())


# 4.    График прогнозируемой функции на участке обучения
with torch.no_grad():
    train_pred = model(X_train).numpy()

plt.figure(figsize=(10, 4))
plt.plot(Y_train.numpy(), label="Эталон")
plt.plot(train_pred, label="Прогноз")
plt.title("График прогнозируемой функции на участке обучения")
plt.grid(True)
plt.legend()
plt.show()


# 5.    Результаты обучения
# таблица
train_results = pd.DataFrame({
    "Эталонное значение": Y_train.numpy().flatten(),
    "Полученное значение": train_pred.flatten(),
})
train_results["Отклонение"] = train_results["Полученное значение"] -
train_results["Эталонное значение"]

print("\nРезультаты обучения (первые 10 значений):")
print(train_results.head(10))


# график
plt.figure(figsize=(7, 4))
plt.plot(losses)
plt.xlabel("Эпоха")
```

```
plt.ylabel("MSE")
plt.title("График изменения ошибки в зависимости от итерации")
plt.grid(True)
plt.show()


# 6.    Результаты прогнозирования
with torch.no_grad():
    test_pred = model(X_test).numpy()

test_results = pd.DataFrame( { "Эталонное значение": Y_test.numpy().flatten(),
"Полученное значение": test_pred.flatten(), } )
test_results["Отклонение"] = test_results["Полученное значение"] -
test_results["Эталонное значение"]

print("\nРезультаты прогнозирования (первые 10 значений):")
print(test_results.head(10))
```

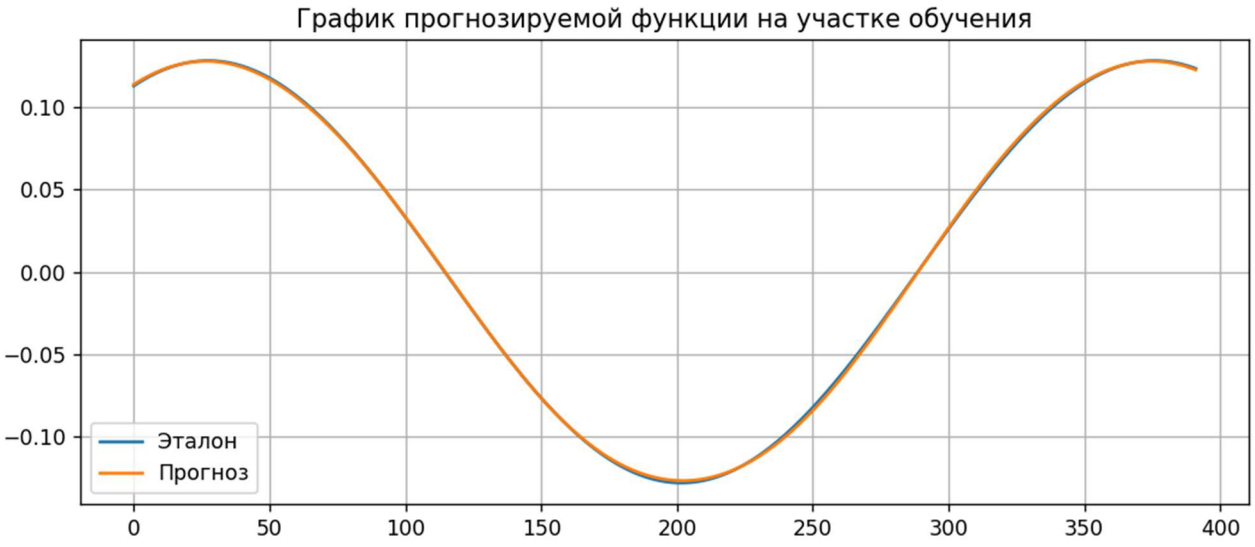
Для выбора оптимального значения α была проведена серия экспериментов с параметрами *[0.001, 0.005, 0.01, 0.05, 0.1]*. На каждом шаге сеть обучалась в течение 500 эпох, после чего вычислялась MSE на тестовой выборке.

```
Подбор оптимального α:
  α=0.001       MSE=0.000096
  α=0.005       MSE=0.000081
  α=0.010       MSE=0.000071
  α=0.050       MSE=0.000003
  α=0.100       MSE=0.000001

Оптимальное значение α: 0.100, минимальная ошибка: 0.000001
```

По результатам видно, что при α = 0.1 достигается минимальная ошибка. Значит, для дальнейшей работы будет использоваться именно это значение.

**График прогнозируемой функции на участке обучения:**



На графике показано, насколько хорошо модель повторяет исходную функцию на тех данных, на которых обучалась.

**Результаты обучения:**

Таблица со столбцами: эталонное значение, полученное значение, отклонение.

Результаты обучения (первые 10 значений):

| | Эталонное значение | Полученное значение | Отклонение |
|---|---|---|---|
| 0 | 0.112729 | 0.113593 | 0.000864 |
| 1 | 0.113806 | 0.114627 | 0.000820 |
| 2 | 0.114847 | 0.115622 | 0.000776 |
| 3 | 0.115850 | 0.116580 | 0.000730 |
| 4 | 0.116815 | 0.117500 | 0.000685 |
| 5 | 0.117743 | 0.118381 | 0.000638 |
| 6 | 0.118632 | 0.119224 | 0.000592 |
| 7 | 0.119483 | 0.120027 | 0.000545 |
| 8 | 0.120294 | 0.120792 | 0.000498 |
| 9 | 0.121067 | 0.121517 | 0.000450 |

Отклонения маленькие (< 0.001), что подтверждает успешное обучение сети.

**График изменения ошибки в зависимости от итерации:**


График показывает, что в процессе обучения среднеквадратичная ошибка (MSE) быстро падает и почти сразу стабилизируется. То есть сеть уверенно сходится.

**Результаты прогнозирования:**

Результаты прогнозирования (первые 10 значений):

| | Эталонное значение | Полученное значение | Отклонение |
|---|---|---|---|
| 0 | 0.122620 | 0.121811 | -0.000810 |
| 1 | 0.121934 | 0.121109 | -0.000825 |
| 2 | 0.121209 | 0.120370 | -0.000839 |
| 3 | 0.120443 | 0.119592 | -0.000851 |
| 4 | 0.119639 | 0.118778 | -0.000862 |
| 5 | 0.118796 | 0.117925 | -0.000871 |
| 6 | 0.117914 | 0.117036 | -0.000878 |
| 7 | 0.116994 | 0.116110 | -0.000884 |
| 8 | 0.116035 | 0.115147 | -0.000888 |
| 9 | 0.115039 | 0.114149 | -0.000891 |

Разница между эталоном и предсказанием тоже небольшая, что говорит о хорошем качестве прогноза.

**Вывод:** изучила 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.