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

Лабораторная работа №4
По дисциплине «Основы машинного обучения»
Тема: **«Введение в нейронные сети: построение многослойного перцептрона»**

Выполнила:
Студентка 3 курса
Группы АС-65
Сунцова М. Д.
Проверил:
Крощенко А. А.

Брест 2025

Цель: построить, обучить и оценить многослойный перцептрон (MLP) для решения задачи классификации.

Вариант 8

Задания:

1. Импорт библиотек и подготовка данных
2. Определение архитектуры нейронной сети
3. Инициализация модели, функции потерь и оптимизатора
4. Написание цикла обучения (Training Loop)

Оценка модели (Evaluation)

• Seeds
• Задача: классифицировать семена на три сорта (3 класса).
• Архитектура:
  ○ входной слой;
  ○ один скрытый слой с 7 нейронами (ReLU);
  ○ выходной слой с 3 нейронами (Softmax).
• Эксперимент: увеличьте количество нейронов в скрытом слое до 14 и оцените, как это повлияло на точность.

Код программы:

```
import torch
import torch.nn as nn
import torch.optim as optim
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, f1_score
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv("seeds_dataset.txt", delim_whitespace=True, header=None)
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

encoder = LabelEncoder()
y = encoder.fit_transform(y)

scaler = StandardScaler()
X = scaler.fit_transform(X)

torch.manual_seed(42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran-
dom_state=42)

X_train = torch.tensor(X_train, dtype=torch.float32)
X_test = torch.tensor(X_test, dtype=torch.float32)
y_train = torch.tensor(y_train, dtype=torch.long)
y_test = torch.tensor(y_test, dtype=torch.long)
```

```
class MLP(nn.Module):
    def __init__(self, hidden_size=7):
        super(MLP, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(7, hidden_size),
            nn.ReLU(),
            nn.Linear(hidden_size, 3)
        )

    def forward(self, x):
        return self.model(x)

hidden_size = 7
# hidden_size = 14
model = MLP(hidden_size=hidden_size)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

epochs = 100
for epoch in range(epochs):
    model.train()
    y_pred = model(X_train)
    loss = criterion(y_pred, y_train)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    if epoch % 10 == 0:
        print(f"Epoch {epoch}, Loss: {loss.item():.4f}")

model.eval()
with torch.no_grad():
    y_logits = model(X_test)
    y_pred = torch.argmax(y_logits, dim=1)

accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred, average='macro')

print(f"\nAccuracy: {accuracy:.4f}")
print(f"F1-score: {f1:.4f}")
```

```
Epoch 0, Loss: 1.1808
Epoch 10, Loss: 1.1443
Epoch 20, Loss: 1.1099
Epoch 30, Loss: 1.0769
Epoch 40, Loss: 1.0447
Epoch 50, Loss: 1.0132
Epoch 60, Loss: 0.9821
Epoch 70, Loss: 0.9508
Epoch 80, Loss: 0.9192
Epoch 90, Loss: 0.8872

Accuracy: 0.7381
F1-score: 0.7378
```

Увеличим количество нейронов в скрытом слое до 14:

```
Epoch 0, Loss: 1.2160
Epoch 10, Loss: 1.1684
Epoch 20, Loss: 1.1231
Epoch 30, Loss: 1.0785
Epoch 40, Loss: 1.0325
Epoch 50, Loss: 0.9833
Epoch 60, Loss: 0.9301
Epoch 70, Loss: 0.8723
Epoch 80, Loss: 0.8119
Epoch 90, Loss: 0.7512

Accuracy: 0.9048
F1-score: 0.9018
```

Увеличение скрытого слоя до 14 нейронов улучшило точность с 73% до 90%, а F1-score - с 0.738 до 0.902. Модель стала лучше распознавать классы благодаря большей выразительности.

Вывод: построили, обучили и оценили многослойный перцептрон (MLP) для решения задачи классификации.