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»**

Выполнил:
Студент 3 курса
Группы АС-65
Лопато А. В.
Проверил:
Крощенко А. А.

Брест 2025

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

Вариант 1

Ход работы:

Задание:

Классификация ирисов

• Iris

• Задача: определить вид ириса (3 класса).

• Архитектура:

о входной слой;

о один скрытый слой с 10 нейронами (ReLU);

о выходной слой с 3 нейронами (Softmax).

• Эксперимент: попробуйте добавить второй скрытый слой с 5 нейронами и сравните точность.

```
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

#подготовка данных
iris = load_iris()
X, y = iris.data, iris.target

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

#преобразование в тензоры PyTorch
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)

#архитектура нейронной сети
class IrisNet(nn.Module):
    def __init__(self, input_size=4, hidden1=10, hidden2=5, output_size=3):
        super(IrisNet, self).__init__()
        self.single_layer = nn.Sequential(
            nn.Linear(input_size, hidden1),
            nn.ReLU(),
            nn.Linear(hidden1, output_size)
        )
        self.double_layer = nn.Sequential(
            nn.Linear(input_size, hidden1),
            nn.ReLU(),
            nn.Linear(hidden1, hidden2),
            nn.ReLU(),
            nn.Linear(hidden2, output_size)
        )

    def forward(self, x, layers='single'):
        if layers == 'single':
            return self.single_layer(x)
        return self.double_layer(x)

#инициализация моделей
model_single = IrisNet()
model_double = IrisNet()

criterion = nn.CrossEntropyLoss()
optimizer_single = optim.Adam(model_single.parameters(), lr=0.001)
optimizer_double = optim.Adam(model_double.parameters(), lr=0.001)

#цикл обучения
def train_model(model, optimizer, architecture='single', epochs=100):
    for epoch in range(epochs):
        model.train()
        outputs = model(X_train, architecture)
        loss = criterion(outputs, y_train)
```

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

        if (epoch + 1) % 20 == 0:
            print(f'Эпоха [{epoch+1}/{epochs}], Потери: {loss.item():.4f}')

print("Обучение с одним скрытым слоем:")
train_model(model_single, optimizer_single, 'single')

print("\nОбучение с двумя скрытыми слоями:")
train_model(model_double, optimizer_double, 'double')

#оценка моделей
def evaluate_model(model, architecture='single'):
    model.eval()
    with torch.no_grad():
        outputs = model(X_test, architecture)
        _, predicted = torch.max(outputs, 1)
        accuracy = accuracy_score(y_test, predicted)
        return accuracy

acc_single = evaluate_model(model_single, 'single')
acc_double = evaluate_model(model_double, 'double')

print(f"\nТочность с одним скрытым слоем: {acc_single:.4f}")
print(f"Точность с двумя скрытыми слоями: {acc_double:.4f}")

#сравнение результатов
if acc_single > acc_double:
    print("Модель с одним скрытым слоем показала лучшую точность")
elif acc_double > acc_single:
    print("Модель с двумя скрытыми слоями показала лучшую точность")
else:
    print("Обе модели показали одинаковую точность")
```

```
Обучение с двумя скрытыми слоями:
Эпоха [20/100], Потери: 1.0915
Эпоха [40/100], Потери: 1.0550
Эпоха [60/100], Потери: 0.9936
Эпоха [80/100], Потери: 0.8985
Эпоха [100/100], Потери: 0.8013

Точность с одним скрытым слоем: 0.8000
Точность с двумя скрытыми слоями: 0.6667
Модель с одним скрытым слоем показала лучшую точность
```

Вывод: построил, обучил и оценил многослойный перцептрон (MLP) для решения задачи классификации.