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:
Студент 3 курса
Группы АС-65
Вавдийчик Н.Д.
Проверил:
Крощенко А. А.

Брест 2025

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

Ход работы

Вариант 1 Классификация ирисов

- Iris
- **Задача:** Определить вид ириса (3 класса).
- **Архитектура:**
  - входной слой;
  - один скрытый слой с 10 нейронами (ReLU);
  - выходной слой с 3 нейронами (Softmax).
- **Эксперимент:** попробуйте добавить второй скрытый слой с 5 нейронами и сравните 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, LabelEncoder
from sklearn.metrics import accuracy_score

# Загрузка данных
data = pd.read_csv('iris.csv')
X = data[['sepal.length', 'sepal.width', 'petal.length', 'petal.width']].values
y = data['variety'].values

# Кодирование меток
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Стандартизация
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Разделение на train/test
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_encoded, test_size=0.2,
random_state=42)

# Тензоры
X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train, dtype=torch.long)
y_test_tensor = torch.tensor(y_test, dtype=torch.long)


# 2. Архитектура нейронной сети
class MLP1(nn.Module):
    def __init__(self):
        super(MLP1, self).__init__()
        self.layer1 = nn.Linear(4, 10)
        self.relu = nn.ReLU()
        self.layer2 = nn.Linear(10, 3)
```

```
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        x = self.layer1(x)
        x = self.relu(x)
        x = self.layer2(x)
        x = self.softmax(x)
        return x


class MLP2(nn.Module):
    def __init__(self):
        super(MLP2, self).__init__()
        self.layer1 = nn.Linear(4, 10)
        self.relu = nn.ReLU()
        self.layer2 = nn.Linear(10, 5)
        self.layer3 = nn.Linear(5, 3)
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        x = self.layer1(x)
        x = self.relu(x)
        x = self.layer2(x)
        x = self.relu(x)
        x = self.layer3(x)
        x = self.softmax(x)
        return x


# 3. Инициализация моделей
model1 = MLP1()
model2 = MLP2()

criterion = nn.CrossEntropyLoss()
optimizer1 = optim.Adam(model1.parameters(), lr=0.001)
optimizer2 = optim.Adam(model2.parameters(), lr=0.001)

# 4. Обучение первой модели
print("Модель 1 (1 скрытый слой):")
for epoch in range(200):
    model1.train()
    outputs = model1(X_train_tensor)
    loss = criterion(outputs, y_train_tensor)

    optimizer1.zero_grad()
    loss.backward()
    optimizer1.step()

# 5. Оценка первой модели
model1.eval()
with torch.no_grad():
    test_outputs1 = model1(X_test_tensor)
    _, predicted1 = torch.max(test_outputs1, 1)
    accuracy1 = accuracy_score(y_test_tensor, predicted1)

# 4. Обучение второй модели
print("Модель 2 (2 скрытых слоя):")
for epoch in range(200):
    model2.train()
    outputs = model2(X_train_tensor)
    loss = criterion(outputs, y_train_tensor)
```

```
    optimizer2.zero_grad()
    loss.backward()
    optimizer2.step()

# 5. Оценка второй модели
model2.eval()
with torch.no_grad():
    test_outputs2 = model2(X_test_tensor)
    _, predicted2 = torch.max(test_outputs2, 1)
    accuracy2 = accuracy_score(y_test_tensor, predicted2)

# Результаты
print(f"\nТочность модели 1: {accuracy1:.4f}")
print(f"Точность модели 2: {accuracy2:.4f}")
```

```
Модель 1 (1 скрытый слой):
Модель 2 (2 скрытых слоя):

Точность модели 1: 0.8333
Точность модели 2: 0.5333
```

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