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
Зинчук М.С.
Проверил:
Крощенко А. А.

Брест 2025

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

Ход работы

Вариант 6 Диагностика диабета

- Pima Indians Diabetes
- **Задача:** предсказать наличие диабета (бинарная классификация).
- **Архитектура:**
  - входной слой;
  - два скрытых слоя: первый с 12 нейронами, второй с 8 (ReLU);
  - выходной слой с 1 нейроном (Sigmoid).
- **Эксперимент:** удалите второй скрытый слой (оставив один с 12 нейронами). Сравните recall для класса "диабет" в обеих моделях.

Код программы:

```
# 1. Импорт библиотек и подготовка данных
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, recall_score

# Загрузка данных
data = pd.read_csv('pima-indians-diabetes (1).csv', comment='#', header=None)
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# 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, 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.float32).view(-1, 1)
y_test_tensor = torch.tensor(y_test, dtype=torch.float32).view(-1, 1)


# 2. Архитектура нейронной сети
class MLP1(nn.Module):
    def __init__(self):
        super(MLP1, self).__init__()
        self.layer1 = nn.Linear(8, 12)
        self.relu = nn.ReLU()
        self.layer2 = nn.Linear(12, 8)
        self.output = nn.Linear(8, 1)
        self.sigmoid = nn.Sigmoid()
```

```
    def forward(self, x):
        x = self.layer1(x)
        x = self.relu(x)
        x = self.layer2(x)
        x = self.relu(x)
        x = self.output(x)
        x = self.sigmoid(x)
        return x


class MLP2(nn.Module):
    def __init__(self):
        super(MLP2, self).__init__()
        self.layer1 = nn.Linear(8, 12)
        self.relu = nn.ReLU()
        self.output = nn.Linear(12, 1)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = self.layer1(x)
        x = self.relu(x)
        x = self.output(x)
        x = self.sigmoid(x)
        return x


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

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

# 4. Обучение первой модели
print("Модель 1 (2 скрытых слоя):")
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 = (test_outputs1 > 0.5).float()
    accuracy1 = accuracy_score(y_test_tensor, predicted1)
    recall1 = recall_score(y_test_tensor, predicted1)

# 4. Обучение второй модели
print("Модель 2 (1 скрытый слой):")
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 = (test_outputs2 > 0.5).float()
    accuracy2 = accuracy_score(y_test_tensor, predicted2)
    recall2 = recall_score(y_test_tensor, predicted2)

# Результаты
print(f"\nСравнение моделей:")
print(f"Модель 1 (2 скрытых слоя) - Accuracy: {accuracy1:.4f}, Recall: {recall1:.4f}")
print(f"Модель 2 (1 скрытый слой) - Accuracy: {accuracy2:.4f}, Recall: {recall2:.4f}")
```

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

Сравнение моделей:
Модель 1 (2 скрытых слоя) - Accuracy: 0.7468, Recall: 0.6727
Модель 2 (1 скрытый слой) - Accuracy: 0.7727, Recall: 0.6545
```

Вывод: построила, обучила и оценила многослойный перцептрон (MLP) для решения задачи классификации.