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

Лабораторная работа №4
По дисциплине: «ОМО»

Выполнил:
Студент 3-го курса
Группы АС-66
Николова М.С
Проверил:
Крощенко А.А.

Брест 2025

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

Ход работы

Вариант 6

• Pima Indians Diabetes

• Задача: предсказать наличие диабета (бинарная классификация).

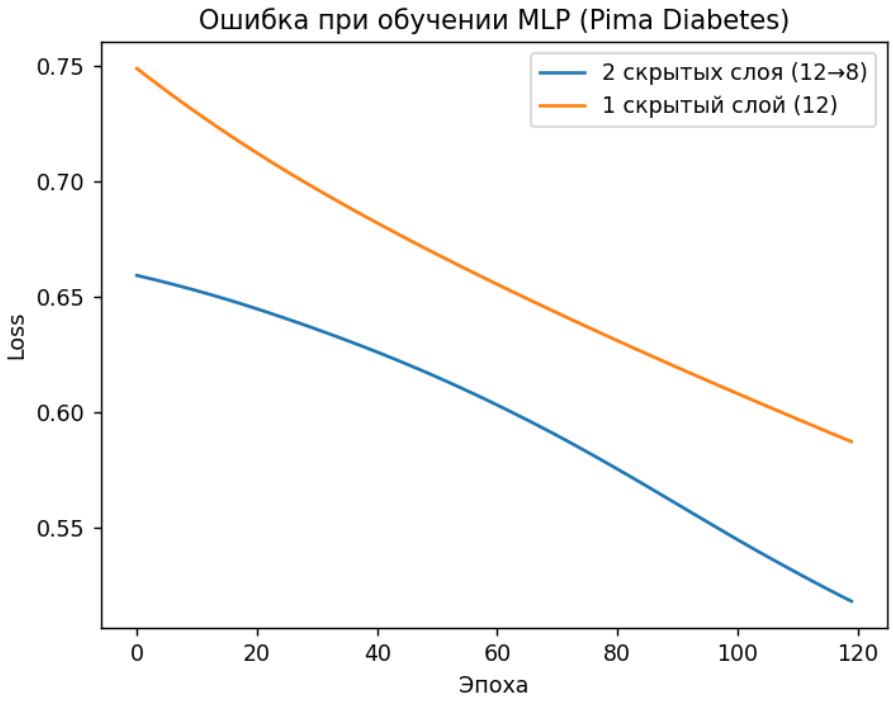
• Архитектура: о входной слой; о два скрытых слоя: первый с 12 нейронами, второй с 8 (ReLU); о выходной слой с 1 нейроном (Sigmoid).

• Эксперимент: удалите второй скрытый слой (оставив один с 12 нейронами). Сравните recall для класса "диабет" в обеих моделях.

```
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, f1_score, recall_score
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("pima-indians-diabetes.csv", comment="#", header=None)
df.columns = [
    'Pregnancies',
    'Glucose',
    'BloodPressure',
    'SkinThickness',
    'Insulin',
    'BMI',
    'DiabetesPedigreeFunction',
    'Age',
    'Class'
]
X = df.drop("Class", axis=1).values
y = df["Class"].values


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

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

X_train = torch.tensor(X_train, dtype=torch.float32)
X_test  = torch.tensor(X_test,  dtype=torch.float32)
y_train = torch.tensor(y_train.reshape(-1,1), dtype=torch.float32)
y_test  = torch.tensor(y_test.reshape(-1,1), dtype=torch.float32)

input_dim = X_train.shape[1]


#  2 скрытых слоя (12 → 8)
class MLP_2hidden(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(input_dim, 12)
```

```
        self.fc2 = nn.Linear(12, 8)
        self.fc3 = nn.Linear(8, 1)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.relu(self.fc2(x))
        x = self.fc3(x)
        return x

#  1 скрытый слой (12)
class MLP_1hidden(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(input_dim, 12)
        self.fc2 = nn.Linear(12, 1)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.fc2(x)
        return x

def train_model(model, X_train, y_train, X_test, y_test, epochs=120):
    criterion = nn.BCEWithLogitsLoss()
    optimizer = optim.Adam(model.parameters(), lr=0.001)
    losses = []

    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()

        losses.append(loss.item())

    model.eval()
    with torch.no_grad():
        logits = model(X_test)
        probs = torch.sigmoid(logits)
        y_pred_classes = (probs > 0.5).float()

    acc = accuracy_score(y_test, y_pred_classes)
    f1  = f1_score(y_test, y_pred_classes)
    recall = recall_score(y_test, y_pred_classes)

    return acc, f1, recall, losses


modelA = MLP_2hidden()
accA, f1A, recallA, lossesA = train_model(modelA, X_train, y_train, X_test, y_test)

modelB = MLP_1hidden()
accB, f1B, recallB, lossesB = train_model(modelB, X_train, y_train, X_test, y_test)
```

```
print("=== Модель A: 2 скрытых слоя (12 → 8) ===")
print("Accuracy:", round(accA, 4))
print("F1      :", round(f1A, 4))
print("Recall  :", round(recallA, 4))

print("\n=== Модель B: 1 скрытый слой (12) ===")
print("Accuracy:", round(accB, 4))
print("F1      :", round(f1B, 4))
print("Recall  :", round(recallB, 4))

plt.plot(lossesA, label="2 скрытых слоя (12→8)")
plt.plot(lossesB, label="1 скрытый слой (12)")
plt.xlabel("Эпоха")
plt.ylabel("Loss")
plt.title("Ошибка при обучении MLP (Pima Diabetes)")
plt.legend()
plt.show()
```



Модель A: 2 скрытых слоя (12 → 8)
Accuracy: 0.7273
F1   : 0.5532
Recall : 0.4727

Модель B: 1 скрытый слой (12)
Accuracy: 0.7208
F1   : 0.6055
Recall : 0.6
Вывод: я построила, обучила и оценила многослойный перцептрон (MLP) для решения задачи классификации.