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 2 курса
Группы АС-66
Колбашко А. В.
Проверил:
Крощенко А. А.

Брест 2025

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

Ход работы

Вариант 9 Определение вида стекла

- Glass Identification
- **Задача:** классифицировать тип стекла (многоклассовая задача).
- **Архитектура:**
  - входной слой;
  - два скрытых слоя по 10 нейронов в каждом (ReLU);
  - выходной слой с необходимым количеством нейронов (Softmax).
- **Эксперимент:** попробуйте обучить модель с одним скрытым слоем на 20 нейронов. Какая архитектура оказалась лучше?

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, classification_report
import numpy as np

data = pd.read_csv("C:\\Users\\wlksm\\OneDrive\\Рабочий
стол\\Уник\\PP\\Parser\\4\\src\\glass.csv")

X = data.drop("Type", axis=1).values
y = data["Type"].values

unique_classes = np.sort(np.unique(y))
class_to_index = {c: i for i, c in enumerate(unique_classes)}
y = np.array([class_to_index[c] for c in y])

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

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.25, random_state=16, stratify=y
)

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)

num_features = X_train.shape[1]
num_classes = len(unique_classes)

# Два скрытых слоя по 10 нейронов
class MLP_two_hidden(nn.Module):
    def __init__(self, input_size, num_classes):
```

```
        super(MLP_two_hidden, self).__init__()
        self.net = nn.Sequential(
            nn.Linear(input_size, 10),
            nn.ReLU(),
            nn.Linear(10, 10),
            nn.ReLU(),
            nn.Linear(10, num_classes)
        )

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

# один скрытый слой на 20 нейронов
class MLP_one_hidden(nn.Module):
    def __init__(self, input_size, num_classes):
        super(MLP_one_hidden, self).__init__()
        self.net = nn.Sequential(
            nn.Linear(input_size, 20),
            nn.ReLU(),
            nn.Linear(20, num_classes)
        )

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

model1 = MLP_two_hidden(num_features, num_classes)
model2 = MLP_one_hidden(num_features, num_classes)

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

def train_model(model, optimizer, X_train, y_train, 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()
    return model

model1 = train_model(model1, optimizer1, X_train, y_train)
model2 = train_model(model2, optimizer2, X_train, y_train)

def evaluate_model(model, X_test, y_test):
    model.eval()
    with torch.no_grad():
        outputs = model(X_test)
        _, preds = torch.max(outputs, 1)
    acc = accuracy_score(y_test, preds)
    f1 = f1_score(y_test, preds, average="weighted")
    return acc, f1, preds

acc1, f1_1, preds1 = evaluate_model(model1, X_test, y_test)
acc2, f1_2, preds2 = evaluate_model(model2, X_test, y_test)

print("\nРезультаты")
print(f"2 скрытых слоя по 10 нейронов: Accuracy = {acc1:.3f}, F1 = {f1_1:.3f}")
print(f"1 скрытый слой на 20 нейронов: Accuracy = {acc2:.3f}, F1 = {f1_2:.3f}")

print("\nОценка моделей")
print(classification_report(y_test, preds1))
print(classification_report(y_test, preds2))
```

Результат выполнения программы:

Результаты

2 скрытых слоя по 10 нейронов: Accuracy = 0.352, F1 = 0.183

1 скрытый слой на 20 нейронов: Accuracy = 0.519, F1 = 0.462

Оценка моделей

1) 2 скрытых слоя по 10 нейронов

	precision	recall	f1-score	support
0	0.00	0.00	0.00	18
1	0.35	1.00	0.52	19
2	0.00	0.00	0.00	4
3	0.00	0.00	0.00	3
4	0.00	0.00	0.00	2
5	0.00	0.00	0.00	8
accuracy			0.35	54
macro avg	0.06	0.17	0.09	54
weighted avg	0.12	0.35	0.18	54

2) 1 скрытый слой по 20 нейронов

	precision	recall	f1-score	support
0	0.50	0.33	0.40	18
1	0.44	0.79	0.57	19
2	0.00	0.00	0.00	4
3	0.00	0.00	0.00	3
4	0.00	0.00	0.00	2
5	0.88	0.88	0.88	8
accuracy			0.52	54
macro avg	0.30	0.33	0.31	54
weighted avg	0.45	0.52	0.46	54