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
Ракецкий П. П.
Проверил:
Крощенко А. А.

Брест 2025

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

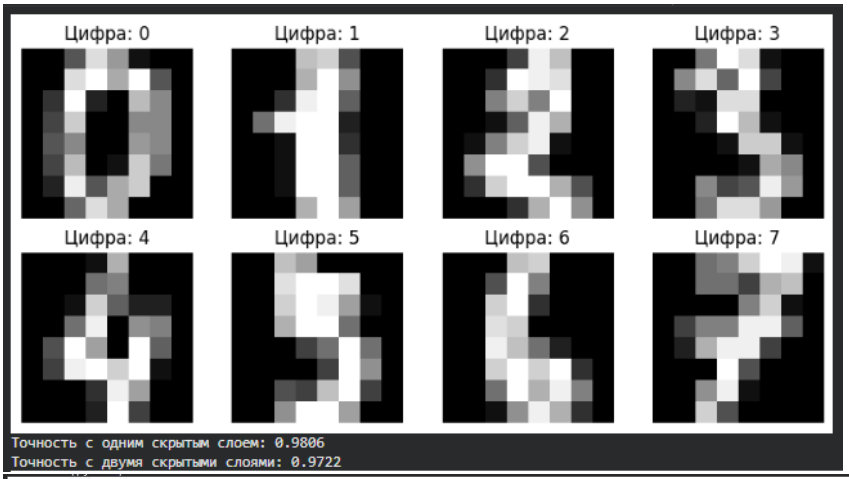
Вариант 4

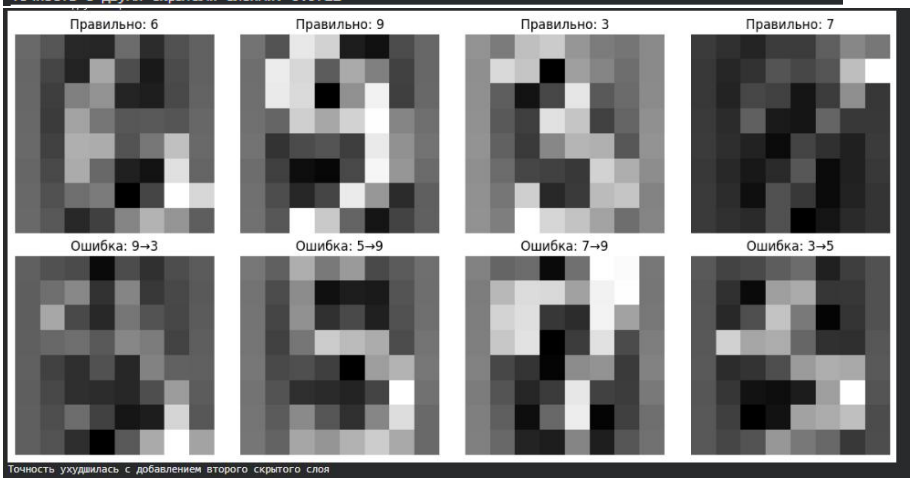
Общий план для всех вариантов:
1. Импорт библиотек и подготовка данных
• импортируйте torch, torch.nn, torch.optim, а также sklearn для загрузки данных и их предобработки;
• загрузите датасет, выполните стандартизацию (StandardScaler) и кодирование признаков;
• разделите данные на обучающую и тестовую выборки;
• преобразуйте данные (признаки и метки) в тензоры PyTorch: torch.tensor(X_train, dtype=torch.float32).
2. Определение архитектуры нейронной сети
• создайте класс, наследуемый от torch.nn.Module;
• в методе __init__ определите все слои, которые будете использовать (например, nn.Linear, nn.ReLU, nn.Dropout);
• в методе forward опишите последовательность применения слоев к входным данным.
3. Инициализация модели, функции потерь и оптимизатора
• создайте экземпляр вашей модели: model = MLP();
• определите функцию потерь. Для бинарной классификации используйте nn.BCEWithLogitsLoss, для многоклассовой – nn.CrossEntropyLoss;
• определите оптимизатор:
optimizer = torch.optim.Adam(model.parameters(), lr=0.001).
4. Написание цикла обучения (Training Loop)
• запустите цикл на определенное количество эпох;
• внутри цикла:
1. переведите модель в режим обучения: model.train();
2. сделайте предсказание (forward pass):
y_pred = model(X_train);
3. рассчитайте потери (loss):
loss = criterion(y_pred, y_train);
4. обнулите градиенты: optimizer.zero_grad();
5. выполните обратное распространение ошибки:
loss.backward();
6. сделайте шаг оптимизации: optimizer.step().
5. Оценка модели (Evaluation)
• переведите модель в режим оценки: model.eval();
• используйте with torch.no_grad():, чтобы отключить расчет

градиентов;
• сделайте предсказания на тестовых данных;
• преобразуйте выходные данные (логиты) в предсказанные классы (например, с помощью torch.argmax или проверки порога > 0);
• рассчитайте метрики (accuracy, f1-score и т.д.), используя sklearn.metrics.
Вариант 4 Распознавание рукописных цифр
• Digits
• Задача: распознать цифру от 0 до 9 (10 классов).
• Архитектура:
о входной слой;
о один скрытый слой с 32 нейронами (ReLU);
о выходной слой с 10 нейронами (Softmax).
• Эксперимент: сравните результаты с архитектурой, где два скрытых слоя по 32 нейрона. Улучшилась ли точность?

```
# 1. Импорт библиотек и подготовка данных
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np

# Загрузка датасета
digits = load_digits()
X, y = digits.data, digits.target

# Покажем пример изображения
plt.figure(figsize=(8, 4))
for i in range(8):
    plt.subplot(2, 4, i + 1)
    plt.imshow(digits.images[i], cmap='gray')
    plt.title(f'Цифра: {digits.target[i]}')
    plt.axis('off')
plt.tight_layout()
plt.show()

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

# Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, ran-
dom_state=42)

# 2. Модель с одним скрытым слоем (32 нейрона)
model_one_layer = MLPClassifier(
    hidden_layer_sizes=(32,),
    activation='relu',
    solver='adam',
    learning_rate_init=0.001,
    max_iter=500,
    random_state=42
)
```

```
# 3. Модель с двумя скрытыми слоями (по 32 нейрона)
model_two_layers = MLPClassifier(
    hidden_layer_sizes=(32, 32),
    activation='relu',
    solver='adam',
    learning_rate_init=0.001,
    max_iter=500,
    random_state=42
)

# 4. Обучение моделей
model_one_layer.fit(X_train, y_train)
model_two_layers.fit(X_train, y_train)

# 5. Оценка моделей
y_pred_one = model_one_layer.predict(X_test)
y_pred_two = model_two_layers.predict(X_test)

accuracy_one = accuracy_score(y_test, y_pred_one)
accuracy_two = accuracy_score(y_test, y_pred_two)

print(f"Точность с одним скрытым слоем: {accuracy_one:.4f}")
print(f"Точность с двумя скрытыми слоями: {accuracy_two:.4f}")

# Визуализация примеров предсказаний
plt.figure(figsize=(12, 6))

# Примеры правильных предсказаний
correct_indices = np.where(y_pred_one == y_test)[0][:4]

for i, idx in enumerate(correct_indices):
    plt.subplot(2, 4, i + 1)
    plt.imshow(X_test[idx].reshape(8, 8), cmap='gray')
    plt.title(f'Правильно: {y_test[idx]}')
    plt.axis('off')

# Примеры неправильных предсказаний
wrong_indices = np.where(y_pred_one != y_test)[0]
if len(wrong_indices) > 0:
    for i, idx in enumerate(wrong_indices[:4]):
        plt.subplot(2, 4, i + 5)
        plt.imshow(X_test[idx].reshape(8, 8), cmap='gray')
        plt.title(f'Ошибка: {y_test[idx]}→{y_pred_one[idx]}')
        plt.axis('off')

plt.tight_layout()
plt.show()

if accuracy_two > accuracy_one:
    print("Точность улучшилась с добавлением второго скрытого слоя")
elif accuracy_one > accuracy_two:
    print("Точность ухудшилась с добавлением второго скрытого слоя")
else:
    print("Точность осталась одинаковой")
```





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