



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
**«Московский государственный технический университет имени  
Н.Э. Баумана**  
(национальный исследовательский университет)  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

## **Лабораторная работа № 7 по дисциплине «Методы машинного обучения»**

**Тема** Классификатор на базе многослойного персептрона

**Студент** Сапожков А.М.

**Группа** ИУ7-23М

**Преподаватель** Солодовников В.И.

# Содержание

<b>ВВЕДЕНИЕ</b>	4
<b>1 Аналитическая часть</b>	5
1.1 Классификация	5
1.2 Искусственная нейронная сеть	5
1.3 Методы обучения многослойного персептрана	5
<b>2 Конструкторская часть</b>	7
2.1 Обучение моделей	7
<b>3 Технологическая часть</b>	8
3.1 Средства реализации	8
3.2 Реализация алгоритмов	8
<b>4 Исследовательская часть</b>	15
4.1 Среда для тестирования	15
4.2 Использование ReLu в качестве функции активации	15
4.3 Использование сигмоиды ( $b = 1$ ) в качестве функции активации	18
4.4 Использование сигмоиды ( $b = 5$ ) в качестве функции активации	21
4.5 Использование сигмоиды ( $b = 10$ ) в качестве функции активации для построения разделяющих гиперплоскостей на основе весов нейронов скрытого слоя	24
4.6 Добавление класса для отделения данных, далеко отстоящих от исходных классов	28
<b>ЗАКЛЮЧЕНИЕ</b>	31

# ВВЕДЕНИЕ

Искусственные нейронные сети, в частности многослойные персептроны, являются одним из ключевых инструментов в машинном обучении для решения задач классификации. Эти модели, основанные на принципе имитации работы биологических нейронных сетей, позволяют эффективно аппроксимировать сложные нелинейные зависимости и обрабатывать многомерные данные, что делает их универсальными классификаторами.

Целью данной лабораторной работы является изучение нейросетевого подхода на примере построения классификатора на базе многослойного персептрана.

Задачи данной лабораторной работы:

- 1) осуществить генерацию исходных данных, которые представляют собой двумерное признаковое пространство, сгруппированное в 6 или более областей, отнесённых не менее чем к 4 классам;
- 2) для сгенерированного датасета осуществить построение классификатора на базе многослойного персептрана;
- 3) обосновать выбор числа слоев и нейронов в каждом слое;
- 4) сравнить работу нейросети в зависимости от выбранной функции активации (сигмомида с разными значениями параметра крутизны ( $b=1, b=100$ ); ReLU );
- 5) обосновать момент остановки процесса обучения;
- 6) оценить точность, полноту, F-меру;
- 7) построить матрицу ошибок;
- 8) предусмотреть дополнительную возможность ввода пользователем новых, не входящих в сгенерированный датасет данных.

# **1 Аналитическая часть**

## **1.1 Классификация**

Классификация (classification) — это задача присвоения меток класса (class label) наблюдениям (Observation) объектам из предметной области. Множество допустимых меток класса конечно. В свою очередь класс — это множество всех объектов с данным значением метки. Требуется построить алгоритм, способный классифицировать (присвоить метку) произвольный объект из исходного множества. Классификация, как правило, на этапе настройки использует обучение с учителем.

## **1.2 Искусственная нейронная сеть**

Математическая модель, а также ее программные или аппаратные реализации, построенная в некотором смысле по образу и подобию сетей нервных клеток живого организма. Под искусственной нейронной сетью в дальнейшем будем понимать сеть искусственных нейронов, соединенных между собой. Здесь предполагается, что нейроны могут соединяться между собой произвольным образом и образовывать таким образом разнообразные нейронные структуры.

- С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа.
- С точки зрения математики, обучение нейронных сетей — это многопараметрическая задача нелинейной оптимизации.
- С точки зрения кибернетики, нейронная сеть используется в задачах адаптивного управления и как алгоритмы для робототехники.
- С точки зрения развития вычислительной техники и программирования, нейронная сеть — способ решения проблемы эффективного параллелизма.
- С точки зрения искусственного интеллекта, ИНС является основой философского течения коннекционизма и основным направлением в структурном подходе по изучению возможности построения (моделирования) естественного интеллекта с помощью компьютерных алгоритмов.

## **1.3 Методы обучения многослойного персептрона**

Обучение нейронной сети — интерактивный процесс корректировки синаптических весов и порогов. В процессе обучения нейронная сеть получает и обобщает знания об окружающей среде и тех данных, с которыми ей придется оперировать. Существуют два концептуальных подхода к обучению нейронных сетей: обучение с учителем и обучение без учителя.

Обучение персептрона:

- обучение Хебба (без учителя);
- стохастические методы обучения («имитация обжига»);

— обратное распространение ошибки (Backpropagation).

## **2 Конструкторская часть**

### **2.1 Обучение моделей**

Для обучения моделей использовалась следующая архитектура нейронной сети:

- Входной слой — 2 нейрона (в соответствии с размерностью исходных данных).
- Скрытый слой — 6 нейронов.
- Выходной слой — 4 нейрона (в соответствии с количеством целевых классов).

В качестве функций активации по заданию лабораторной работы будут использоваться функции ReLu и логистическая с параметрами 1 и 5.

В качестве метрик обучения предлагается использовать accuracy, precision и recall.

### 3 Технологическая часть

#### 3.1 Средства реализации

В качестве языка программирования для реализации алгоритмов был выбран язык программирования Python ввиду наличия библиотек для обучения регрессионных моделей, таких как sklearn и numpy.

#### 3.2 Реализация алгоритмов

На листинге 3.1 представлена реализация алгоритма обучения классификатора на основе многослойного персептрона.

Листинг 3.1 — Классификация с использованием нейросетевого подхода

```
import math
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator, FormatStrFormatter
from sklearn.neural_network._multilayer_perceptron import
    MLPClassifier
from sklearn.neural_network._base import ACTIVATIONS
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, recall_score,
    precision_score, f1_score, confusion_matrix, classification_report
    , matthews_corrcoef
import seaborn as sns

import tensorflow
from tensorflow.keras import losses, metrics, optimizers
from tensorflow.keras.initializers import HeNormal, GlorotNormal
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras import backend
from tensorflow.keras.utils import to_categorical
import tensorflow.math as tfm

import warnings
warnings.filterwarnings('ignore')

centers = [[0, 0], [3.5, 3.5], [5.5, 5.5], [9, 9], [7.5, 1], [1,
    7.5]]
```

```

X, y = make_blobs(n_samples=1200, centers=centers, random_state=7)
y = list(map(lambda x: x if (x < 2) else x - 1, y))
y = list(map(lambda x: x if (x < 4) else x - 1, y))

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
print(np.unique(y_test))

plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, edgecolors='k',
            label='Обучающая выборка')
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=plt.cm.RdBu,
            marker='x', s=50, label='Тестовая выборка')
plt.title('Сгенерированные данные')
plt.xlabel('Признак 1')
plt.ylabel('Признак 2')
plt.show()

feature_vector_length = 2
num_classes = 4
input_shape = (feature_vector_length,)

y_train = to_categorical(y_train, num_classes)
y_test = to_categorical(y_test, num_classes)

def plot_surfaces(model, X_train, y_train, X_test, y_test, h=0.02):
    x_min, x_max = X[:, 0].min() - 6, X[:, 0].max() + 6
    y_min, y_max = X[:, 1].min() - 6, X[:, 1].max() + 6
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()], verbose=1)
    Z = np.argmax(Z, axis=1).reshape(xx.shape)

    plt.figure(figsize=(10, 10))
    plt.contourf(xx, yy, Z, cmap=plt.cm.RdBu, alpha=.8)
    plt.scatter(X_train[:, 0], X_train[:, 1], c=np.argmax(y_train, axis=1),
                cmap=plt.cm.RdBu, edgecolors='k', label='Обучающая выборка')
    plt.scatter(X_test[:, 0], X_test[:, 1], c=np.argmax(y_test, axis=1),
                cmap=plt.cm.RdBu, marker='x', s=50, label='Тестовая выборка')
    plt.title(f'Разделяющие поверхности')

```

```

plt.xlabel('Признак 1')
plt.ylabel('Признак 2')
plt.legend()
plt.show()

def plot_results(metrics, title=None, ylabel=None, xlim=None, ylim=
    None, metric_name=None, color=None):
    fig, ax = plt.subplots(figsize=(15, 4))

    if not (isinstance(metric_name, list) or isinstance(metric_name,
        tuple)):
        metrics = [metrics,]
        metric_name = [metric_name,]

    for idx, metric in enumerate(metrics):
        ax.plot(metric, color=color[idx])

    plt.xlabel("Epoch")
    plt.ylabel(ylabel)
    plt.title(title)
    plt.xlim(xlim)
    plt.ylim(ylim)

    ax.xaxis.set_major_locator(MultipleLocator((xlim[1] - xlim[0])/10))
    ax.xaxis.set_major_formatter(FormatStrFormatter('%d'))
    ax.xaxis.set_minor_locator(MultipleLocator(1))
    plt.grid(True)
    plt.legend(metric_name)
    plt.show()
    plt.close()

def learn(X_train, y_train, X_test, y_test, activation_input,
    activation_hidden, activation_output, additional_metrics=[]):
    num_classes = y_train.shape[1]
    model = Sequential()
    initializer = GlorotNormal()
    model.add(InputLayer(input_shape=input_shape, activation=
        activation_input, kernel_initializer=initializer))
    model.add(Dense(6, activation=activation_hidden, kernel_initializer
        =initializer))
    model.add(Dense(num_classes, activation=activation_output,

```

```

        kernel_initializer=initializer))

metrics = ['accuracy']
metrics.extend(additional_metrics)
model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=metrics,
)

max_epochs = 200
epochs_per_iter = 25
train_loss = []
train_acc = []
valid_loss = []
valid_acc = []

epoch = 0
while epoch < max_epochs:
    training_results = model.fit(X_train, y_train, batch_size=16,
        initial_epoch=epoch, epochs=epoch+epochs_per_iter, verbose=1,
        validation_data=(X_test, y_test))
    print(training_results.history.keys())
    train_loss.extend(training_results.history["loss"])
    train_acc.extend(training_results.history["accuracy"])
    valid_loss.extend(training_results.history["val_loss"])
    valid_acc.extend(training_results.history["val_accuracy"])
    plot_surfaces(model, X_train, y_train, X_test, y_test, 0.05)
    epoch += epochs_per_iter

plot_results([ train_loss, valid_loss ],
             ylabel="Loss",
             xlim = [0, max_epochs],
             ylim = [0.0, 1.5],
             metric_name=["Training Loss", "Validation Loss"],
             color=["g", "b"])

plot_results([ train_acc, valid_acc ],
             ylabel="Accuracy",
             xlim = [0, max_epochs],
             ylim = [0.0, 1.0],
             metric_name=["Training Accuracy", "Validation Accuracy"]
             ],

```

```

        color=["g", "b"])

y_pred = model.predict(X_test)
print("\nClassification Report:")
print(classification_report(np.argmax(y_test, axis=1), np.argmax(
    y_pred, axis=1)))
print("\nAdditional Metrics:")
mcc = matthews_corrcoef(np.argmax(y_test, axis=1), np.argmax(y_pred,
    axis=1))
print(f"MCC: {mcc:.4f}\n")
plt.figure(figsize=(8, 6))
cm = confusion_matrix(np.argmax(y_test, axis=1), np.argmax(y_pred,
    axis=1))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
return model

model_1 = learn(X_train, y_train, X_test, y_test, 'relu', 'relu', 'softmax',
    ['precision', 'recall'])
model_2 = learn(X_train, y_train, X_test, y_test, 'sigmoid', 'sigmoid',
    'softmax')

def sigmoid_b5(x):
    return tfm.divide(1.0, tfm.add(1.0, tfm.exp(tfm.multiply(x, -5.0))))
    )

actv = Activation(sigmoid_b5)
model_3 = learn(X_train, y_train, X_test, y_test, actv, actv, 'softmax')
model_4 = learn(X_train, y_train, X_test, y_test, 'sigmoid', 'sigmoid',
    'sigmoid', ['precision', 'recall'])

weights, biases = model_4.layers[-2].get_weights()
print("Весовая матрица скрытого слоя:\n", weights)
print("Смещения скрытого слоя:\n", biases)
print()

x_min, x_max = X[:, 0].min() - 3, X[:, 0].max() + 3

```

```

y_min, y_max = X[:, 1].min() - 3, X[:, 1].max() + 3
x_vals = np.linspace(x_min, x_max, 100)
plt.figure(figsize=(10,8))
plt.scatter(X_train[:,0], X_train[:,1], c=np.argmax(y_train, axis=1),
            edgecolors='k', label='Обучающая
            выборка')
plt.scatter(X_test[:,0], X_test[:,1], c=np.argmax(y_test, axis=1),
            marker='x', s=80, label='Тестовая
            выборка')
colors = ['red', 'blue', 'green', 'orange', 'magenta', 'pink']
for i in range(len(weights[0])):
    w = weights[:, i]
    b = biases[i]
    if np.abs(w[1]) > 1e-6:
        y_vals = -(w[0]/w[1])*x_vals - b/w[1]
        plt.plot(x_vals, y_vals, color=colors[i], linestyle='--',
                  label=f'Разделяющая линия для нейрона {i}')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.title("Разделяющие линии, полученные из весов нейронов скрытого слоя")
plt.xlabel("Признак 1")
plt.ylabel("Признак 2")
plt.legend()
plt.show()

center_x, center_y = np.mean(X[:, 0]), np.mean(X[:, 1])
r = 12
max_class = np.max(y)
X_new, y_new = X, y
for angle in np.linspace(0, 2*math.pi, 720):
    X_new = np.vstack([X_new, [center_x+r*math.cos(angle), center_y+r*
        math.sin(angle)]])
    y_new = np.hstack([y_new, np.asarray([max_class+1]).reshape((1,))])
X_train_new, X_test_new, y_train_new_src, y_test_new_src =
    train_test_split(X_new, y_new, test_size=0.2)
plt.scatter(X_train_new[:, 0], X_train_new[:, 1], c=y_train_new_src,
            edgecolors='k', label='Обучающая
            выборка')
plt.scatter(X_test_new[:, 0], X_test_new[:, 1], c=y_test_new_src,
            cmap=plt.cm.RdBu, marker='x', s=50, label='Тестовая
            выборка')
plt.title('Сгенерированные данные')
plt.xlabel('Признак 1')

```

```

plt.ylabel('Признак 2')
plt.show()
y_train_new = to_categorical(y_train_new_src, num_classes+1)
y_test_new = to_categorical(y_test_new_src, num_classes+1)
model_5 = learn(X_train_new, y_train_new, X_test_new, y_test_new,
    'relu', 'relu', 'softmax', ['precision', 'recall'])

X_user = np.array([
    [0, 0],
    [12, -3],
    [15, -4],
    [-5, 13]
], dtype=np.float64)

y_pred = model_5.predict(X_user)
np.argmax(y_pred, axis=1)

h=0.1
x_min, x_max = X_new[:, 0].min() - 3, X_new[:, 0].max() + 3
y_min, y_max = X_new[:, 1].min() - 3, X_new[:, 1].max() + 3
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min,
    y_max, h))
Z = model_5.predict(np.c_[xx.ravel(), yy.ravel()], verbose=1)
Z = np.argmax(Z, axis=1).reshape(xx.shape)

plt.figure(figsize=(10, 10))
plt.contourf(xx, yy, Z, cmap=plt.cm.RdBu, alpha=.7)
plt.scatter(X_train_new[:, 0], X_train_new[:, 1], c=np.argmax(
    y_train_new, axis=1), cmap=plt.cm.RdBu, edgecolors='k', label=''
    'Обучающая
    выборка')
plt.scatter(X_test_new[:, 0], X_test_new[:, 1], c=np.argmax(
    y_test_new, axis=1), cmap=plt.cm.RdBu, s=50, label='Тестовая
    выборка')
plt.scatter(X_user[:, 0], X_user[:, 1], c=np.argmax(y_pred, axis=1),
    cmap=plt.cm.cividis, marker='x', s=125, label='Дополнительные
    точки')
plt.title(f'Разделяющие поверхности')
plt.xlabel('Признак 1')
plt.ylabel('Признак 2')
plt.legend()
plt.show()

```

## 4 Исследовательская часть

### 4.1 Среда для тестирования

Для тестирования разработанного алгоритма применялась облачная платформа Google Colab, не требующая установки ПО на локальный компьютер.

### 4.2 Использование ReLu в качестве функции активации

Листинг 4.1 — Отчёт по результатам классификации

Classification Report:				
	precision	recall	f1-score	support
0	1.00	1.00	1.00	48
1	0.95	0.96	0.95	77
2	1.00	0.95	0.98	43
3	0.96	0.97	0.97	72
accuracy			0.97	240
macro avg	0.98	0.97	0.97	240
weighted avg	0.97	0.97	0.97	240
Additional Metrics:				
MCC:	0.9603			

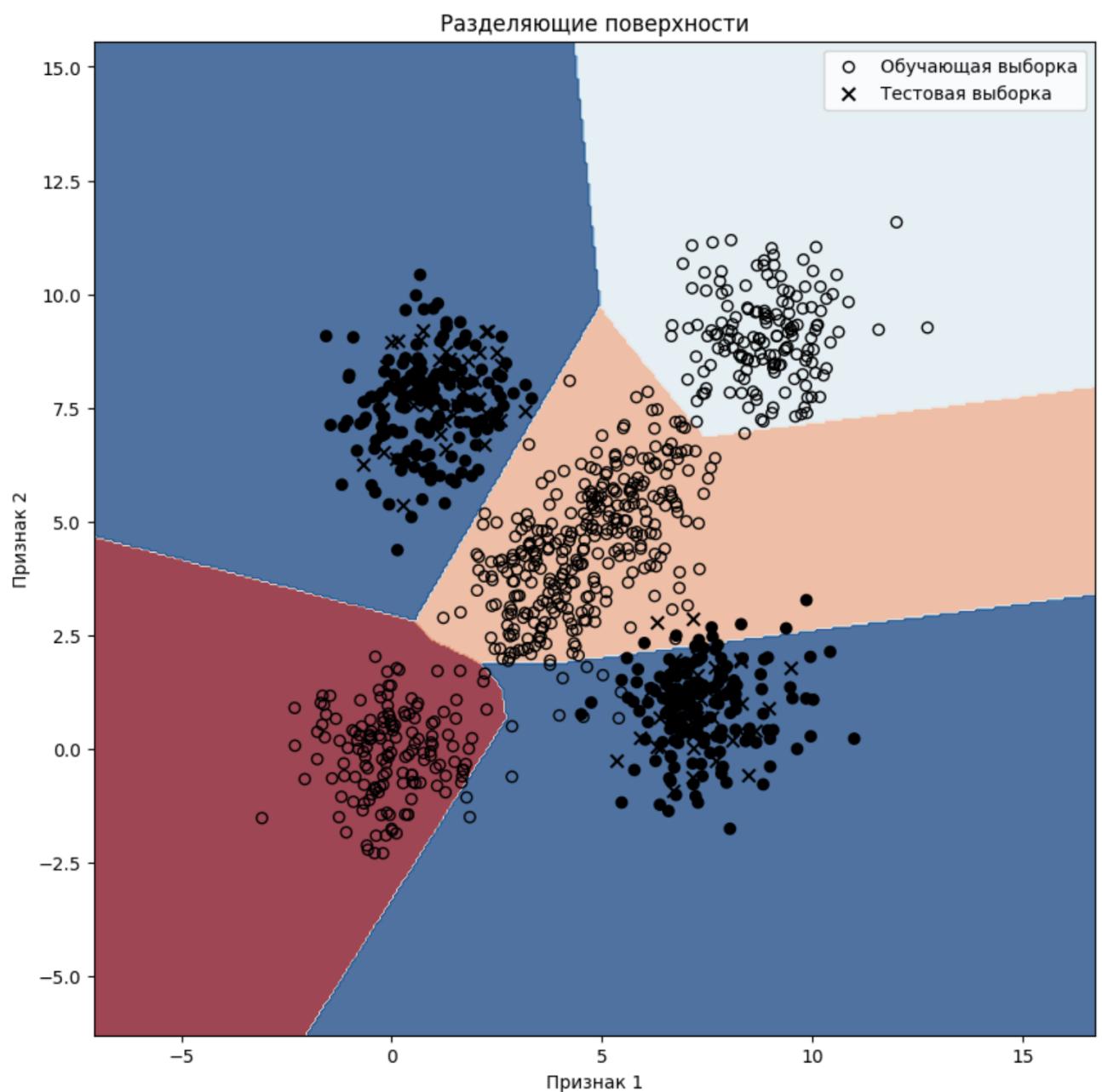


Рисунок 4.1 — Разделяющие поверхности

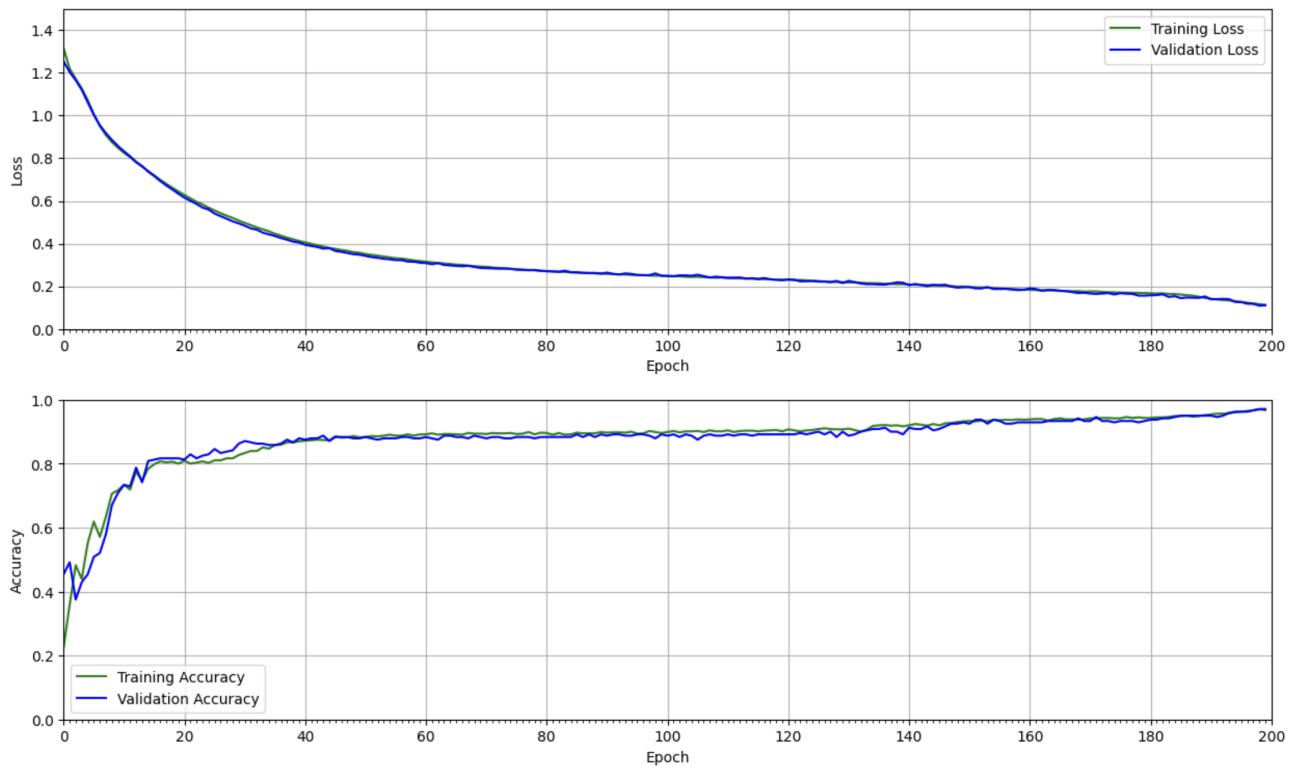


Рисунок 4.2 — Кривая обучения

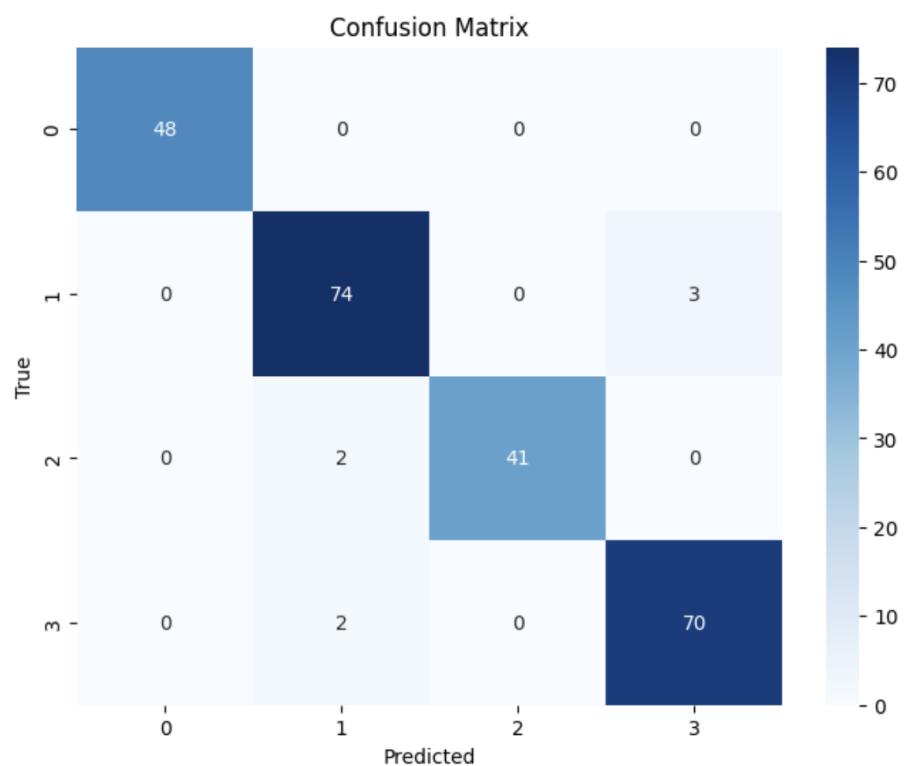


Рисунок 4.3 — Матрица ошибок

## 4.3 Использование сигмоиды ( $b = 1$ ) в качестве функции активации

Листинг 4.2 — Отчёт по результатам классификации

Classification Report:				
	precision	recall	f1-score	support
0	1.00	1.00	1.00	48
1	0.90	0.84	0.87	77
2	0.90	0.88	0.89	43
3	0.88	0.96	0.92	72
accuracy			0.92	240
macro avg	0.92	0.92	0.92	240
weighted avg	0.92	0.92	0.92	240
Additional Metrics:				
MCC:	0.8873			

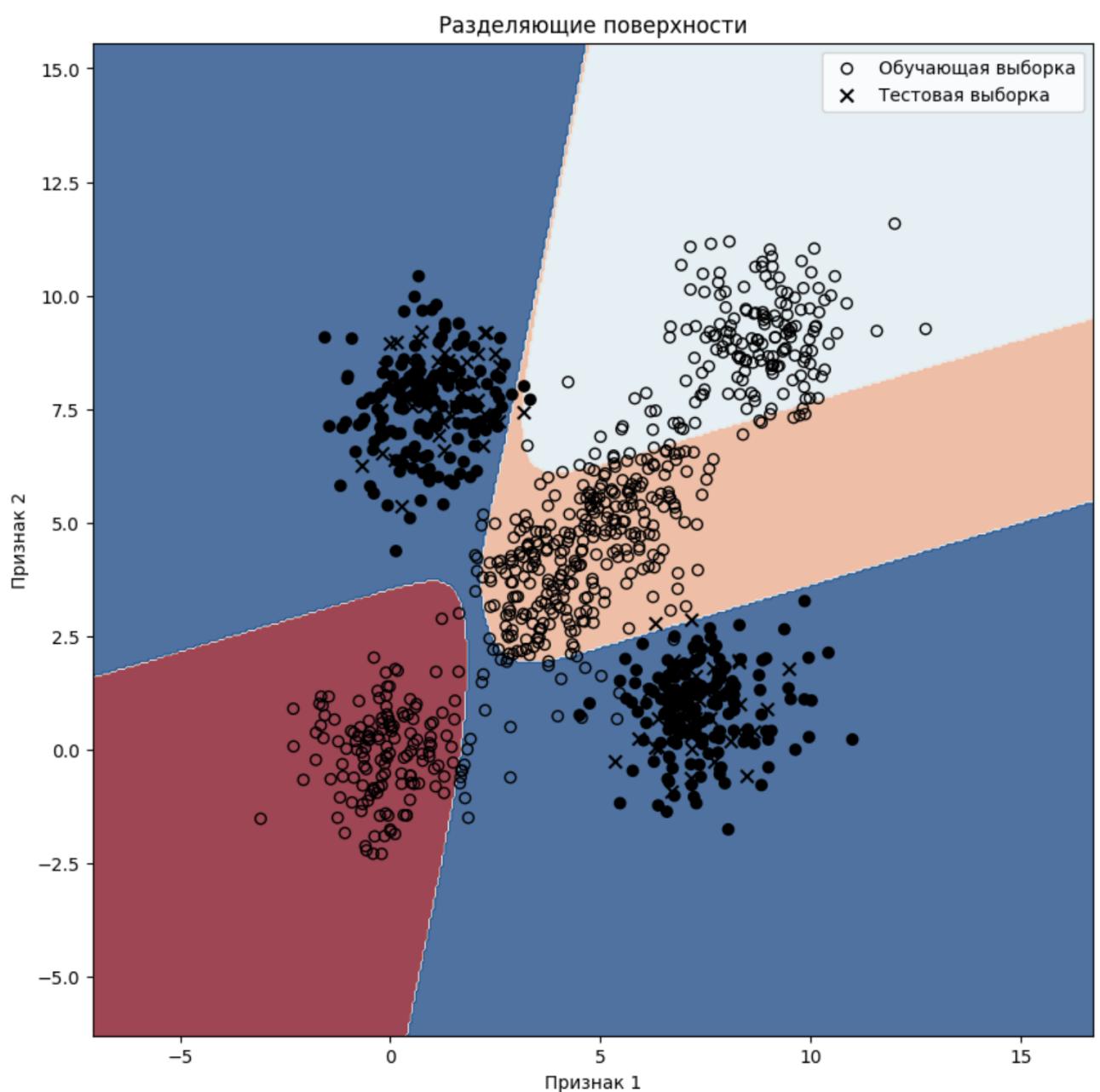


Рисунок 4.4 — Разделяющие поверхности

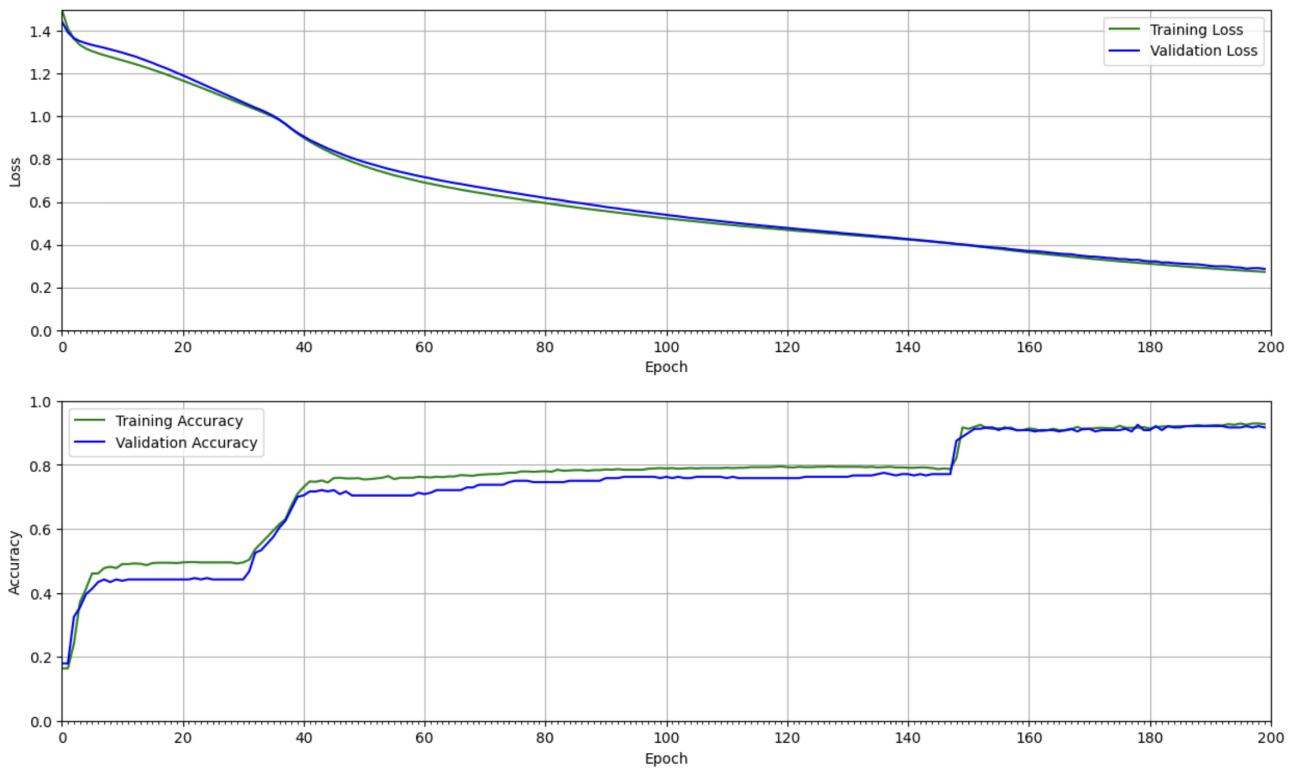


Рисунок 4.5 — Кривая обучения

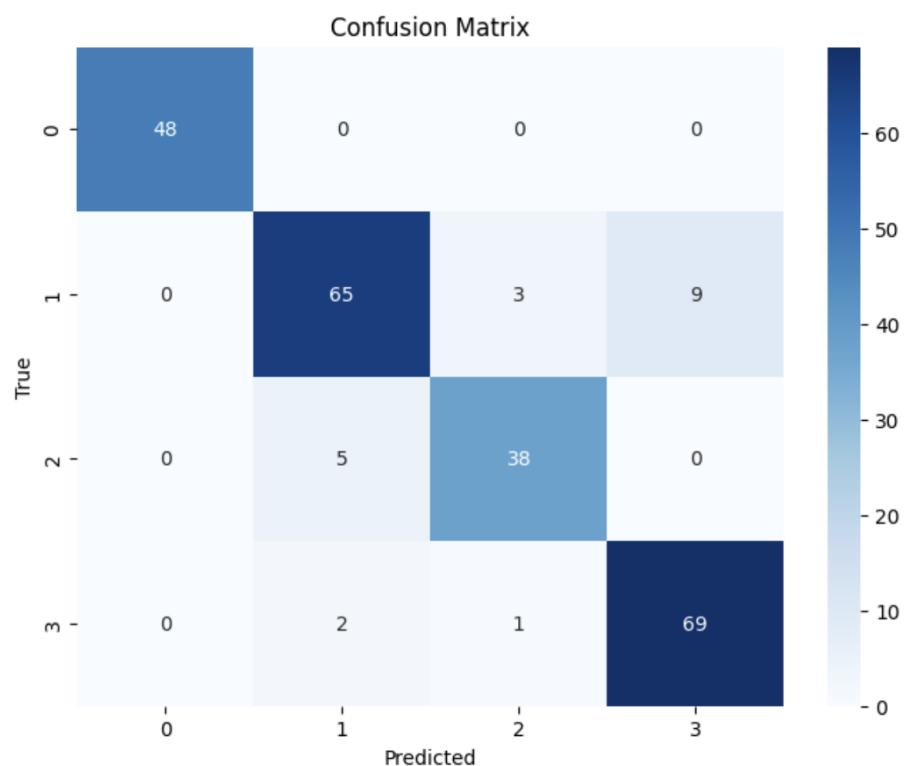


Рисунок 4.6 — Матрица ошибок

## 4.4 Использование сигмоиды ( $b = 5$ ) в качестве функции активации

Листинг 4.3 — Отчёт по результатам классификации

Classification Report:				
	precision	recall	f1-score	support
0	1.00	0.96	0.98	48
1	0.90	0.94	0.92	77
2	0.93	0.88	0.90	43
3	0.95	0.96	0.95	72
accuracy			0.94	240
macro avg	0.94	0.93	0.94	240
weighted avg	0.94	0.94	0.94	240
Additional Metrics:				
MCC:	0.9149			

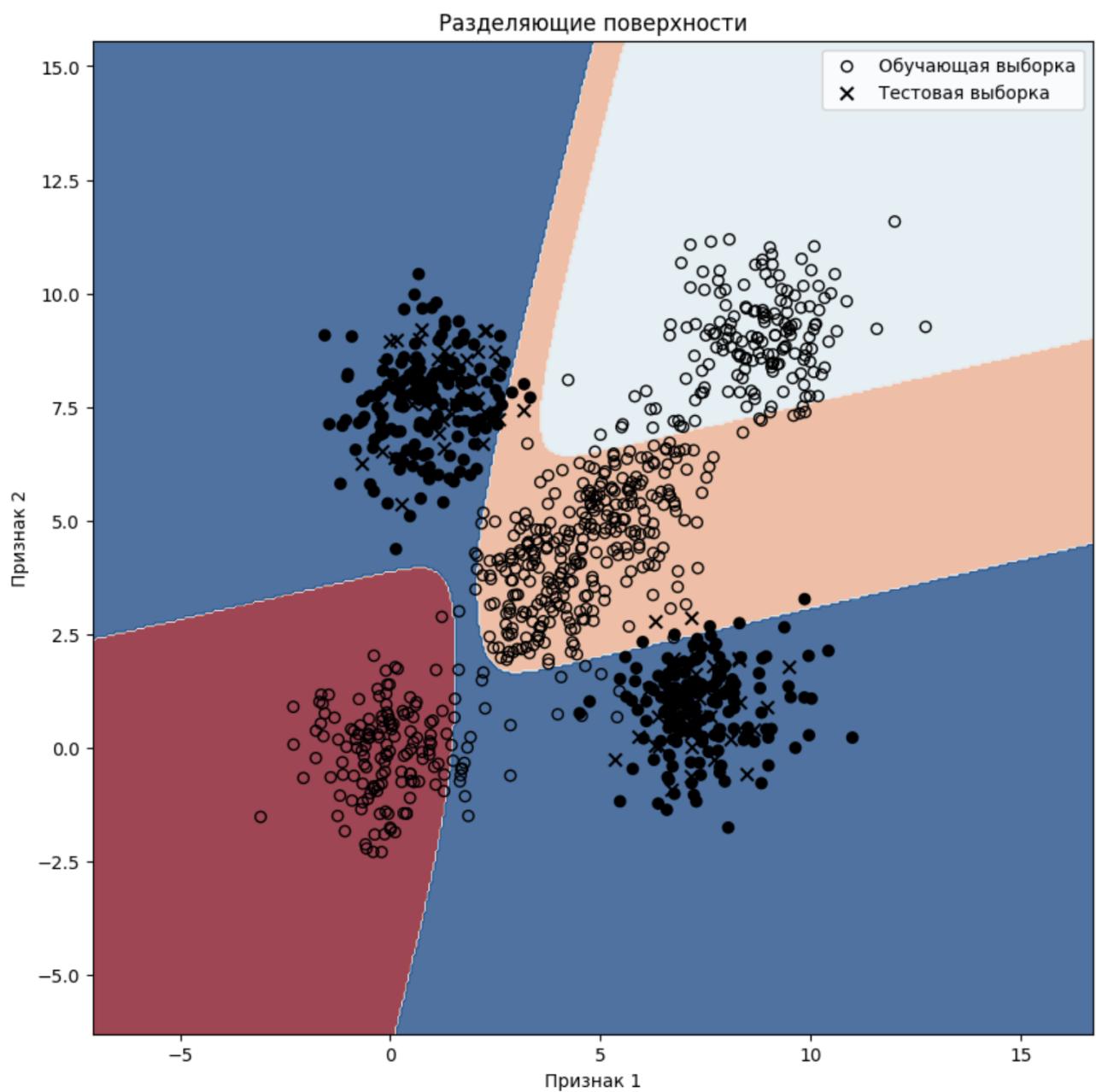


Рисунок 4.7 — Разделяющие поверхности

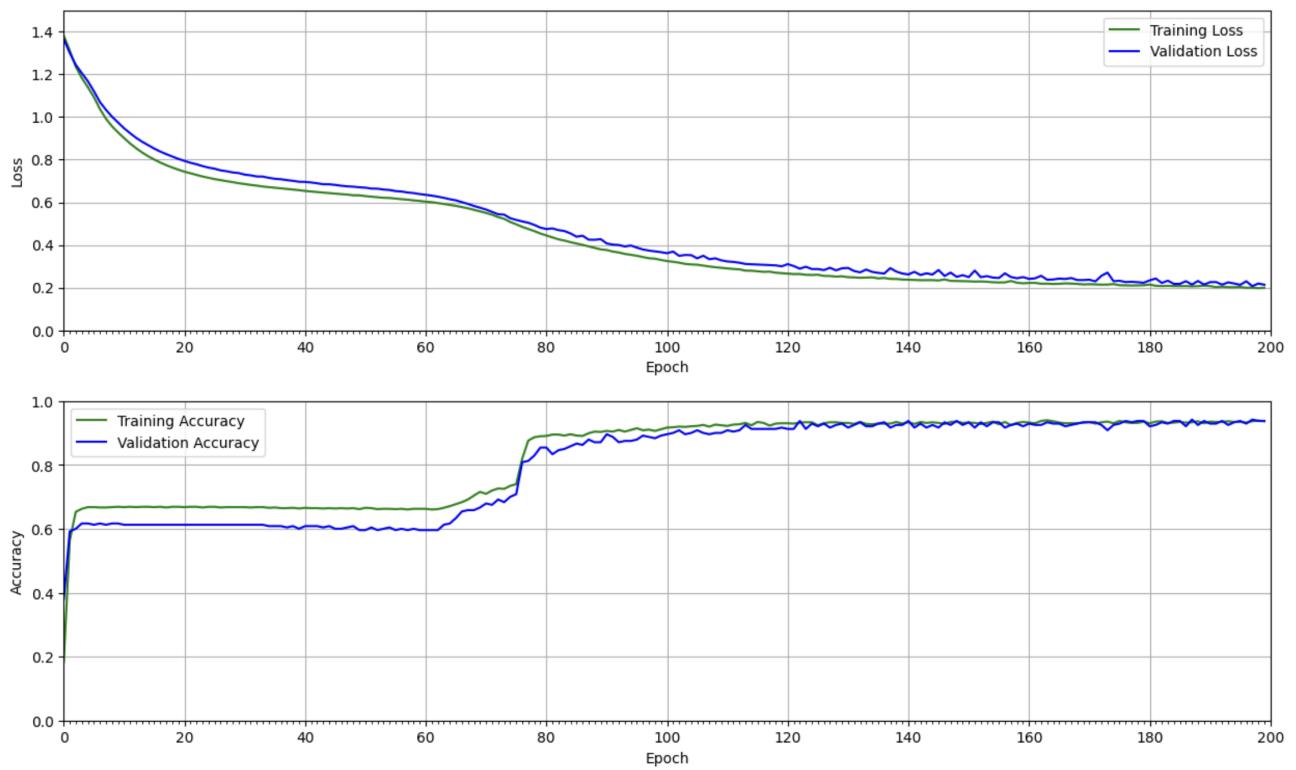


Рисунок 4.8 — Кривая обучения

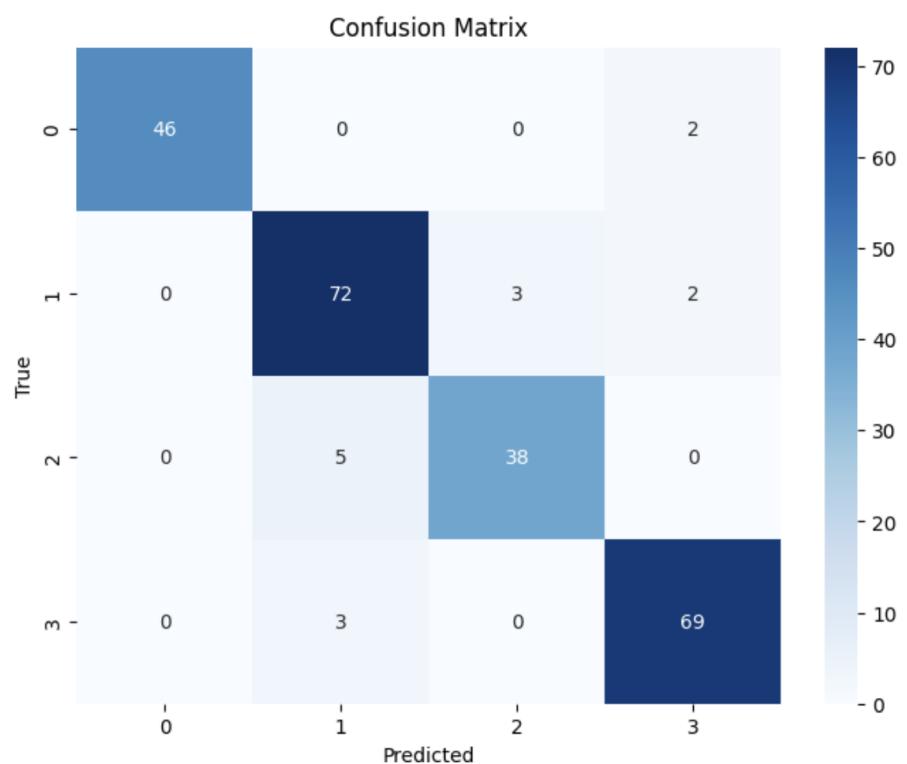


Рисунок 4.9 — Матрица ошибок

## 4.5 Использование сигмоиды ( $b = 10$ ) в качестве функции активации для построения разделяющих гиперплоскостей на основе весов нейронов скрытого слоя

Листинг 4.4 — Отчёт по результатам классификации

```
Classification Report:  
precision    recall    f1-score   support  
  
          0       1.00      1.00      1.00      47  
          1       0.97      0.98      0.98      63  
          2       1.00      0.98      0.99      43  
          3       0.99      0.99      0.99      87  
  
accuracy                           0.99      240  
macro avg       0.99      0.99      0.99      240  
weighted avg     0.99      0.99      0.99      240  
  
Additional Metrics:  
MCC: 0.9829
```

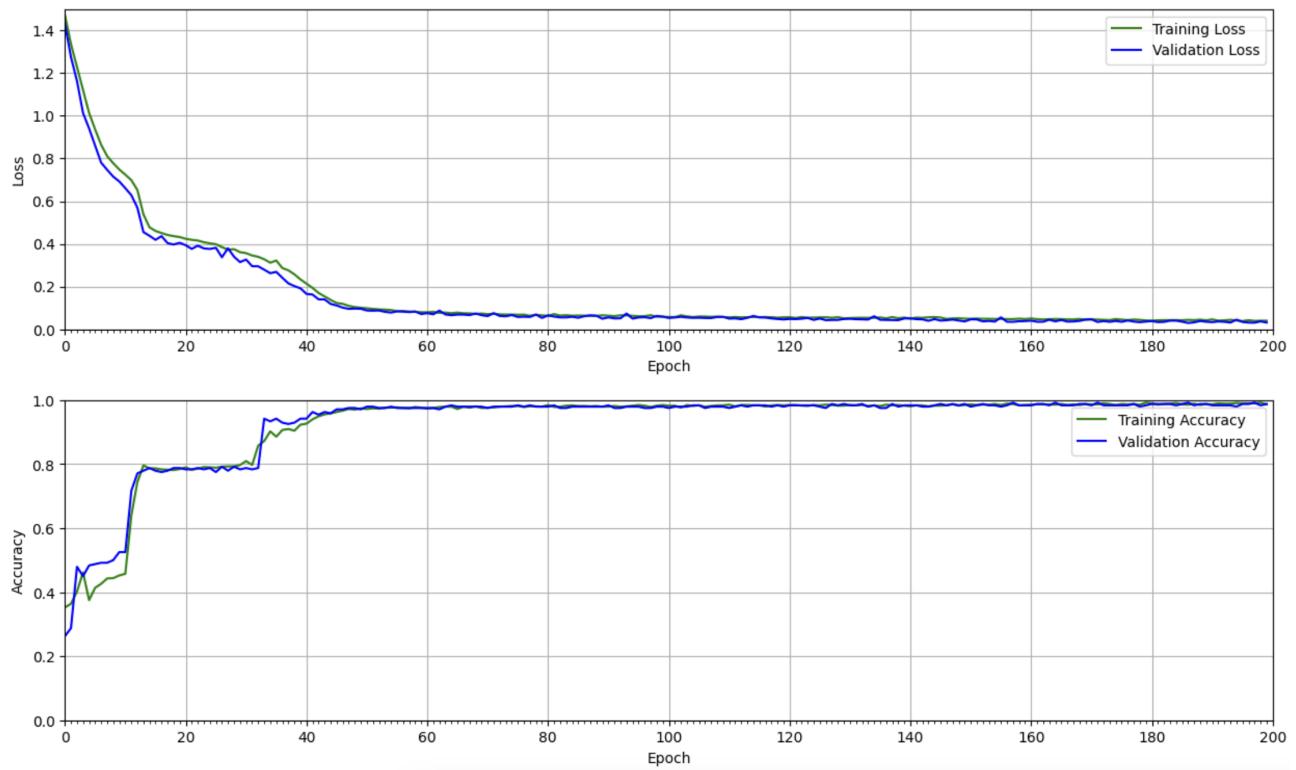


Рисунок 4.10 — Кривая обучения

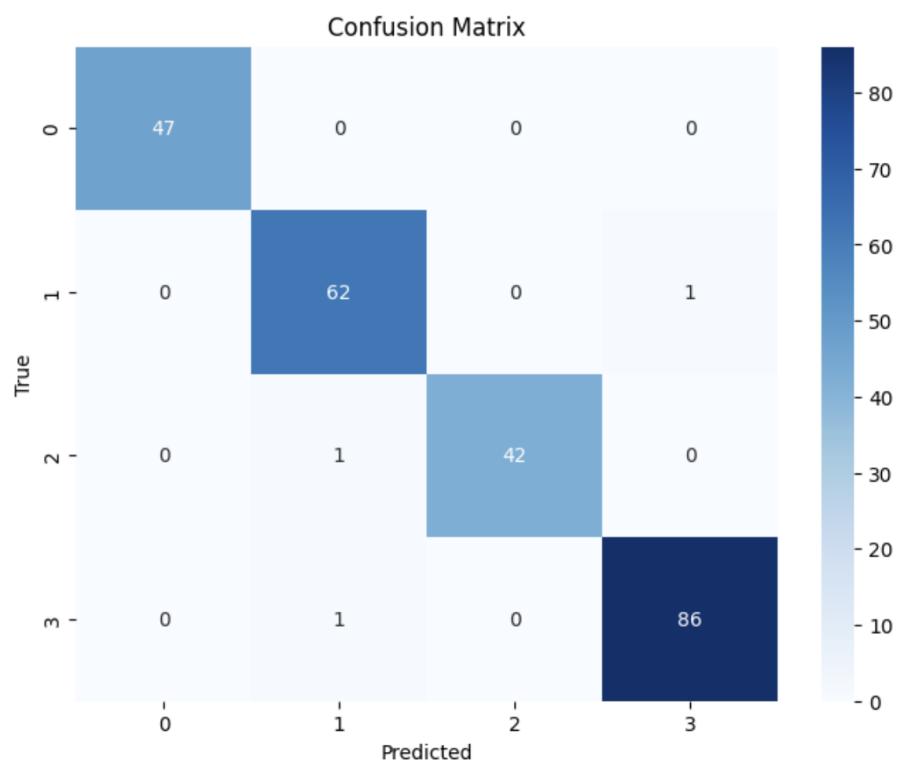


Рисунок 4.11 — Матрица ошибок

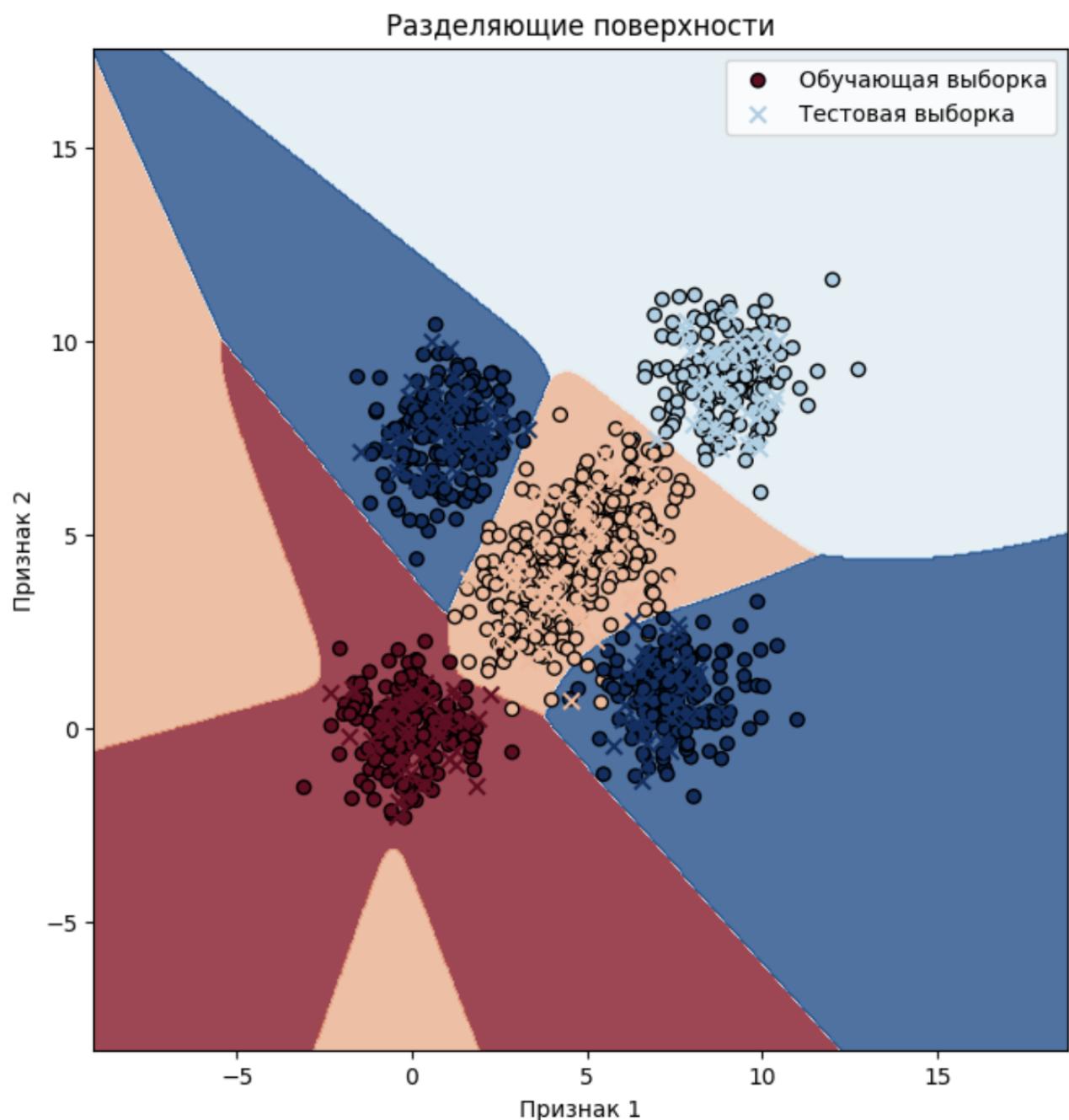


Рисунок 4.12 — Разделяющие поверхности

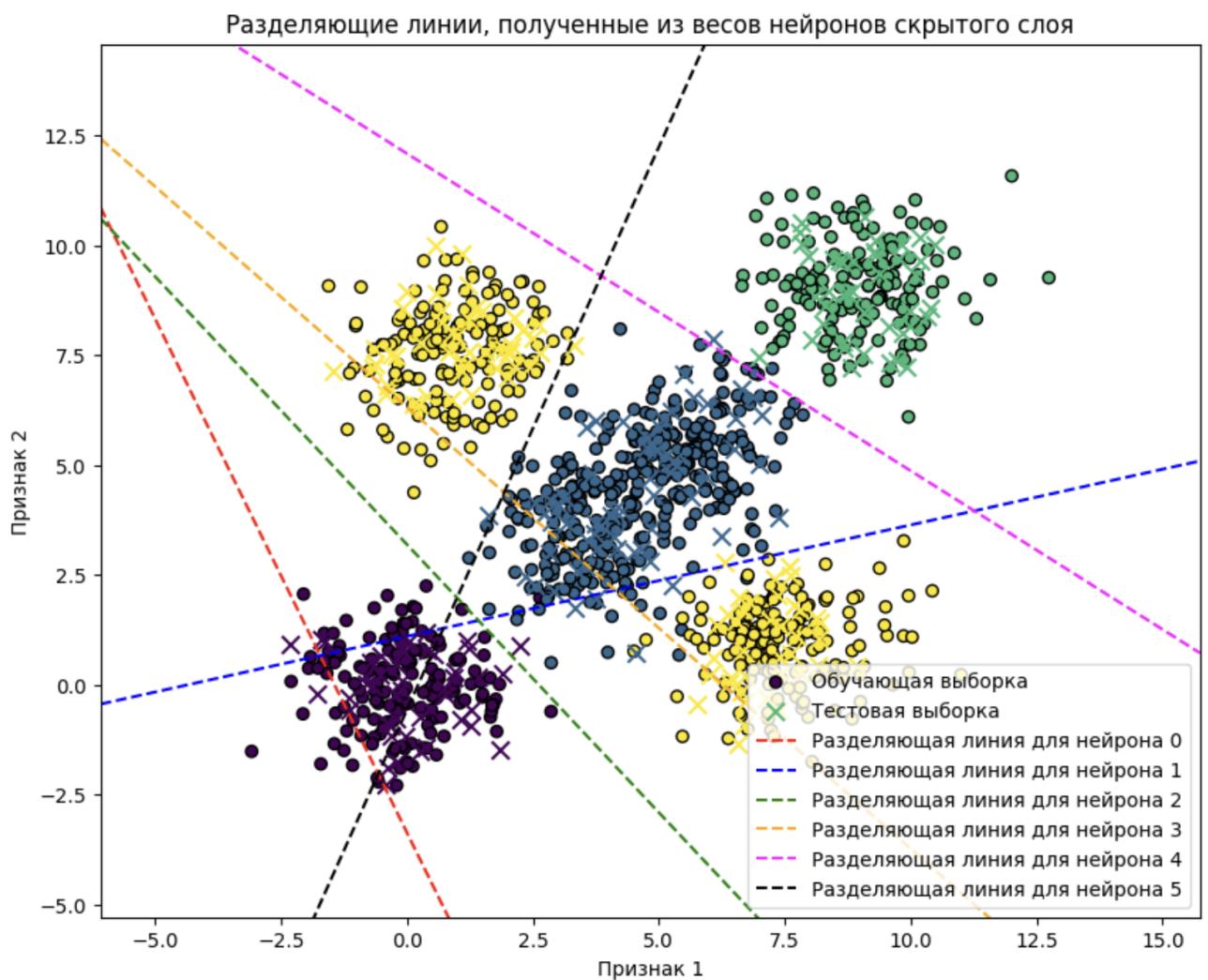


Рисунок 4.13 — Разделяющие гиперплоскости нейронов скрытого слоя

## 4.6 Добавление класса для отделения данных, далеко отстоящих от исходных классов

Листинг 4.5 — Отчёт по результатам классификации

Classification Report:				
	precision	recall	f1-score	support
0	0.92	0.87	0.89	39
1	0.99	1.00	0.99	84
2	0.95	0.98	0.96	41
3	0.85	0.98	0.91	95
4	0.94	0.82	0.88	125
accuracy			0.92	384
macro avg	0.93	0.93	0.93	384
weighted avg	0.93	0.92	0.92	384
Additional Metrics:				
MCC:	0.8998			

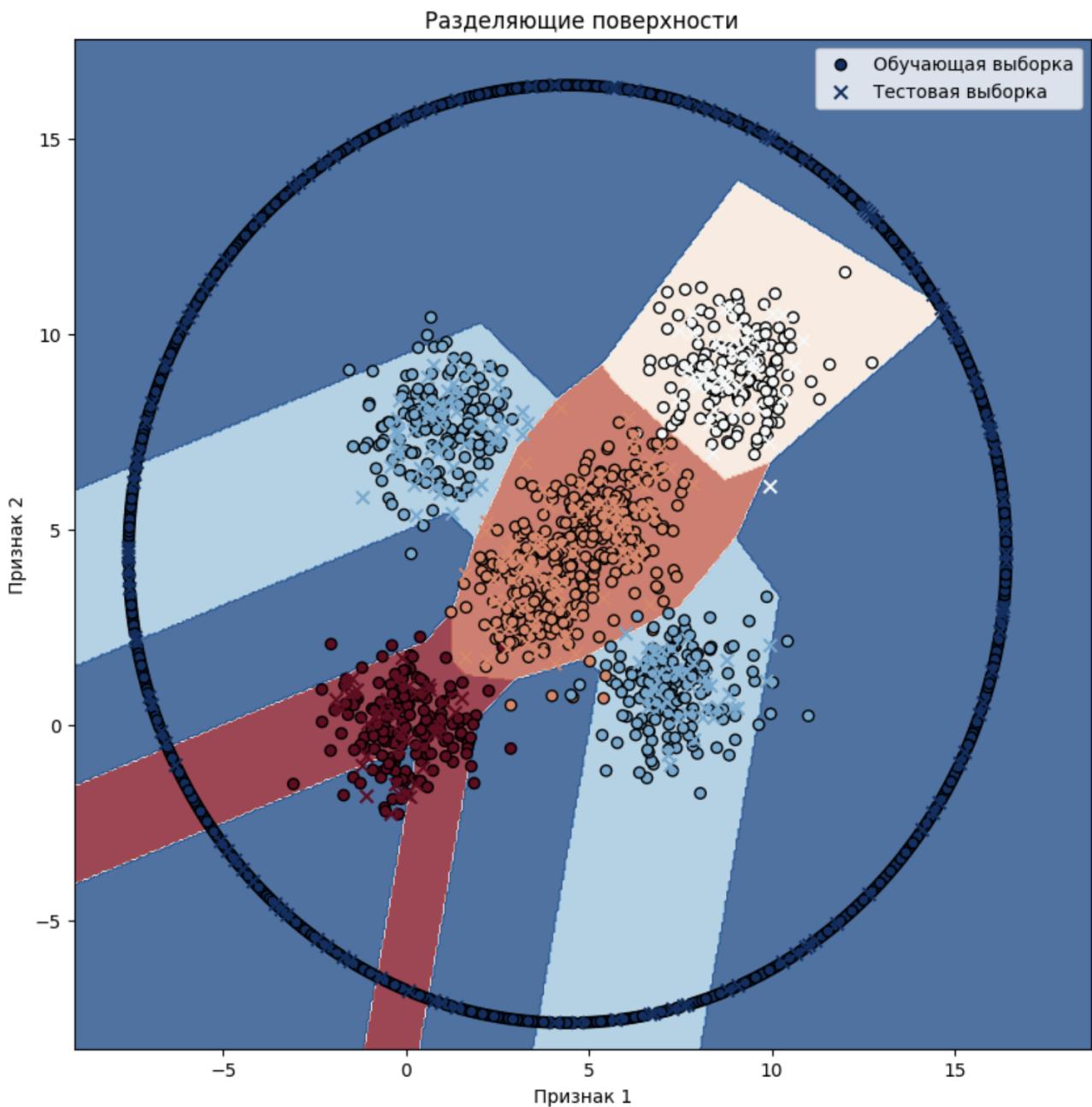


Рисунок 4.14 — Разделяющие поверхности

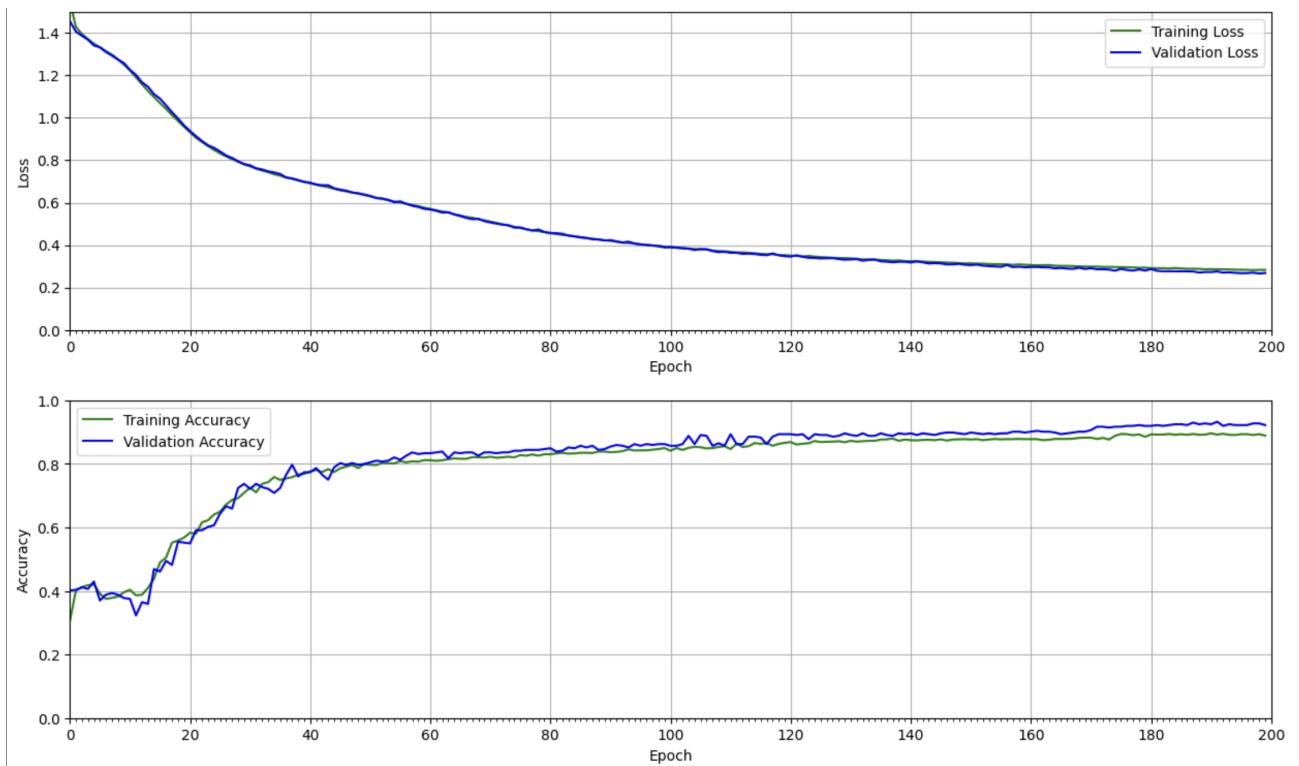


Рисунок 4.15 — Кривая обучения

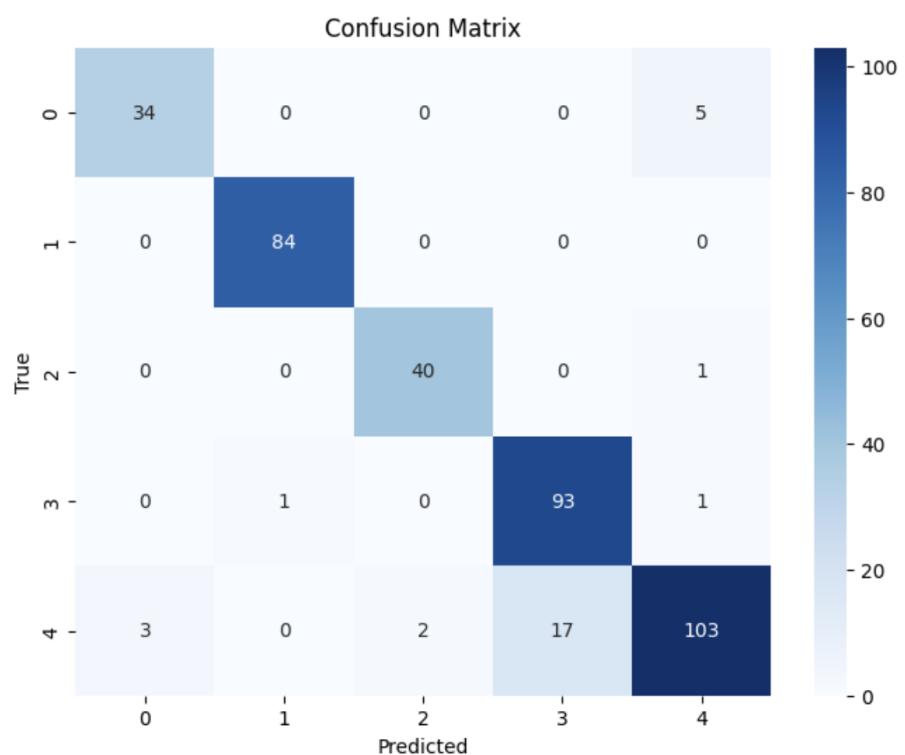


Рисунок 4.16 — Матрица ошибок

# ЗАКЛЮЧЕНИЕ

В рамках лабораторной работы было проведено изучение нейросетевого подхода на примере построения классификатора на базе многослойного персептрона.

1. Осуществлена генерация исходных данных.
2. Для сгенерированного датасета построен классификатор на базе многослойного персептрона.
3. Дано обоснование выбора числа слоев и нейронов в каждом слое.
4. Проведено сравнение работы нейросети в зависимости от выбранной функции активации (сигмоида с разными значениями параметра крутизны ( $b=1, b=100$ ); ReLU ).
5. Обоснован момент остановки процесса обучения.
6. Оценены точность, полнота, F-мера.
7. Построена матрицу ошибок.
8. Предусмотрена дополнительную возможность ввода пользователем новых, не входящих в сгенерированный датасет данных.

Для построенного классификатора максимальное значение метрики MCC составила 0.9829.