МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению лабораторных работ

«Технологии и методы распознавания образов»

Направление подготовки 09.03.02 «Информационные системы и технологии»
Направленность (профиль) «Прикладное программирование и интернет-технологии»
Квалификация выпускника: бакалавр

Ставрополь, 2026

Введение

Целью освоения дисциплины «Технологии и методы распознавания образов» является получение знаний, умений и навыков по использованию методов распознавания образов при проведении научных исследований и решении практических задач, формирование на их основе компетенций будущего бакалаврапо направлению подготовки 09.03.02 «Информационные системы и технологии».

Задачами дисциплины являются:

1) изучение основных направлений в области распознавания образов;

2) обучение основным принципам и технологиям разработки инструментов распознавания образов.

3) понимание принципов работы и методов распознавания образов;

4) развитие навыков анализа и обработки данных;

5) умение использовать инструменты и технологии машинного обучения и распознавания образов для решения задач.

Место дисциплины в структуре образовательной программы. Дисциплина «Технологии и методы распознавания образов» относится к базовой части Блока 1(Б1.В.ДВ.06.01). Ее освоение происходит в 8 семестре.

В результате освоения дисциплины обучающийся должен овладеть следующими компетенциями:

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
ПК-11 способен разрабатывать модели объектов профессиональной деятельности с использованием инструментальных средств компьютерного моделирования и обработки больших данных	ИД-1$_{ПК-11}$ Использует инструментальные средства компьютерного моделирования и обработки больших данных	Результаты выбора моделей объектов профессиональной деятельности являются верными. Представленный документ не содержит ошибок. Решены все частные задачи в полном объеме.
	ИД-2$_{ПК-11}$ Разрабатывает модели объектов профессиональной деятельности с использованием инструментальных средств компьютерного моделирования	Результаты анализа методов, технологий и средств разработки моделей распознавания образов с использованием инструментальных средств компьютерного моделирования являются верными. Представленный документ не содержит ошибок. Решены все частные задачи в полном объеме.

Лабораторная работа № 1

Создание нейронной сети. Классификация изображений.

Цель и содержание работы: научиться обучению модели нейронной сети для классификации изображений одежды, на примере набора данных Fashion MNIST.

Формируемые компетенции:, ПК-11

Теоретическое обоснование

Одним из наиболее распространенных применений TensorFlow и Keras является распознавание и классификация изображений.

TensorFlow - это библиотека с открытым исходным кодом, созданная для Python командой Google Brain. TensorFlow компилирует множество различных алгоритмов и моделей, позволяя пользователю реализовать глубокие нейронные сети для использования в таких задачах, как распознавание и классификация изображений, а также обработка естественного языка. TensorFlow - это мощный фреймворк, который функционирует путем реализации ряда узлов обработки, каждый из которых представляет математическую операцию, а весь ряд узлов называется «графом».

Говоря о Keras, это высокоуровневый API (интерфейс прикладного программирования), который может использовать функции TensorFlow (а также другие библиотеки ML, такие, как Theano). Keras был разработан с удобством и модульностью в качестве руководящих принципов. С практической точки зрения Keras позволяет реализовать множество мощных, но зачастую сложных функций TensorFlow максимально просто, к тому же он настроен для работы с Python без каких-либо серьезных изменений или настроек.

Распознавание изображения относится к задаче ввода изображения в нейронную сеть и присвоения какой-либо метки для этого изображения. Метка, которую выводит сеть, будет соответствовать заранее определенному классу. Может быть присвоено как сразу несколько классов, так и только один. Если существует всего только один класс, обычно применяется термин «распознавание», тогда как задача распознавания нескольких классов часто

называется «классификацией».

Подмножество классификаций изображений - является уже определением объектов, когда определенные экземпляры объектов идентифицируются как принадлежащие к определенному классу, например, животные, автомобили или люди.

Ярким примером такой классификации является решение самой распространённой капчи — ReCaptcha v2 от Google, где из набора картинок необходимо выбрать только те, которые принадлежат к указанному в описании классу.

Аппаратура и материалы. 64-разрядный (x64) персональный компьютер, процессор с тактовой частотой 1 ГГц и выше, оперативная память 1 Гб и выше, свободное дисковое пространство не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система Windows 7 и выше, Matlab (R2013) и выше.

Указание по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

Методика и порядок выполнения работы

Для работы предлагается использовать Google Colaboratory — это бесплатная интерактивная облачная среда для работы с кодом от Google. Сервис нужен, чтобы писать код в jupyter notebook. Он функционирует по принципу облака, что позволяет работать над одним проектом целой командой.

Программа предоставляет доступ к графическим процессорам GPU и TPU, благодаря которым можно развивать приложения на основе нейросетей.

Как начать работу в Google Colab.

Для начала убедитесь, что вы вошли в аккаунт, а затем создайте отдельную папку под готовые проекты. После запуска блокнота нажмите «Создать» → «Ещё» → Google Colaboratory. Для переименования блокнота щёлкните на имя файла.

Далее определитесь с процессором. Если будете работать на слишком мощном процессоре, то вы можете вылететь из блокнота. Чтобы попасть в настройки процессора, выберите вкладку «Среда выполнения» и команду «Сменить среду управления».

В Colab много встроенных библиотек Python: Pandas, NumPy, Scikit-learn. Для просмотра полного списка, введите команду !pip list.

Библиотека Pandas применяется для анализа и обработки табличных данных. То есть она как Excel, но работает с большей мощностью. Pandas позволяет работать с данными объемом в миллионы строк.

Чтобы импортировать сторонние библиотеки, используйте команды: !pip install имя_библиотеки
import имя_библиотеки.

Чтобы очистить строку, щелкните на значок крестика в левой части окна.

Также при желании можно загрузить tensorflow на свой компьютер, делается это с помощью команды pip install tensorflow в командной строке анаконды. Но с большой вероятностью у вас могут возникнуть трудности с дальнейшим импортированием этих модулей в ваш проект. И если такие ошибки возникнут, вот по этой ссылке вы можете посмотреть самые типичные из них и как можно их устранить https://www.tensorflow.org/install/errors?hl=ru

Поэтому чтобы избежать этих трудностей с установкой на локальном компьютере, будем работать в Google Colaboratory, тут не должно возникнуть ни каких проблем, кроме того эта облачная среда дает вам возможность использовать больше оперативной памяти, чем возможно имеется на вашем компьютере.

Для начала, необходимо подключить нужные библиотеки и модули. Подключим numpy и np для работы с массивами и визуализацией рисунков. Далее импортируем модуль **sequential** это модель нейронных сетей, где слои идут друг за другом. После подключаем тип слоев **Dense**, который означает, что слои будут полносвязными (Полносвязный слой — слой, выходные нейроны которого связаны со всеми входными нейронами. Нейроном в данном случае называется математическая модель искуственного нейрона, основанная на представлении о биологическом нейроне, однако в действительности не имеющая с ним практически

ничего общего). А также подключаем различные утилиты, которые помогут перевести данные в подходящий для keras формат.

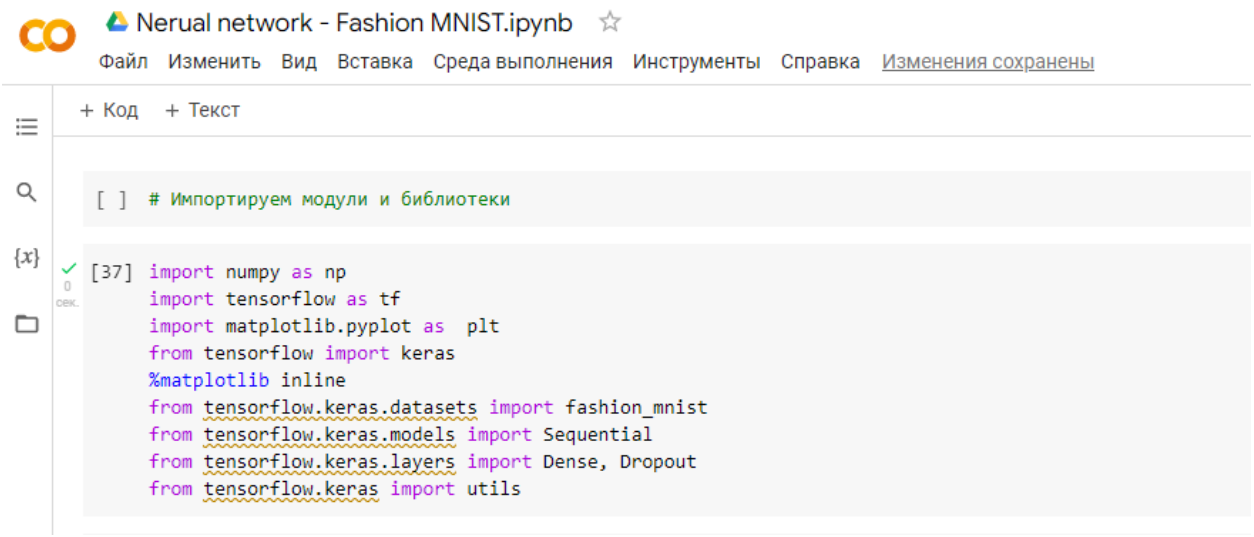


Рисунок 1 – Импорт модулей и библиотек

Модуль для работы fashion mnist уже присутствует keras, т.к он является одним из таких популярных наборов данных, на которых обучаются нейронным сетям все датасайнтисты.

Набор данных Fashion MNIST, который содержит 70 000 изображений в 10 категориях. На изображениях показаны отдельные предметы одежды с низким разрешением (28 на 28 пикселей):

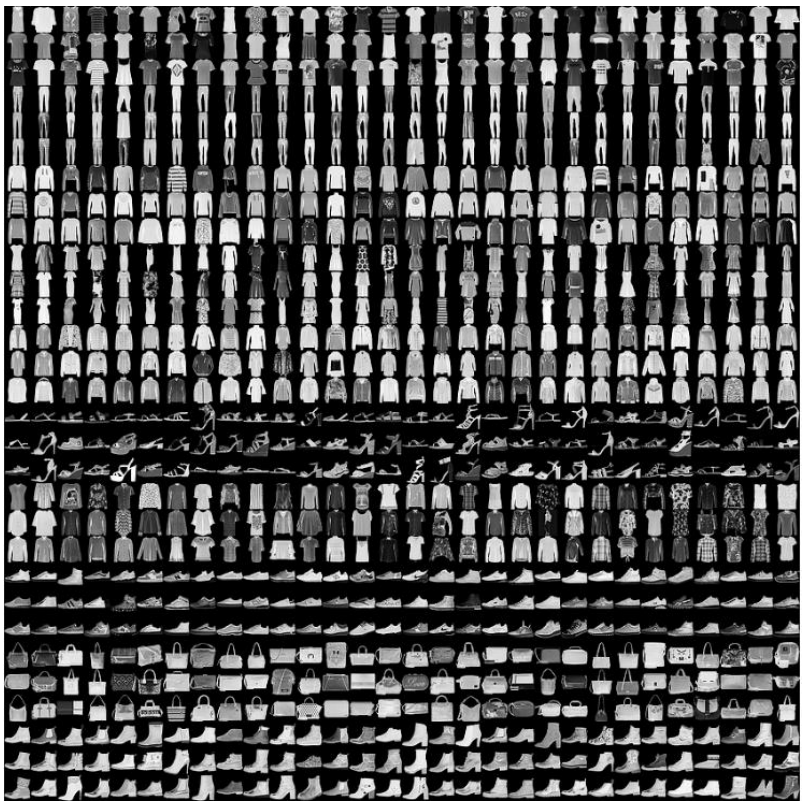


Рисунок 2 – Набор данных Fashion MNIST

Будем использовать 60 000 изображений для обучения сети и 10 000 изображений, чтобы оценить, насколько точно сеть научилась классифицировать изображения. Вы можете получить доступ к Fashion MNIST непосредственно из TensorFlow, просто импортировав и загрузив данные:

Рисунок 3 – Загрузка набора данных

Загрузка набора данных возвращает четыре массива:

Массивы x_train и y_train — это данные, которые использует модель для обучения

Массивы x_test и y_test используются для тестирования модели

Часть x это сами изображения, а часть y это ответы или метки.

Изображения представляют собой NumPy массивы 28x28, значения пикселей которых находятся в диапазоне от 0 до 255. Метки (labels) представляют собой массив целых чисел от 0 до 9. Они соответствуют классу одежды:

Метка	Класс
0	T-shirt (Футболка)
1	Trouser (Брюки)
2	Pullover (Свитер)
3	Dress (Платье)
4	Coat (Пальто)
5	Sandal (Сандали)
6	Shirt (Рубашка)
7	Sneaker (Кроссовки)
8	Bag (Сумка)
9	Ankle boot (Ботильоны)

Имена классов не включены в набор данных, поэтому прописываем самостоятельно:

Nerual network - Fashion MNIST.ipynb

Файл Изменить Вид Вставка Среда выполнения Инструменты Справка Изменения сохранены

+ Код + Текст

```
[ ] # Импортируем модули и библиотеки

import numpy as np
import tensorflow as tf
import matplotlib.pyplot as  plt
from tensorflow import keras
%matplotlib inline
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras import utils

[13] # делим наш датасет  на обучающую и тестовую выборку

[14] (x_train, y_train),(x_test, y_test) = fashion_mnist.load_data()

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step

[15] class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

Рисунок 4 – Добавление имен классов

Далее нужно произвести предварительную обработку данных перед тем как создать нейронную сеть. Но сначала посмотрим, как выглядят изображения.

```
[16] # Предварительная обработк данных

[17] # посмотрим как выглядят изображения

[18] plt.figure()
     plt.imshow(x_train[25])
     plt.colorbar()
     plt.grid(False)
```

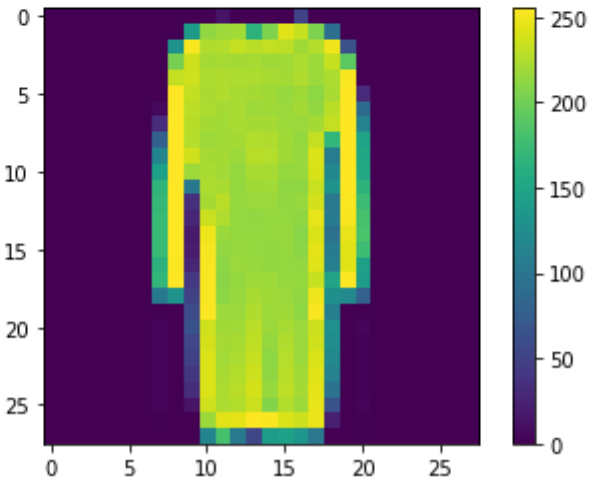


Рисунок 5 – Предварительная обработка данных

В квадратные скобки вставляем индекс от 0 до 59999, т.к в обучающей выборке 60000 рисунков. Сбоку на рисунке можно увидеть значения интенсивностей пикселей от 0 до 255, т.е если пиксель максимально темный, он имеет значение 0, а если максимально светлый значение 255.

Далее выполним нормализацию данных. Для улучшения алгоритма оптимизации, который используется в обучении нейронных сетей, делим интенсивность каждого пикселя изображения на 255, чтобы данные на входе в нейронную сеть находились в диапазоне от 0 до 1.

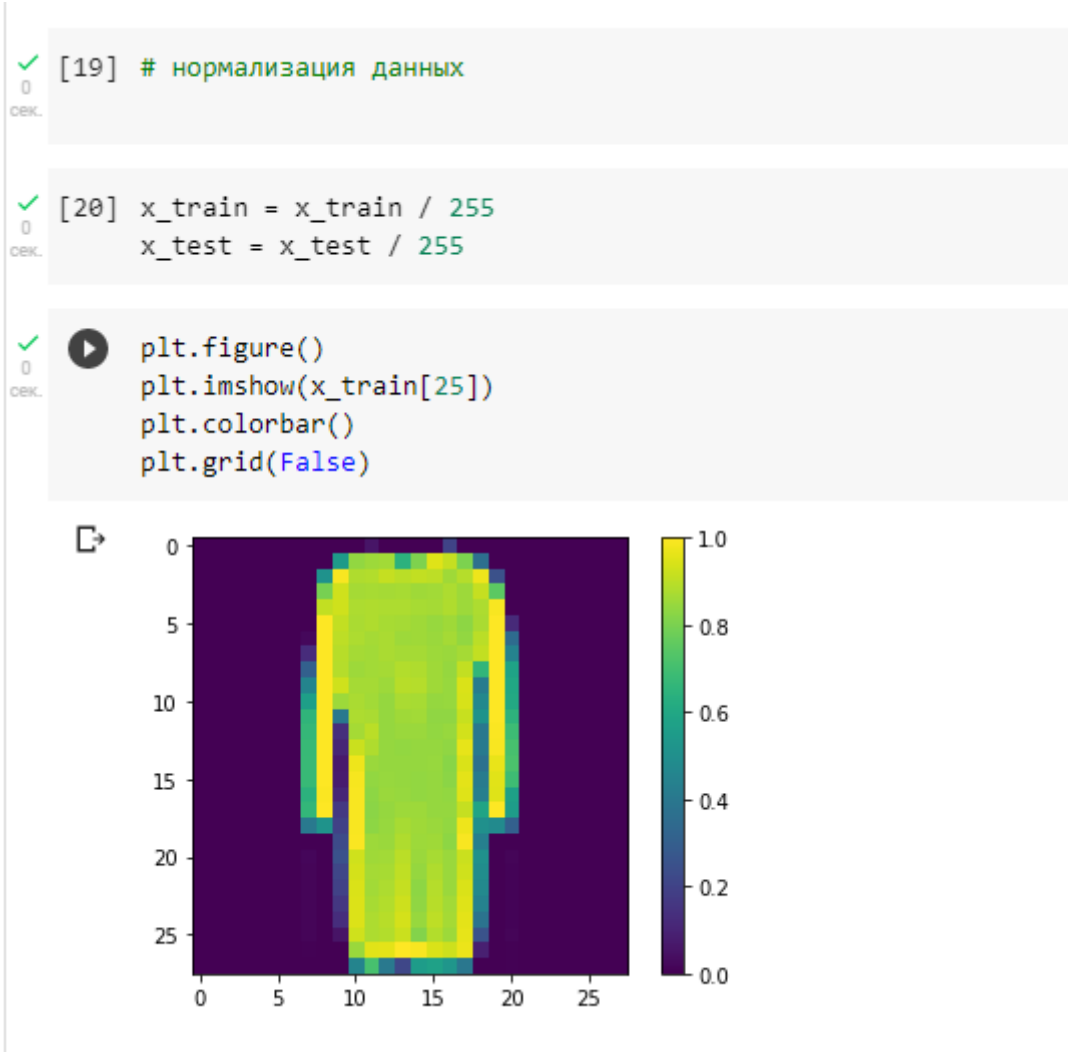


Рисунок 6 – Нормализация данных

Просмотрим несколько изображений на одном экране (25 рисунков по 5 в каждой строке и имя класса), для этого:

```
[22] # посмотрим несколько изображений
```

```
[23] plt.figure(figsize=(10,10))
     for i in range (25):
       plt.subplot(5,5,i+1)
       plt.xticks([])
       plt.yticks([])
       plt.imshow(x_train[i])
       plt.xlabel(class_names[y_train[i]])
```

Рисунок 7 – Просмотр нескольких изображений

Данные подготовлены и далее создаем модель нейронной сети. В нейронных сетях основным строительным блоком является слой, и основная часть глубокого обучения состоит в объединении простых слоев. В keras нейронные сети называются моделями. Будем использовать модель типа sequential, это означает, что слои будут идти последовательно. Создаем последовательную модель. Первый слой сети Flatten преобразует формат изображений из 2-мерного массива в 1-мерный массив, т.е теперь изображение будет поступать в нейрон как строка в 784 пикселя.

После идут два полноценных слоя: первый слой входной полносвязный, здесь определяем сколько нейронов будет в этом слое (в ходе экспериментов установлено, что одним из самых удачных по предсказаниям является 128 нейронов, но можно указать также и 512 нейронов или 800). В каждый нейрон будут поступать все 784 пикселя изображения.

Указываем функцию активации, в нашем случае для входного слоя указываем ‘relu’, она показывает хорошую эффективность в подобных нейронных сетях.

Далее 3-й слой выходной полносвязный, в нем будет 10 нейронов по количеству классов, в качестве функции активации будет функция "softmax", благодаря которой 2-й слой будет возвращать массив из 10 вероятностных оценок сумма которых равна 1.

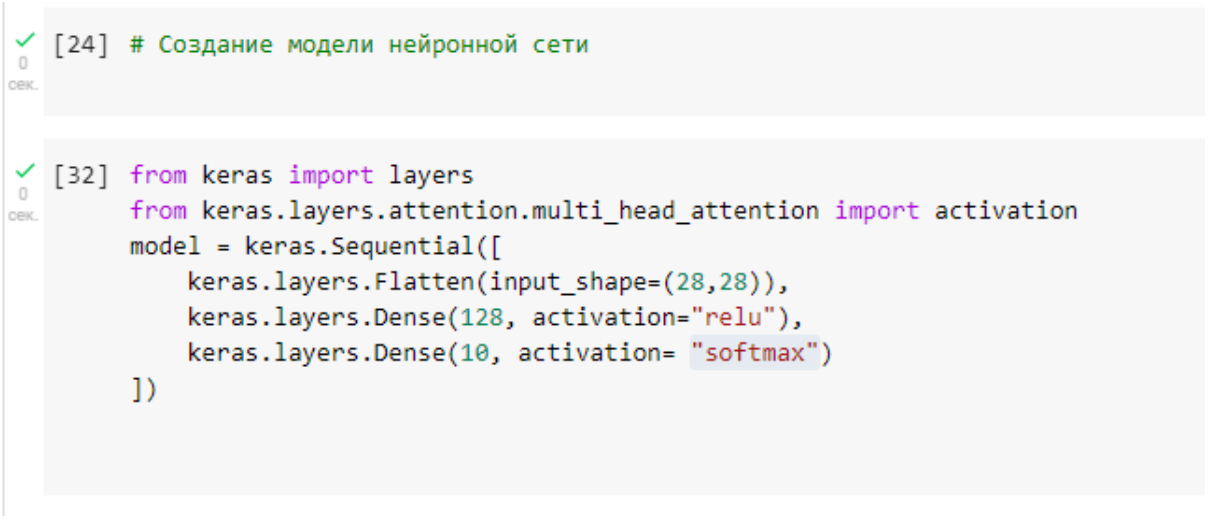


Рисунок 8 – Создание модели нейронной сети

Оптимизатор в нашем случае используем 'adam', также можно использовать также SGD (Стохастический градиентный спуск- оптимизационный алгоритм, отличающийся от обычного градиентного спуска тем, что градиент оптимизируемой функции считается на каждом шаге не как сумма градиентов от каждого элемента выборки, а как градиент от одного, случайно выбранного элемента)

Прежде чем модель будет готова к обучению, ей потребуется еще несколько настроек. Они добавляются во время этапа компиляции модели:

Категориальная перекрестная энтропия (В теории информации перекрёстная энтропия между двумя распределениями вероятностей измеряет среднее число бит, необходимых для опознания события из набора возможностей, если используемая

схема кодирования базируется на заданном распределении вероятностей, вместо «истинного» распределения)

Доля правильных ответов

Loss function (функция ошибки) — измеряет насколько точная модель во время обучения.

Optimizer (оптимизатор) — это то, как модель обновляется на основе данных, которые она видит, и функции потери.

Metrics (метрики) — используется для контроля за этапами обучения и тестирования.

```
[33] # Компиляция модели

     model.compile(optimizer = 'adam',
                   loss = 'sparse_categorical_crossentropy',
                   metrics=['accuracy'])

[43] model.summary()

     Model: "sequential"
     _________________________________________________________________
      Layer (type)                Output Shape              Param #
     =================================================================
      flatten (Flatten)           (None, 784)               0

      dense (Dense)               (None, 128)               100480

      dense_1 (Dense)             (None, 10)                1290

     =================================================================
     Total params: 101,770
     Trainable params: 101,770
     Non-trainable params: 0
     _________________________________________________________________
```

Рисунок 9 – Настройка параметров

Далее приступим к обучению нейронной сети. Обучение выполняется с помощью функции fit, т.к в рамках данной задачи выполняется обучение с учителем, передаем значения с x_train, y_train. Далее указываем параметры – количество эпох. Одна эпоха — это когда весь набор проходит через нейронную сеть один раз. Количество эпох будет разниться в зависимости от наборов данных. В конце каждой строки эпохи указывается функция ошибки.

```
[40] # обучение модели

model.fit(x_train, y_train, epochs=10)

Epoch 1/10
1875/1875 [==============================] - 9s 4ms/step - loss: 0.5008 - accuracy: 0.8232
Epoch 2/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3736 - accuracy: 0.8647
Epoch 3/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.3374 - accuracy: 0.8772
Epoch 4/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3127 - accuracy: 0.8857
Epoch 5/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2954 - accuracy: 0.8913
Epoch 6/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2816 - accuracy: 0.8958
Epoch 7/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2696 - accuracy: 0.9002
Epoch 8/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2571 - accuracy: 0.9049
Epoch 9/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2464 - accuracy: 0.9074
Epoch 10/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2401 - accuracy: 0.9096
<keras.callbacks.History at 0x7fb448db7280>
```

Рисунок 10 – Обучение модели

По мере обучения нейронной сети, т.е с каждой эпохой значение ошибки снижается, а точность повышается. При моделировании модели отображаются показатели потерь (loss) и точности (acc). Эта модель достигает точности около 0,9 (или 90%) по данным обучения.

Далее проверим точность на тестовой выборке, это новые для нейронной сети изображения (10000).

```
[48] # проверка точности предсказания

[49] test_loss, test_acc = model.evaluate(x_test, y_test)
     print('Test accuracy:', test_acc)

313/313 [==============================] - 1s 2ms/step - loss: 0.3254 - accuracy: 0.8817
Test accuracy: 0.8816999793052673
```

Рисунок 11 – Проверка точности предсказания

Качество предсказания немного ниже, но в целом достаточно высокое.

После обучения, будем предсказывать, что изображено на изображениях, для этого используем модель predict и изображения на которых модель обучалась. В квадратных скобках указываем индекс изображения. Выводится 10 значений по количеству нейронов и по количеству классов, данные значения в минусовой степени, а значит вероятность близка к 0, только одно число со значение в -1 степени,

это значит примерно 0,99, а единица это 100% вероятность. Таким образом модель предсказывает, что изображение соответствует последнему классу. Далее проверим и выведем правильный ответ из меток.

```
[ ]  # предсказания

[ ]  predictions = model.predict(x_train)

     1875/1875 [==============================] - 3s 2ms/step

     predictions[0]

     array([1.1451392e-13, 1.3642179e-12, 4.2296875e-14, 1.0918203e-13,
            1.6317009e-14, 8.2893166e-08, 2.3088554e-16, 5.1614526e-04,
            1.0776321e-12, 9.9948364e-01], dtype=float32)

[ ]  np.argmax(predictions[0])

     9

[ ]  y_train[0]
```

```
[52] predictions[0]
```

```
array([1.1451392e-13, 1.3642179e-12, 4.2296875e-14, 1.0918203e-13,
       1.6317009e-14, 8.2893166e-08, 2.3088554e-16, 5.1614526e-04,
       1.0776321e-12, 9.9948364e-01], dtype=float32)
```

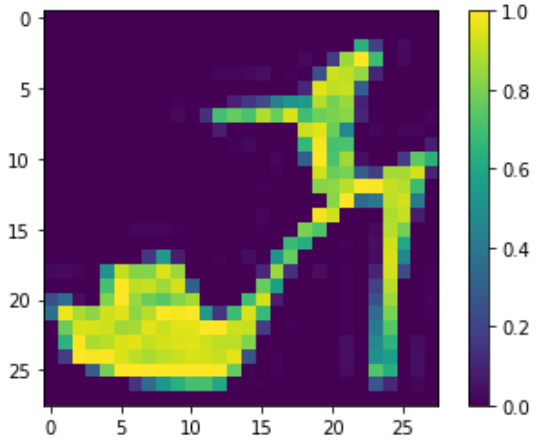
```
[53] np.argmax(predictions[0])
```

```
9
```

```
[54] y_train[0]
```

```
9
```

```
[55] plt.figure()
     plt.imshow(x_train[9])
     plt.colorbar()
     plt.grid(False)
```




```
[56] class_names[np.argmax(predictions[9])]
```

```
'Sandal'
```

Рисунок 12 – Результат работы модели

В итоге был произведен импорт изображений, далее разделили их на обучающую и тестовую выборку, оптимизировали изображения, после создали архитектуру нейронной сети из 3-х слоев, скомпилировали, обучили нейронную сеть и протестировали.

Задание.

1. Повторить все представленные выше шаги и создать нейронную сеть.

2. Протестировать нейронную сеть, написать выводы.
3. *Создать аналогичную нейронную сеть, но с другим набором данных.
4. Оформить отчет по форме: титульный лист, ход выполнения работы со скриншотами и описанием.

Содержание отчета и его форма

Отчёт по лабораторной работе должен содержать следующую информацию:

1) Название лабораторной работы и её номер.
2) ФИО и группу студента.
3) Формулировка индивидуального задания и номер варианта.
4) Ответ на задание в виде кода и результатов работы Matlab.
5) Ответы на контрольные вопросы.

Лабораторная работа № 2

Классификаторы

Цель и содержание работы: познакомить студента с персептроном.

Задачи:

1) Разобрать принцип действия и алгоритм работы персептрона;
2) Разобрать алгоритм минимизации среднеквадратической ошибки (LMS).

Формируемые компетенции: ПК-11

Теоретическое обоснование

Техника получения оптимального байесовского классификатора основывается на оценке функции распределения Гаусса для обучающих данных. Однако, это сложная задача особенно для многомерных данных. Альтернативное решение заключается в отделении классов в многомерном пространстве с помощью разделяющих гиперплоскостей.

Разделяющая гиперплоскость может быть записана в виде

$$w^T x + w_0 = 0\,. \tag{2.1}$$

Также (2.1) можно переписать в виде

$$w'^T x' \equiv [\mathrm{w}^T, \mathrm{w}_0] \begin{bmatrix} x \\ 1 \end{bmatrix} = 0\,, \tag{2.2}$$

где **w** – вектор настраиваемых параметров, w_0 – свободный член, **x** – входной вектор.

С помощью таких гиперплоскостей можно отделить линейно отделимые классы.

Алгоритм адаптации вектора весовых коэффициентов элементарного персептрона можно сформулировать следующим образом.

Если n-ый элемент x(n) обучающего множества корректно классифицирован с помощью весовых коэффициентов w(n), вычисленных на n-ом шаге алгоритма, то вектор весов не корректируется, т.е. действует следующее правило:

$$\begin{aligned} &w(\mathrm{n}+1) = \mathrm{w}(\mathrm{n}), \text{ если } \mathrm{w}^T x(\mathrm{n}) > 0 \text{ и } \mathrm{x}(\mathrm{n}) \in \mathrm{C}_1 \\ &w(\mathrm{n}+1) = w(\mathrm{n}), \text{ если } \mathrm{w}^T x(\mathrm{n}) \le 0 \text{ и } \mathrm{x}(\mathrm{n}) \in \mathrm{C}_2 \end{aligned} \tag{2.3}$$

В противном случае вектор весов персептрона подвергается коррекции в соответствии со следующим правилом:

$$\begin{aligned} &w(\mathrm{n}+1) = \mathrm{w}(\mathrm{n}) - \eta(\mathrm{n})\,\mathrm{x}(\mathrm{n}), \text{ если } \mathrm{w}^T(\mathrm{n}) x(\mathrm{n}) > 0 \text{ и } \mathrm{x}(\mathrm{n}) \in \mathrm{C}_1 \\ &w(\mathrm{n}+1) = w(\mathrm{n}) + \eta(\mathrm{n})\,\mathrm{x}(\mathrm{n}), \text{ если } \mathrm{w}^T(\mathrm{n}) x(\mathrm{n}) \le 0 \text{ и } \mathrm{x}(\mathrm{n}) \in \mathrm{C}_2, \end{aligned} \tag{2.4}$$

где C_1 и C_2 – линейно разделимые классы, а $\eta(n)$ – параметр скорости обучения.

Создадим функцию, которая будет обучать наш линейный нейрон (настраиваемую

гиперплоскость). Сигнатура функции будет следующей: **[w, iter, mis_clas] = perce(X, y, w_ini. rho)**, где X – это матрица размером (l+1)xN (l – размер входного вектора, N – количество паттернов для обучения), y – вектор меток для класса C_1 (+1) и C_2 (-1), w_ini – вектор начальных параметров, которые нужно настроить в процессе обучения так, чтобы гиперплоскость отделяла C_1 от C_2, rho – η=const, т.е. скорость обучения, w – вектор, вычисленный алгоритмом, iter – количество итераций за которые был вычислен w, mis_clas – количество неправильно классифицированных векторов.

Рассмотрим пример.

Сгенерируем 4 множества X_i, каждое – это набор точек в двумерном пространстве. Каждая точка из X_i имеет метку -1, точки случайно разбросаны в квадрате [3, 5]x[3, 5], [2, 4]x[2, 4], [0, 2]x[2, 4], [1, 3]x[1, 3]. Точки, распложенные в квадрате [0, 2]x[0,2] имеют метку класса +1. Требуется визуально отобразить классы, обучить персептрон для η=0.01 и η=0.05, и начальном векторе параметров $[1, 1, -0.5]^T$.

Листинг функции, реализующий персептрон, приведён ниже.

```
function [w,iter,mis_clas]=perce(X,y,w_ini,rho)

[l,N]=size(X);
max_iter=20000; % Максимальное количество итераций,
% которые будет работать персептрон, после этого будет
% считаться, что решение не найдено.

w=w_ini;       % Инициализируем вектор параметров
iter=0;        % Счётчик итераций

mis_clas=N;    % Инициализируем количество неправильно
% классифицированных паттернов

% цикл while по двум условиям
while(mis_clas>0)&&(iter<max_iter)
   iter=iter+1;
   mis_clas=0;

   gradi=zeros(l,1); % Вычисление корректирующего компонента для вектора
   for i=1:N
      if((X(:,i)'*w)*y(i)<0)
         mis_clas=mis_clas+1;
         gradi=gradi+rho*(-y(i)*X(:,i));
      end
   end

   if(iter==1)
      fprintf('\n First Iteration: # Misclassified points = %g \n',mis_clas);
   end

   w=w-rho*gradi; % Обновление вектора параметров
```

```
end
```

Листинг основного кода приведён ниже.

```
close('all');
clear

rand('seed',0);

% Генерируем класс Xi
N=[100 100]; % 100 векторов для каждого класса
l=2; % Размерность входа каждого вектора

% для генерирования паттернов из X2, X3, X4 нужно вместо
% нижней строчки подставить одну из закомментированных строк
x=[3 3]';
% x=[2 2]'; для X2
% x=[0 2]'; для X3
% x=[1 1]'; для X4

% получаем 200 точек для Xi и для точек квадрата [0, 2]x[0, 2]
X1=[2*rand(l,N(1)) 2*rand(l,N(2))+x*ones(1,N(2))];
X1=[X1; ones(1,sum(N))];
y1=[-ones(1,N(1)) ones(1,N(2))]; % получаем метки

% 1. Выводим множество Xi
figure(1), plot(X1(1,y1==1),X1(2,y1==1),'bo',...
X1(1,y1==-1),X1(2,y1==-1),'r.')
figure(1), axis equal

% 2. Запускаем алгоритм обучения персептрна для эта=0.01
rho=0.01; % Скорость обучения
w_ini=[1 1 -0.5]'; % начальные веса
[w,iter,mis_clas]=perce(X1,y1,w_ini,rho)
```

Имеем 4 ситуации X_i, i = 1..4, которые нужно отделить с помощью персептрона (рисунок 2.1 – 2.4).

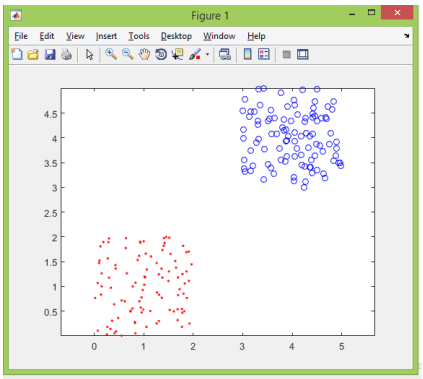


Рисунок 2.1 – Два класса для X_1

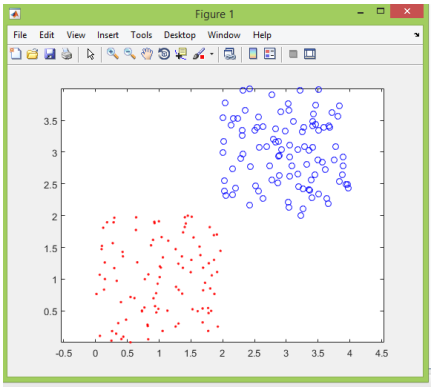


Рисунок 2.2 – Два класса для X_2

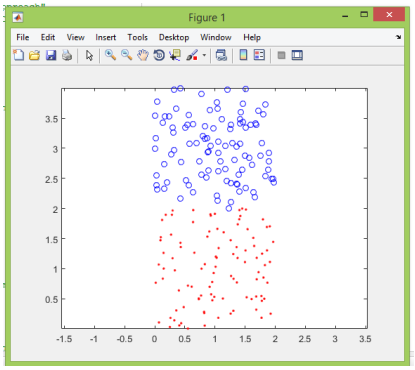


Рисунок 2.3 – Два класса для X_3

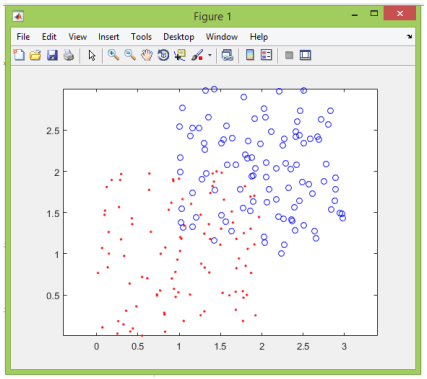


Рисунок 2.4 – Два класса для X_4

Видно, что в случае X_4 (рисунок 2.4) классы линейно не разделимы.

Результаты обучения приведены в таблице 2.1, 2.2.

Таблица 2.1 – Количество итераций, за которые персептрон находит решение в зависимости от η

	X_1	X_2	X_3	X_4
$\eta = 0.01$	134	134	5441	Оптимального решения не найдено

$\eta = 0.05$	5	5	252	Оптимального решения не найдено

Таблица 2.2 – Количество неправильно распознанных паттернов

	X_1	X_2	X_3	X_4
$\eta = 0.01$, $\eta = 0.05$	0	0	0	25

Видно, что при увеличении η увеличивается скорость обучения персептрона. В последнем случае оптимального решения персептрон не находит, поэтому проведённая прямая отделяет не все паттерны класса.

Функция **perce(X,y,w_ini,rho)** запрограммирована для пакетного режима, т.е. вектор **w** корректируется один раз после вычисления поправок для каждого примера. Ниже приведён листинг функции, где персептрон обучается в онлайн-режиме, где корректировки вычисляются для каждого примера, но далее, они не накапливаются, а сразу применяются к вектору весов.

```
function [w,iter,mis_clas]=perce_online(X,y,w_ini,rho)

[l,N]=size(X);
max_iter=10000000; % максимальное количество итераций
% требуется, если классы расположены близко друг к другу

w=w_ini;
iter=0;

mis_clas=N;

while(mis_clas>0)&&(iter<max_iter)
   mis_clas=0;

   for i=1:N
      if((X(:,i)'*w)*y(i)<0)
         mis_clas=mis_clas+1;
         w=w+rho*y(i)*X(:,i);  % Обновляем вектор весов
      end
      iter=iter+1;
   end

   if(iter==1)
      mis_clas
   end
end
```

Вызвать эту функцию можно с помощью следующей сигнатуры:

[w,iter,mis_clas]=perce_online(X1,y1,w_ini,rho).

Если поставить предыдущий эксперимент, то получим следующие результаты, таблица 2.3, 2.4.

Таблица 2.3 – Количество итераций, за которые персептрон находит решение в зависимости от η

	X_1	X_2	X_3	X_4
η = 0.01	600	600	6589400	Оптимального решения не найдено
η = 0.05	400	400	7729200	Оптимального решения не найдено

Таблица 2.4 – Количество неправильно распознанных паттернов

	X_1	X_2	X_3	X_4
η = 0.01, η = 0.05	0	0	0	6

Требуется значительно больше итераций, онлайн-обучение хуже сходится, зато в практическом плане более экономное, т.к. не требует специальных структур для хранения и накопления поправок.

Аппаратура и материалы. 64-разрядный (x64) персональный компьютер, процессор с тактовой частотой 1 ГГц и выше, оперативная память 1 Гб и выше, свободное дисковое пространство не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система Windows 7 и выше, Matlab (R2013) и выше.

Указание по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

Методика и порядок выполнения работы

Создать два частично пересекающихся класса точек (100 точек для каждого класса) как показано в таблице 2.5. Разделить их насколько это возможно с помощью персептрона. Тип

обучения персептрона онлайн или пакетный указан в таблице. Нарисовать найденное решение.

Таблица 2.5 – Индивидуальные варианты

1	$\begin{pmatrix} 0 & A \\ B & 0 \end{pmatrix}$ Онлайн обучение.
2	$\begin{pmatrix} A & 0 \\ B & 0 \end{pmatrix}$ Пакетное обучение.
3	$\begin{pmatrix} 0 & 0 \\ A & B \end{pmatrix}$ Пакетное обучение.
4	$\begin{pmatrix} 0 & A \\ B & 0 \end{pmatrix}$ Онлайн обучение.
5	$\begin{pmatrix} 0 & 0 \\ A & B \end{pmatrix}$ Пакетное обучение.
6	$\begin{pmatrix} A & 0 \\ B & 0 \end{pmatrix}$ Пакетное обучение.
7	$\begin{pmatrix} 0 & A \\ B & 0 \end{pmatrix}$ Онлайн обучение.
8	$\begin{pmatrix} A & 0 \\ B & 0 \end{pmatrix}$ Онлайн обучение.
9	$\begin{pmatrix} 0 & A \\ B & 0 \end{pmatrix}$ Пакетное обучение.
10	$\begin{pmatrix} 0 & 0 \\ A & B \end{pmatrix}$ Онлайн обучение.

Содержание отчета и его форма

Отчёт по лабораторной работе должен содержать следующую информацию:

1) Название лабораторной работы и её номер.
2) ФИО и группу студента.
3) Формулировка индивидуального задания и номер варианта.
4) Ответ на задание в виде кода и результатов работы Matlab.

Вопросы для защиты работы

1. Что такое персептрон?
2. Чем персептрон отличается от сети прямого распространения?
3. Какие задачи можно решать с помощью персептрона?

Лабораторная работа № 3

Создание и обучение нейронной сети на языке высокого уровня среды Matlab

Цель и содержание работы: создать и обучить несколько нейронных сетей, решающих задачи линейного и нелинейного разделения классов, используя встроенный язык программирования Matlab.

Задачи:

— освоить базовые команды для создания и обучения простых сетей в Matlab;

— научиться создавать и обучать сети для линейного и нелинейного разделения классов;

— научиться строить графики, отражающие результаты работы сетей.

Формируемые компетенции: ПК-11

Теоретическое обоснование

Самый гибкий способ по работе с ИНС в Matlab – это использовать встроенный язык программирования для создания и обучения сетей. В данной лабораторной работе на примере линейного разделения классов и задачи XOR будет рассмотрена техника создания и обучения разных нейронных сетей.

Код Matlab будем выделять жирным шрифтом. Комментарий к коду записывается после символа «%» и идёт до конца строчки. При желании, можно получить более подробную справку о любой функции, установив курсор на нужной и нажав клавишу «F1».

Прежде, чем программировать персептрон, рассмотрим пример построения поверхности отклика для обычного нелинейного нейрона с функцией активации – гиперболический тангенс.

```
% Готовим Matlab к работе
close all, clear all, clc, format compact;
% Задаём веса нейрона
w = [4 -2];
% Задаём смещение
b = -3;
% Задаём функцию, которую хотим построить, гипербол. тангенс
func = 'tansig';
% Задаём входной вектор
p = [2 3];
% Вычисляем взвешенную сумму входа и весов
activation_potential = p*w'+b;
```

```
% Вычисляем выход нелинейной части нейрона
neuron_output = feval(func, activation_potential);
% Задаём входной вектор для диапазона от -10 до +10 с шагом 0.25
[p1, p2] = meshgrid(-10:.25:10);
% Вычисляем вектор выхода нейрона
z = feval(func, [p1(:) p2(:)]*w'+b);
% Пересчёт вещественных чисел в воксели для 3D-графика
z = reshape(z, length(p1), length(p2));
% Строим график
plot3(p1, p2, z);
% Включаем сетку
grid on;
% Подписываем оси графика
xlabel('Input 1');
ylabel('Input 2');
zlabel('Neuron output');
```

Разберём код. Команда **close all** закрывает все вспомогательные окна, которые могли быть открыты (окна различных мастеров, вроде nntool, не закрываются). **Clear all** удаляет все переменные из области workspace, **clc** – очищает область окна Command window, где будет набираться код. **Format compact** устанавливает формат отображения для чисел, так для вещественных числе будет отображаться 4 точки после запятой.

Веса задаются обычным вектором **w = [4 -2]**, как видно, значения отделяются друг от друга пробелом, указывать тип не обязательно. Функция активации задаётся строкой **'tansig'**. Далее идёт вычисление взвешенной суммы: скалярное произведение входа на веса и прибавка смещения. Вектора **p** и **w** нельзя просто так взять и перемножить, т.к. поправилам линейной алгебры один из них должен быть транспонирован, что и делается с вектором **w** с помощью **w'**. Можно посмотреть значение переменной, оно должно быть -1. Далее вычисляем функцию активации при входе равном -1. Делаем это через **feval()**. Как понятно из названия, функция вычисляет значение некоторой функции (первый параметр) от вектора входов (второй параметр). Причём, как и многие другие фунции Matlab, она перегружена, т.е. её вызов может происходить при разных способах записи второго параметра. В первом случае вектор редуцируется к скалярной величине, выход равен - 0.7616.

Далее присваиваем переменным **p1** и **p2** значения от -10 до +10 с шагом 0.25. Делаем это через **meshgrid()**, которая создаёт прямоугольную сетку в специальном формате пригодном для построения графиков. Но для построения самого графика нам нужно знать значения не только **p1** и **p2**, но и выхода нейрона. Теперь вычисляем эти значения для вектора **z** с помощью **feval()**, видно, что теперь второй параметр напрямую записывается в виде скалярного произведения двух векторов плюс смещение. Двоеточие означает перевод значения в формат double (используется для научных и инженерных вычислений. Никогда

не используйте float для этих задач в таких языках как C/C++, чтобы избежатьпереполнения переменной).

Функция **reshape()** пересчитывает числовые значения в воксели для отображения на графике. **Plot3()** строит трёхмерный график, рисунок 3.1. **Grid on** включает сетку на этом графике, рисунок 3.2, 3.3. **Xlabel()**, **ylabel()**, **zlabel()** – подписывают оси у графика.

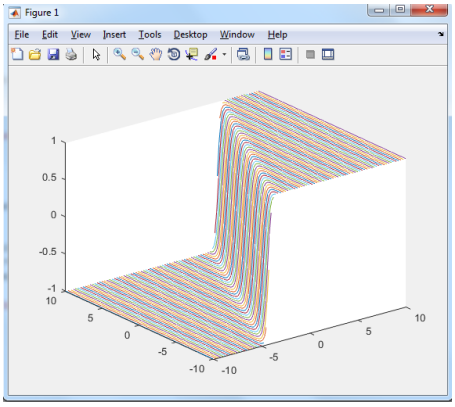


Рисунок 3.1 – Получившийся 3D-гарфик без сетки и подписей осей

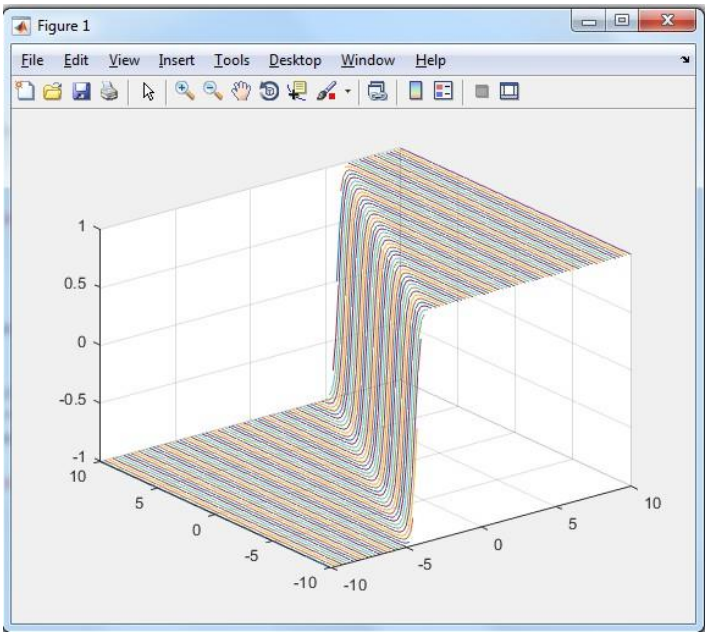


Рисунок 3.2 – Добавили сетку на фон

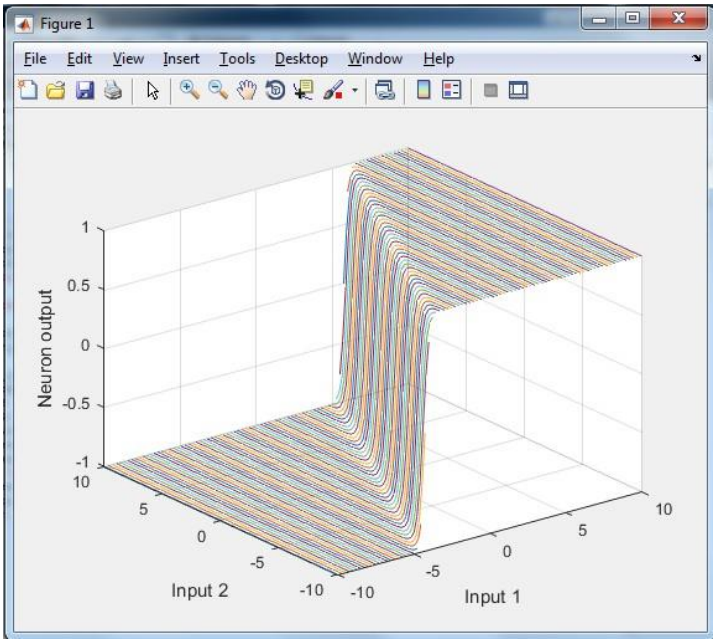


Рисунок 3.3 – Добавили подписи для осей x, y, z

Теперь создадим и обучим двухслойную сеть на одном примере.

```
close all, clear all, clc, format compact
% Входной транспонированный вектор
inputs = [1:6]'
% Выходной вектор, который должна промоделировать сеть
outputs = [1 2]';
% Задаём общую структуру сети
net = network(1, 2, [1; 0], [1; 0], [0 0; 1 0], [0 1]);
% Показываем её
view(net);
% Задаём количество промежуточных слоёв
net.layers{1}.size = 5;
% Задаём функции активации на них
net.layers{1}.transferFcn = 'logsig';
% Ещё раз отображаем сеть
view(net);
% Устанавливаем размерность входа и выхода
net = configure(net, inputs, outputs);
% Смотрим окончательный вариант сети
view(net);
% Получаем изначальный выход сети без обучения
initial_output = net(inputs);
% Устанавливаем метод обучения
net.trainFcn = 'trainlm';
% Устанавливаем функцию ошибки
net.performFcn = 'mse';
% Обучаем сеть на одном примере
net = train(net, inputs, outputs);
% Опять подаём вход, но уже на обученную сеть,
% сеть должна промоделировать выход [1 2]
final_output = net(inputs);
```

Входной вектор будет равен массиву из шести элементов от 1 до 6, т.к. диапазон записывается через «:». Функция **network()** задаёт общую структуру сети. Рассмотрим более подробно, что означает каждый параметр.

Первый параметр означает количество входов (не в смысле размерность входного вектора, а в смысле, – сколько векторов будет подано на вход сети. Допустим, в сверточных сетях обычное дело, что на вход подаётся не один, а два или три массива, каждый из которых как-то связан с дальнейшим слоем). У нас этот параметр равен 1. Второй параметр означает количество слоёв в сети. В данном случае имеем двухслойную сеть. Третий параметр – булев вектор, который означает какие нейроны будут иметь смещение, а какие –нет. Т.к. вектор [1; 0], то очевидно, что все нейроны первого слоя будут иметь смещение, а нейроны второго слоя – нет. Четвёртый параметр – это также булев вектор, который означает, – с какими слоями связан вход. В данном случае используется классическая схема, где вход связна только с последующим слоем, а следовательно, вектор [1; 0]. Пятый параметр – это матрица связей между слоями, которая означает, какой слой с каким связан. Она записывается в виде вектора. Имеем [0 0; 1 0], если записать в виде матрицы, то получится $\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$. Тогда понятно, что первый слой связан со вторым слоем, т.к. единица располагается в позиции (1, 2), где 1 – это номер столбца, 2 – номер строки. Шестой параметр [0 1] – булев вектор, который показывает, с какими слоями связан выход. При таких значениях выход связан с предпоследним слоем.

При такой конфигурации получим структуру сети как на рисунке 3.4.

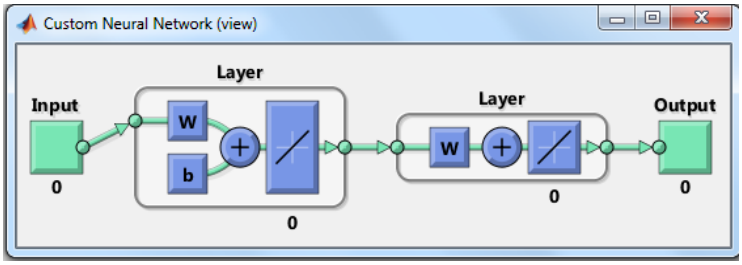


Рисунок 3.4 – Изначальная структура сети

Изменим третий параметр с [1; 0] на [1; 1], получим структуру как на рисунке 3.5.

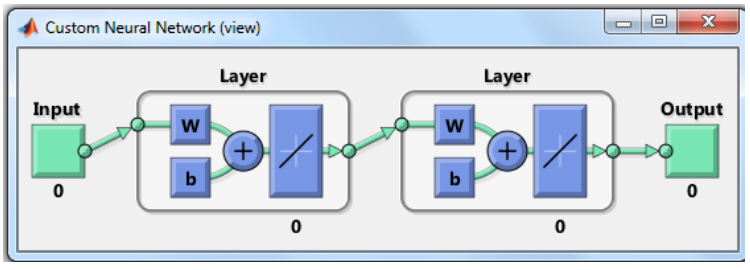


Рисунок 3.5 – Видно, что у второго слоя тоже появилось смещение

Изменим теперь и четвёртый параметр на [1; 1], получим структуру сети как на рисунке 3.6.

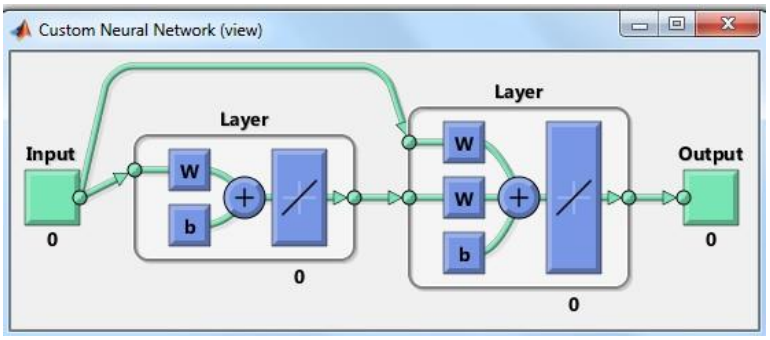


Рисунок 3.6 – Добавилась связь входа со вторым слоем

Изменим остальные два вектора на единичные, получим структуру как на рисунке 3.7.

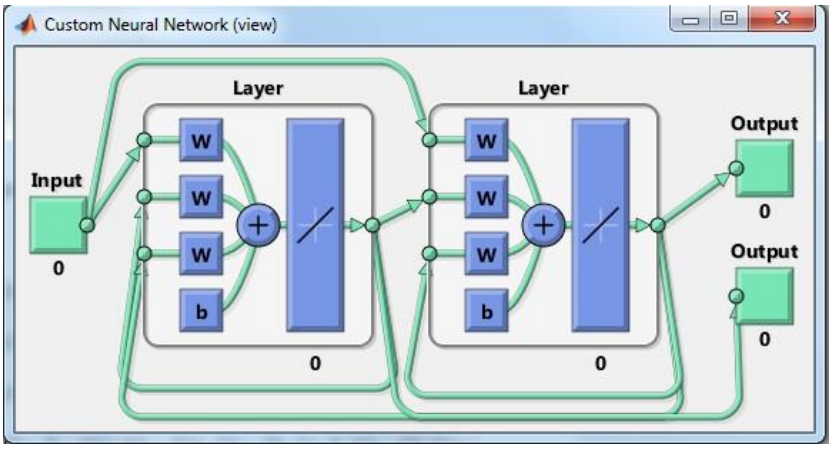


Рисунок 3.7 – Появился второй выход, связанный с первым слоем, а также каждый слой связан с самим собой и второй слой связан с первым (подаёт свои сигналы на вход первого слоя)

Net.layers{1}.size = 5 – присваиваем количество нейронов первому слою сети (индекс записывается в фигурных скобках). Видно, что обращение к данным параметрам осуществляется классическим образом для объектов: через точку. Т.е. сеть в Matlab – это объект (по терминологии объектно-ориентированного программирования (ООП)).

Небольшое примечание. Концепция ООП напрашивается для описания ИНС, т.к. вполне логично, что объект «сеть» включает в себя объекты – «слой», а те – «нейрон». Однако, нужно помнить, что при таком подходе оперативная память выделяется и группируется под конкретные объекты и в случае попыток быстрой реализации ИНС,

возможно, с использованием ассемблера, не получится быстро реализовывать матричные операции. Короче, если требуется быстрая реализация сети, то от ООП-описания придётся отказаться.

Net.layers{1}.transferFcn = 'logsig' – присваивает слоям первого слоя функцию активации сигмойд (без отрицательных значений), рисунок 3.8.

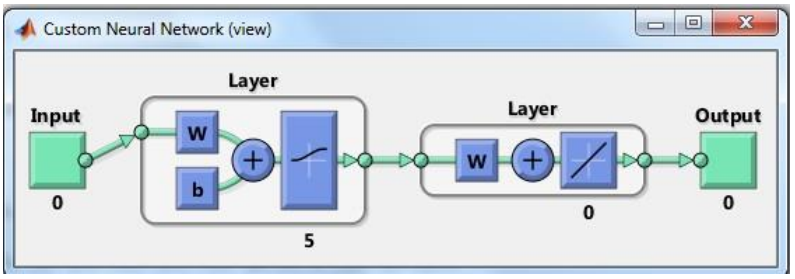


Рисунок 3.8 – Меняем функцию активации у нейронов первого слоя

Для установки входного и выходного слоя можно сконфигурировать сеть по отношению к вектору входа и выхода: **net = configure(net, inputs, outputs)**, рисунок 3.9.

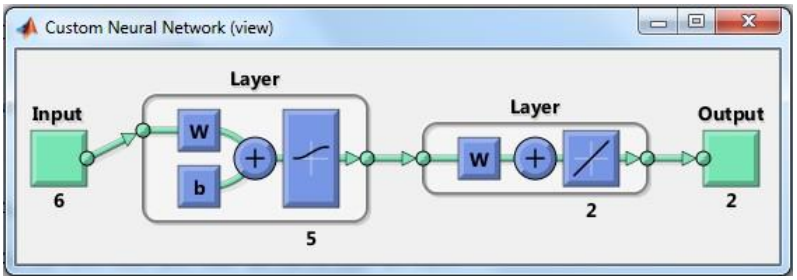


Рисунок 3.9 – Окончательно получаем общую структуру сети, где размер входа – 6, а выхода – 2

Вектор **initial_output** получит значения $[0\ 0]^T$, т.к. веса не инициализированы, т.е. сеть ничего делать не умеет. Для начала обучения нужно как минимум задать метод обучения и функцию ошибки, что и делается с помощью **net.trainFcn = 'trainlm'** и **net.performFcn = 'mse'**. **Trainlm** – это обучение по методу Левенберга-Марквардта, а **MSE** (mean square error) – это среднеквадратическая ошибка. **Train()** – запускает обучение сети и возвращает объект – обученную сеть. Процесс обучения показан на рисунке 3.10.

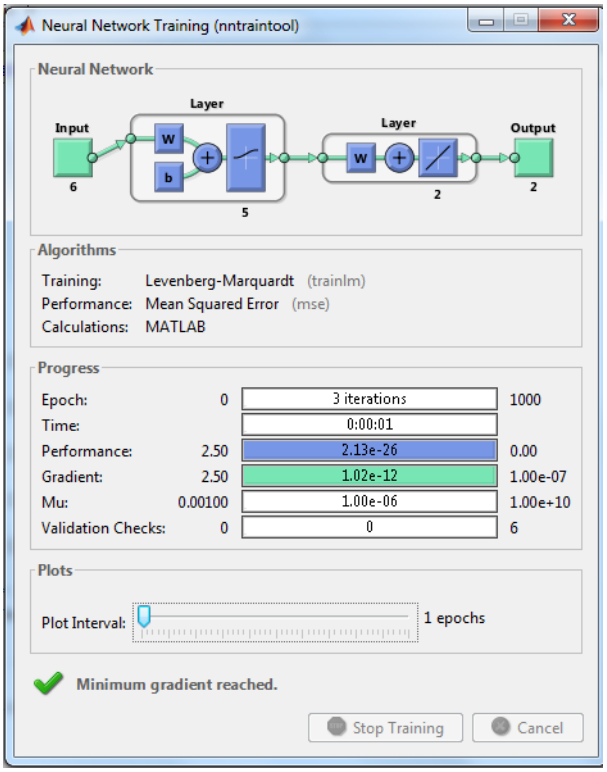


Рисунок 3.10 – Обучение сети

Позже формируем вектор **final_output**, который содержит ответ сети на вход, т.е. 1.0000 и 2.0000 (помним про формат **compact**).

Теперь решим задачу о разделении классов с помощью персептрона, подобную той, которая была решена в лабораторной работе 3.

```
close all, clear all, clc, format compact
% Задаём количество паттернов каждого класса
N = 20;
% Задаём смещение
offset = 5;
% Создаём входные значения каждого класса
x = [randn(2, N) randn(2, N)+offset];
% Создаём выходные значения для этих классов
y = [zeros(1, N) ones(1, N)];
% Открываем окно для рисования
figure(1);
% Наносим точки (паттерны)
plotpv(x, y);
% Создаём объект-сеть типа Персептрон
net = perceptron;
% Обучаем персептрон
net = train(net, x, y);
% Рисуем общую структуру сети
view(net);
% Возвращаемся в окно для рисования и рисуем прямую линию,
% которая разделит два класса
figure(1);
```

plotpc(net.IW{1}, net.b{1});

rand(2, N) – создаёт массив размером 2xN, числа которого являются случайные величины, распределённые по нормальному закону с математическим ожиданием 0 и среднеквадратическим отклонением 1. Таким образом, будут созданы две матрицы 2xN, содержащие координаты точек-паттернов в двумерном пространстве, причём координаты второй матрицы будут смещены на 5 позиций, т.е. заранее гарантируется, что классы будут линейно разделимые. Обратим внимание, что эти две матрицы на самом деле записаны в одну матрицу размером 2x2*N. **Zeros(1, N)** и **ones(1, N)** создаёт массив нулей и единиц размером N, это метки для входных паттернов.

Figure(1) – вызывает окно, содержащее канву для рисования, рисунок 3.11.

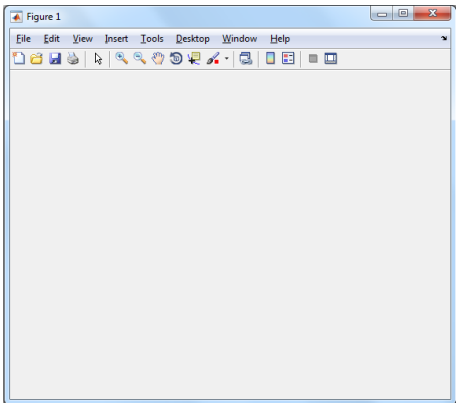


Рисунок 3.11 – Окно для рисования

Plotpv(x, y) – наносит точки на канву с координатами x и метками y, рисунок 3.12. Причём нули отображаются кружками, а единицы плюсами.

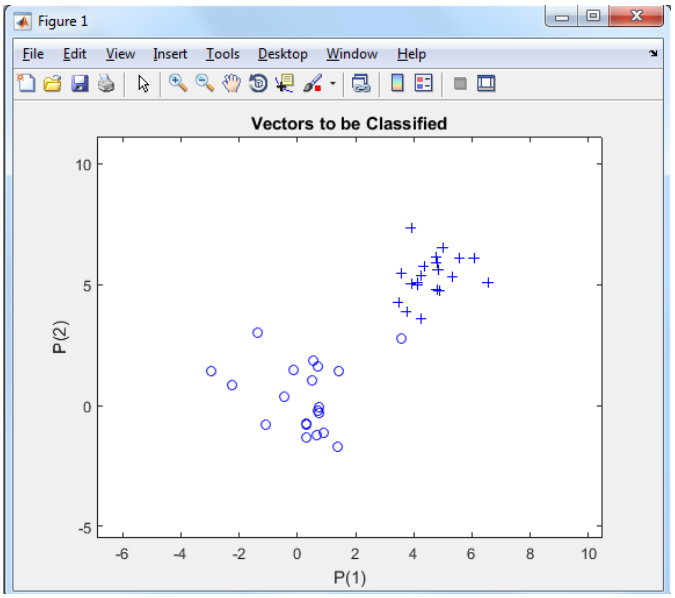


Рисунок 3.12 – Два созданных нами класса, видно, что они линейно разделимы

Net = perceptron – создаём сеть типа Персептрон, если после записи команды не ставить точку с запятой, то среда развернёт все установленные по умолчанию параметры, рисунок 3.13.

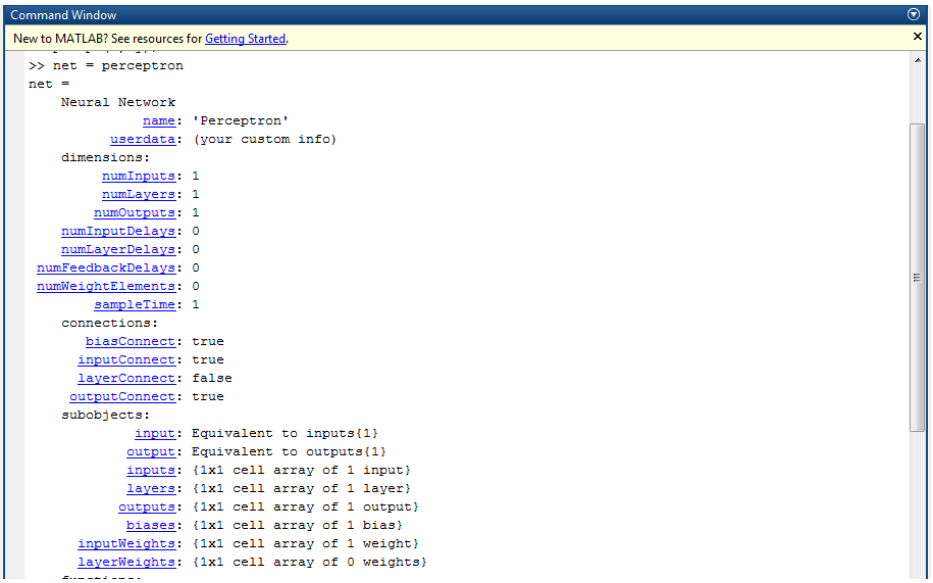


Рисунок 3.13 – Методы и свойства объекта perceptron

Далее происходит обучение сети. Общая структура сети представлена на рисунке 3.14.

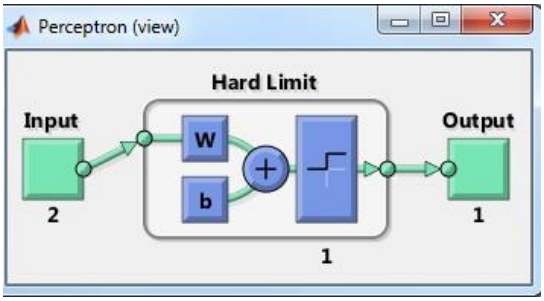


Рисунок 3.14 – Общая структура сети

Результаты обучения представлены на рисунке 3.15. Видно, что потребовалось 29 итераций.

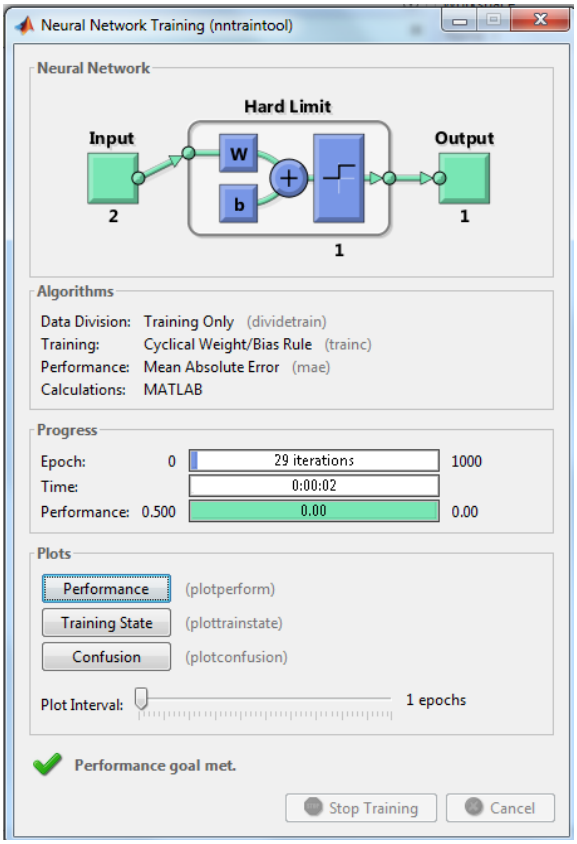


Рисунок 3.15 – Результаты обучения персептрона

На рисунке 3.16 представлен график настройки персептрона. График носит характерную зубчатую форму. После каждой эпохи высчитывается MSE между предполагаемыми ответами и реальными. Напомним, что при адаптации не используется спуск по поверхности ошибки, поэтому ждать постепенного снижения ошибки обучения не следует. Обучение идёт до тех пор, пока не будет найдена первая прямая, отделяющая один класс от другого.

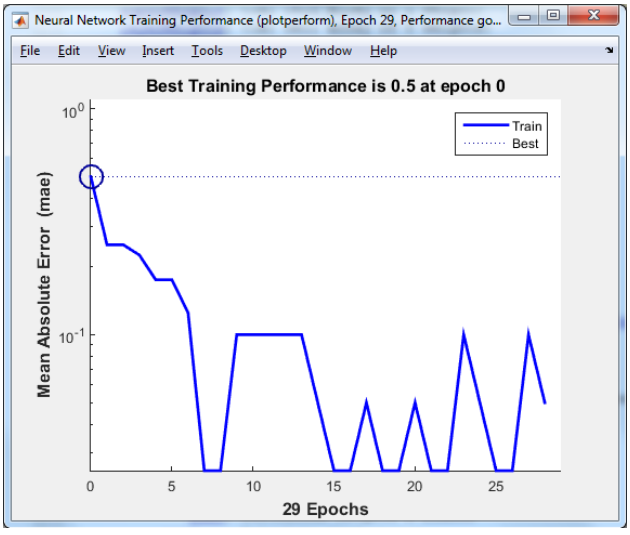


Рисунок 3.16 – График настройки весовых коэффициентов персептрона

Plotpc(net.IW{1}, net.b{1}) – строит прямую линию с соответствующими коэффициентами. **IW** – обозначает слой коэффициентов между входом и первым слоем, **b{1}** – смещение первого слоя. Итоговая прямая показана на рисунку 3.17.

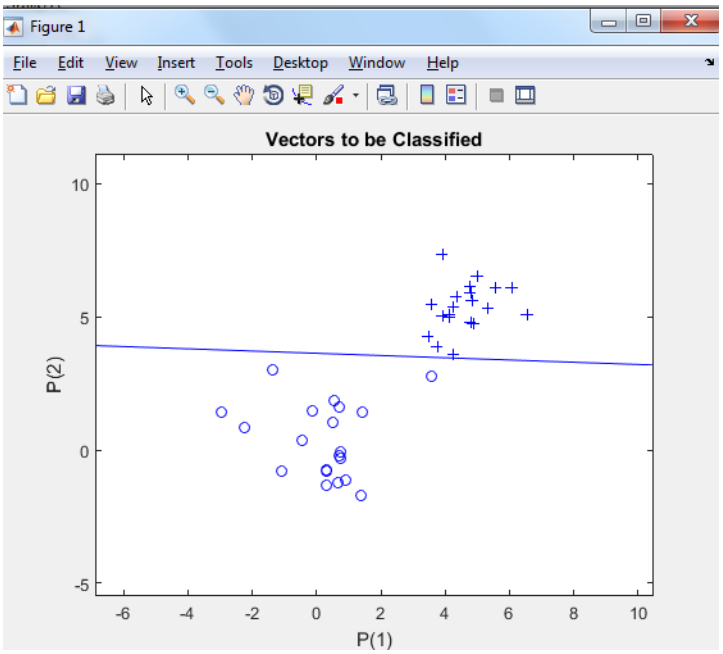


Рисунок 3.17 – Полученное решение по разделению двух классов

Теперь рассмотрим решение задачи разделения на 4 класса с помощью персептрона и обычной многослойной сети прямого распространения. Расположение классов оставим таким же, как и в задаче XOR (по углам «квадрата»), но теперь у нас не два класса, а 4, а также каждый класс представлен 30 паттернами.

Решение этой задачи с помощью персептрона.

```
close all, clear all, clc, format compact
% Каждый класс имеет 30 паттернов
K = 30;
% Смещение для классов 0.6
q = .6;
% Задаём 4 класса в виде векторов A, B, C, D (2 строки и 30 столбцов)
A = [rand(1, K)-q; rand(1, K)+q];
B = [rand(1, K)+q; rand(1, K)+q];
C = [rand(1, K)+q; rand(1, K)-q];
D = [rand(1, K)-q; rand(1, K)-q];
% Рисуем на канве точки A определённого типа (третий параметр)
plot(A(1, :), A(2, :), 'bs');
% Фиксируем канву
hold on;
grid on;
% На той же канве рисуем остальные классы
plot(B(1, :), B(2, :), 'r+');
plot(C(1, :), C(2, :), 'go');
plot(D(1, :), D(2, :), 'm*');
```

```
text(.5-q, .5+2*q, 'Class A');
text(.5+q, .5+2*q, 'Class B');
text(.5+q, .5-2*q, 'Class C');
text(.5-q, .5-2*q, 'Class D');
% Задаём кодировку классов (метки)
a = [0 1]';
b = [1 1]';
c = [1 0]';
d = [0 0]';
% Получаем общий вектор входных значений
P = [A B C D];
% Получаем общий вектор меток
T = [repmat(a, 1, length(A)) repmat(b, 1, length(B)) repmat(c, 1, length(B)) repmat(d, 1, length(D))];
% Сеть типа «персептрон»
net = perceptron;
% Пусть среднеквадратическая ошибка не равна 0
E = 1;
% Установка параметра о том, что будет минимум один прогон
net.adaptParam.passes = 1;
% Построили изначально ответ персептрона, до обучения
linehandle = plotpc(net.IW{1}, net.b{1});
% Счётчик по циклам
n = 0;
% Цикл будет идти пока среднеквадратическая ошибка не
% станет равной нулю или пока не достигнем значения итераций в 1000
while (sse(E) & n<1000)
        n = n + 1;
        % Обучить персептрон
        [net, Y, E] = adapt(net, P, T);
        % Нарисовать линии
        linehandle = plotpc(net.IW{1}, net.b{1}, linehandle);
        drawnow;
end
view(net);
% Проверка работы персептрона на входном векторе
p = [0.7; 1.2];
% Выдаст класс (1; 1)
y = net(p);
```

rand(1, K) – генерирует числа из равномерного распределения в виде вектора (первый параметр) размером K, в итоге **A = [rand(1, K)-q; rand(1, K)+q]** – генерируется матрица 2xK, что соответствует координатам точек.

Plot(A(1, :), A(2, :), 'bs') – рисует на канве точки из матрицы A, переводя их в вещественный формат. Параметр **'bs'** – это цвет и форма точек: b – синий и s – квадрат. **'r+'** – красный (r) крест (+), **'go'** – зелёный (g) кружок (o), **'m*'** – пурпурная (m) звездочка (*).

Text(.5-q, .5+2*q, 'Class A') – наносит текст на канву с координатами, записанными как два первых параметра.

Общий вид отображения классов представлен на рисунке 3.18.

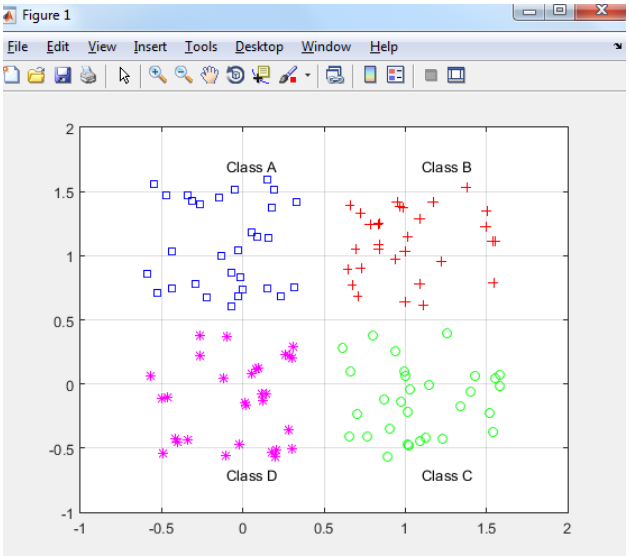


Рисунок 3.18 – Классы, которые предстоит разделить

Для получения вектора T необходимо раскопировать транспонированные вектора **a**, **b**, **c**, **d**. Для этой цели используется **repmat()**: **repmat(что копируем, копий по строкам, копий по столбцам)**. В результате вектор **T** примет вид как на рисунке 3.19: матрица из двух строк и 30 * 4 = 120 колонок.

```
>>  T = [repmat(a, 1, length(A)) repmat(b, 1, length(B)) repmat(c, 1, length(C)) repmat(d, 1, length(D))]
T =
  Columns 1 through 17
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
  Columns 18 through 34
     0     0     0     0     0     0     0     0     0     0     0     0     0     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
  Columns 35 through 51
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
  Columns 52 through 68
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     0     0     0     0     0     0     0     0
  Columns 69 through 85
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 86 through 102
     1     1     1     1     1     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 103 through 119
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
  Column 120
     0
     0
```

Рисунок 3.19 – Общий вид вектора меток T

Цикл **while()** – это цикл с предусловие, поэтому он может не выполниться ни разу, а следовательно, перед ним необходимо построить возможное решение (т.к. при создании сети **net = perceptron**) веса получают случайные значения.

[net, Y, E] = adapt(net, P, T) – процесс адаптации весов. **Net** – получит обученную сеть, **Y** – вектор ответов сети для входа **P**, **E** – ошибки сети для меток **T** и входов **P**. **Sse(E)** –

сумма квадратов этих ошибок.

На рисунке 3.20 показана структура сети, видно, что единственный слой сети содержит два нейрона, что соответствует двум прямым линиям для разделения 4-х классов.

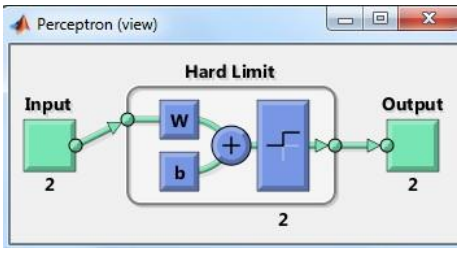


Рисунок 3.20 – Общая структура персептрона

Возможный ответ сети приведён на рисунке 3.21.

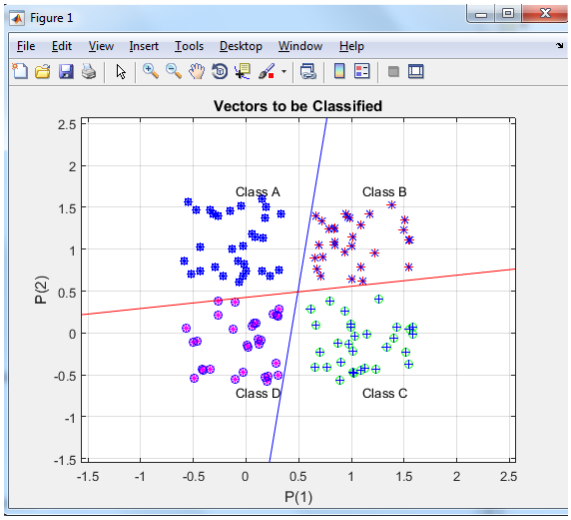


Рисунок 3.21 – Решения задачи разбития на 4 класса

Рассмотрим решение XOR-проблемы с помощью многослойной сети прямого распространения, но расширим представление каждого класса до 100 паттернов.

```
close all, clear all, clc, format compact
K = 100;
q = .6;
% Данные для обучения
%Обратите внимание на то, что randn() отделяются с помощью
%",". Если опустить этот символ, то A будет не 2x100, а 1x200.
A = [rand(1, K)-q; rand(1, K)+q];
B = [rand(1, K)+q; rand(1, K)+q];
C = [rand(1, K)+q; rand(1, K)-q];
D = [rand(1, K)-q; rand(1, K)-q];
figure(1);
% Наносим точки на канву
plot(A(1, :), A(2, :), 'k+');
hold on;
grid on;
plot(B(1, :), B(2, :), 'bd');
```

```
plot(C(1, :), C(2, :), 'k+');
plot(D(1, :), D(2, :), 'bd');
% Наносим названия классов
text(.5-q, .5+2*q, 'Class A');
text(.5+q, .5+2*q, 'Class B');
text(.5+q, .5-2*q, 'Class A');
text(.5-q, .5-2*q, 'Class B');
% Определяем метки для классов
a = -1, c = -1, b = 1, d = 1;
% Формируем входной вектор
P = [A B C D];
% Формируем вектор ответов
T = [repmat(a, 1, length(A)) repmat(b, 1, length(B)) repmat(c, 1, length(C)) repmat(d, 1, length(D))];
% Определяем сеть прямого распространения
net = feedforwardnet([5 3]);
% Вся выборка под обучающие примеры
net.divideParam.trainRatio = 1;
% Нет валидационной выборки
net.divideParam.valRatio = 0;
% Нет тестовой выборки
net.divideParam.testRatio = 0;
% Обучаем сеть
[net, tr, Y, E] = train(net, P, T);
view(net);
figure(2);
% График ожидаемых ответов сети
% график получается зубчатым, т.к. метки для классов
%определены как -1 и +1
plot(T', 'linewidth', 2);
hold on;
% График реальных ответов сети
plot(Y', 'r--');
grid on;
% Рисуем легенду к графикам
legend('Targets', 'Network response', 'location', 'best');
% Определяем видимый диапазон по оси y
ylim([-1.25 1.25]);
% Создаём  набор 601 числа
span = -1:.005:2;
% P1 и P2 станут матрицами размером 601x601
[P1, P2] = meshgrid(span, span);
% Транспонируем вектор резмером 601*601 = 361201
pp = [P1(:) P2(:)]';
% Получаем выходы сети
aa = net(pp);
% Возвращаемся к первой канве
figure(1);
% Рисуем трёхмерную сетку
mesh(P1, P2, reshape(aa, length(span), length(span))-5);
% Подбираем лучшее сочетание цветов
colormap cool;
```

На рисунке 3.22 показаны паттерны входных классов.

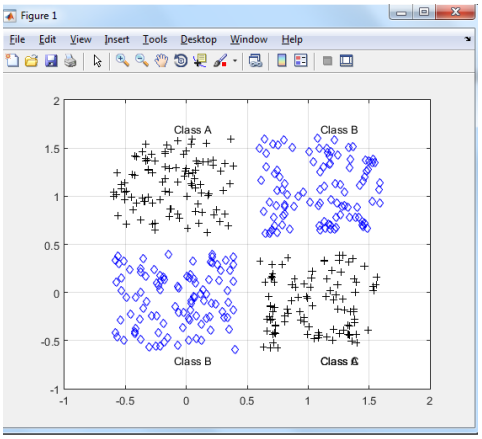


Рисунок 3.22 – Входные данные для двух классов, которые нужно разделить

Net = feedforwardnet([5 3]) – создаём сеть прямого распространения с двумя слоями, в первом будет 5 нейронов, во втором – 3. Структура сети показана на рисунке 3.23.

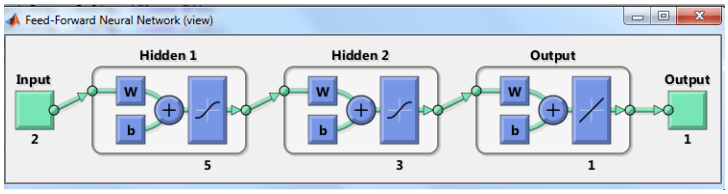


Рисунок 3.23 – Общая структура сети

Обратим внимание, что выходной нейрон имеет линейную функцию активации.

Все паттерны войдут в обучающую выборку, поэтому **net.divideParam.trainRatio = 1**, валидационную и тестовую выборку установим в 0.

Обучение сети осуществляется через **[net, tr, Y, E] = train(net, P, T)** – где **tr** – переменная, содержащая информацию об обучении (в среде Matlab её можно открыть и посмотреть на входящие в неё поля и их значения). Процесс обучения показан на рисунке 3.24, кривая обучения на рисунке 3.25.

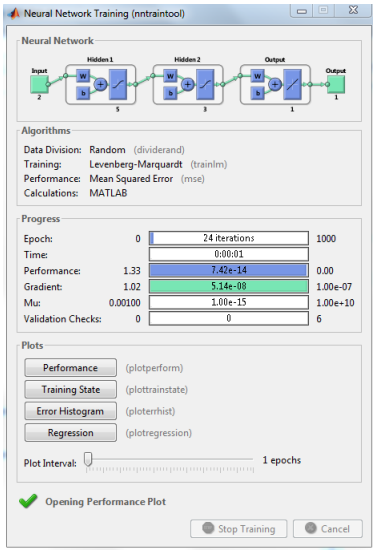


Рисунок 3.24 – Процесс обучения сети, потребовалось 24 итерации

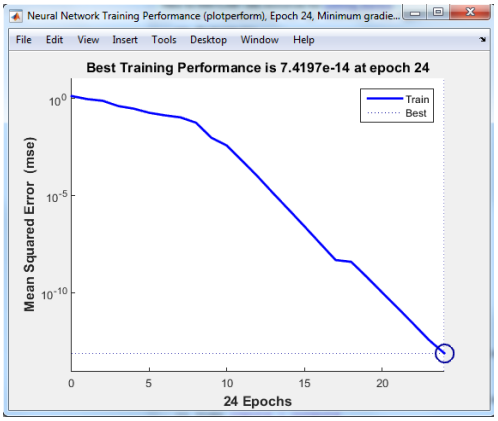


Рисунок 3.25 – Кривая обучения (MSE), лучший показатель меньше 10^{-10}

Далее отображаем на новой канве ожидаемый ответ сети, рисунок 3.26.

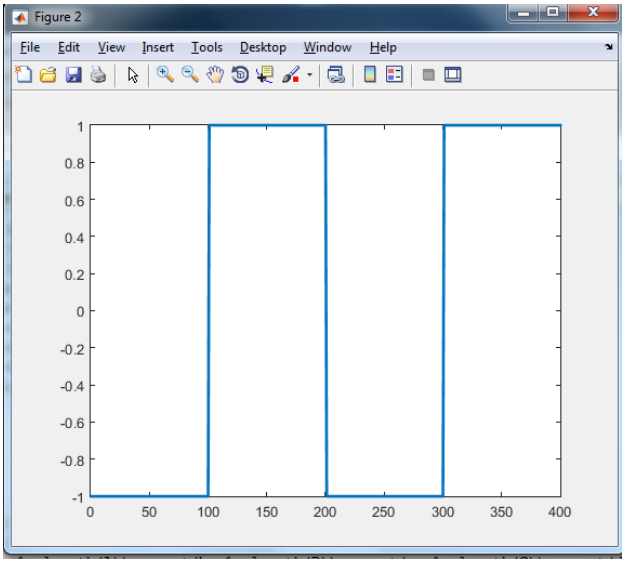


Рисунок 3.26 – Ожидаемый ответ сети (классы кодировались ±1)

На той же канве отображаем реальный ответ сети, рисунок 3.27. Видно, что графики совпадают.

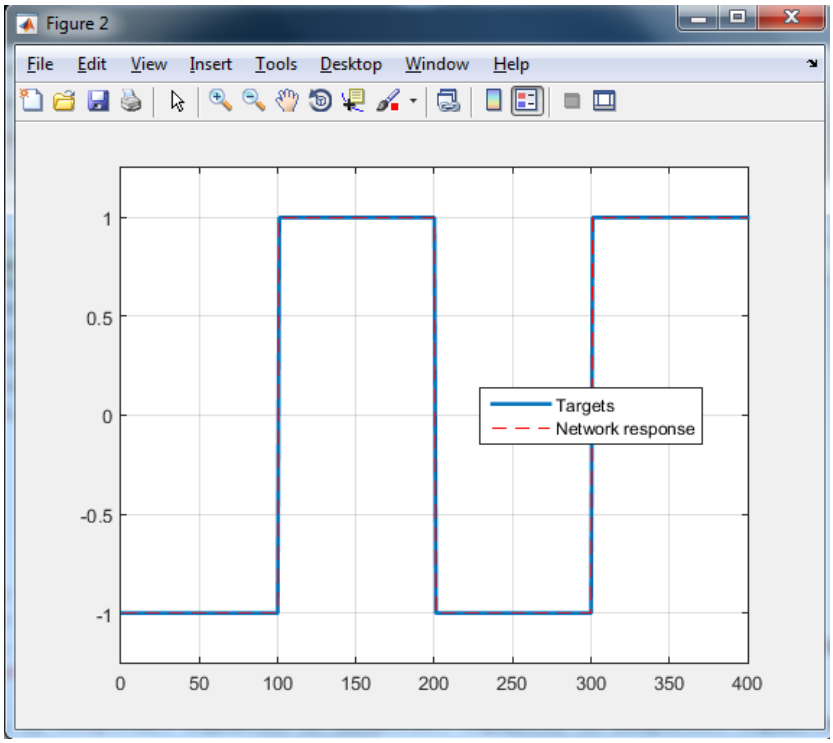


Рисунок 3.27 – Реальный ответ сети, графики совпадают

Ylim([-1.25 1.25]) – ограничивает ось OY данными значениями.

Чтобы отобразить получившееся решение уже нельзя просто построить прямые линии (как в случае с персептроном). Для этого нужно поверх первой канвы отобразить значения выходов сети на очень мелкой сетке (входные данные) и закрасить получившиеся ответы. Граница решения для разбиения двух классов окажется видна, рисунок 3.28.

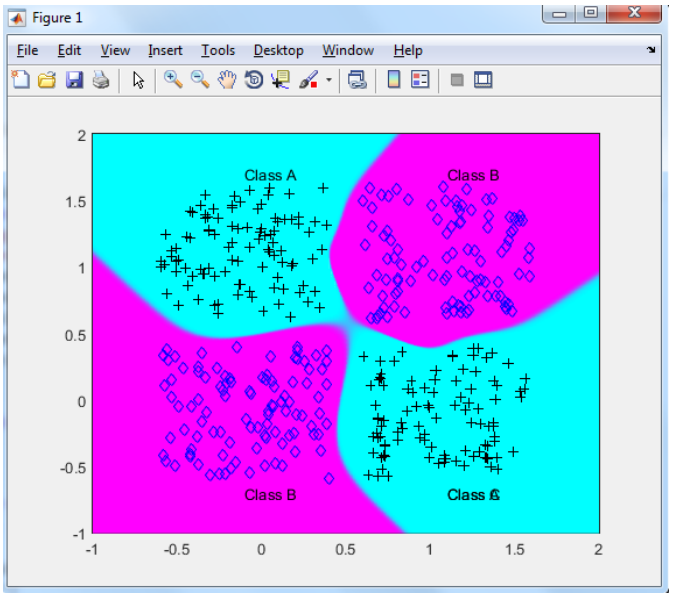


Рисунок 3.28 – Получившаяся граница решения для задачи XOR

Примечание. Сеть со структурой 2-5-3-1 явно избыточна для решения данной задачи. Можно переопределить структуру сети, как было показано в начале данной лабораторной работы:

```
net.layers{1}.size = 5;
% Получаем двухслойную сеть, где скрытый слой имеет 5 нейронов
net = configure(net, P, T);
% Меняем функцию активацию на выходном нейроне на нелинейную
net.layers{2}.transferFcn = 'tansig';
net.divideParam.trainRatio = 1;
net.divideParam.valRatio = 0;
net.divideParam.testRatio = 0;
[net, tr, Y, E] = train(net, P, T);
```

Структура такой сети показана на рисунке 3.29.

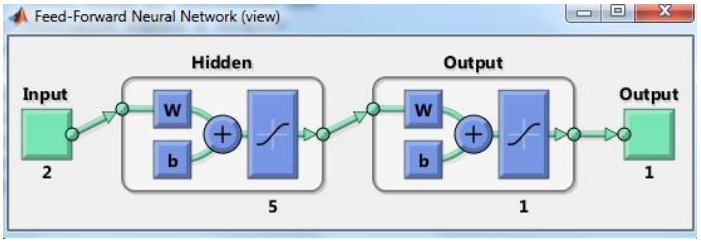


Рисунок 3.29 – Данная сеть тоже справится с задачей XOR

Однако, как видно из рисунка 3.30, ошибка обучения опустится в район 10^{-9}, что больше, чем у первой сети, что в целом хуже.

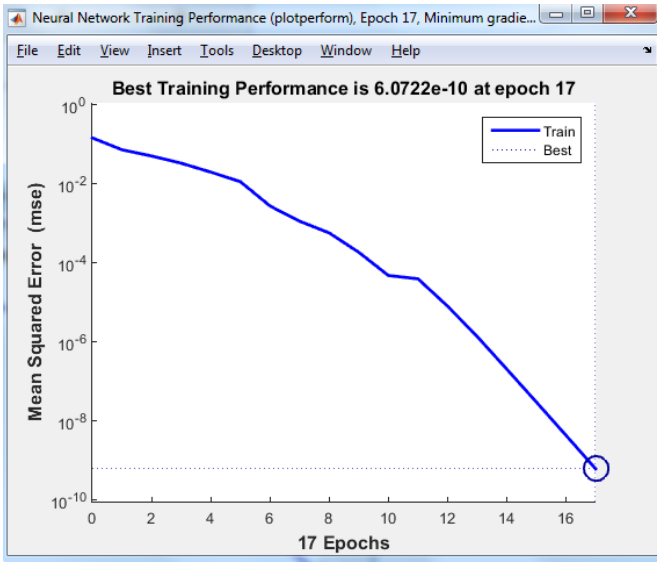


Рисунок 3.30 – Ошибка обучения для сети 2-5-1

Граница решений для этой сети показана на рисунке 3.31.

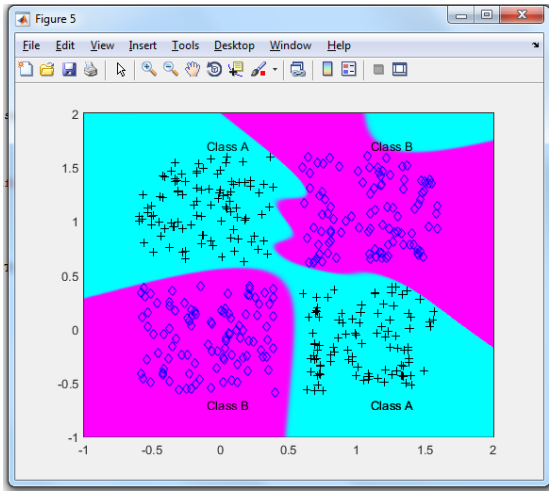


Рисунок 3.31 – Граница решений для сети с пятью скрытыми нейронами

Сеть со структурой 2-3-1 даст ошибку, показанную на рисунке 3.32.

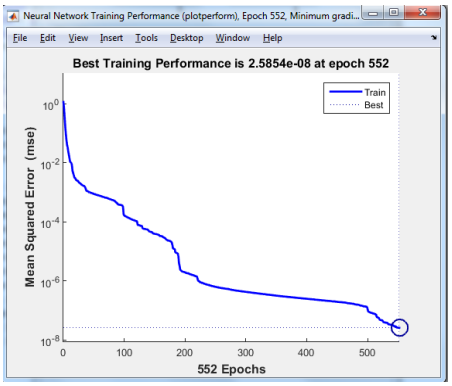


Рисунок 3.32 – Ошибка обучения для сети 2-3-1

Получившаяся граница решений для сети с тремя скрытыми нейронами показана на рисунке 3.33.

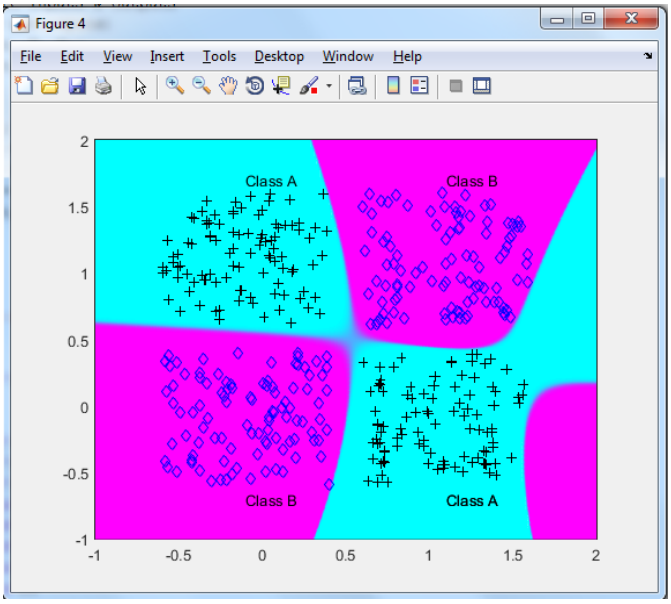


Рисунок 3.33 – Граница решений для сети из трёх нейронов в скрытом слое

Можно сделать вывод, что чем меньше количество нейронов в скрытом слое, тем грубее будут разделены классы и граница решений может в некоторых местах принимать хаотический вид (рисунок 3.31).

Конечно, такой разброс в решениях получается вследствие лёгкой разделимости классов.

Теперь применим сеть прямого распространения к разделению 4-х классов (ранее решали эту задачу с помощью персептрона).

```
close all, clear all, clc, format compact
K = 100;
q = .6;
A = [rand(1, K)-q; rand(1, K)+q];
B = [rand(1, K)+q; rand(1, K)+q];
C = [rand(1, K)+q; rand(1, K)-q];
D = [rand(1, K)-q; rand(1, K)-q];
figure(1);
plot(A(1, :), A(2, :), 'k+');
hold on;
grid on;
plot(B(1, :), B(2, :), 'b*');
plot(C(1, :), C(2, :), 'kx');
plot(D(1, :), D(2, :), 'bd');
text(.5-q, .5+2*q, 'Class A');
text(.5+q, .5+2*q, 'Class B');
text(.5+q, .5-2*q, 'Class C');
text(.5-q, .5-2*q, 'Class D');
% Кодируем классы
a = [-1 -1 -1 +1]';
b = [-1 -1 +1 -1]';
c = [-1 +1 -1 -1]';
d = [+1 -1 -1 -1]';
P = [A B C D];
T = [repmat(a, 1, length(A)) repmat(b, 1, length(B)) repmat(c, 1, length(C)) repmat(d, 1, length(D))];
net =feedforwardnet([4 3]);
net.divideParam.trainRatio = 1;
net.divideParam.valRatio = 0;
net.divideParam.testRatio = 0;

[net, tr, Y, E] = train(net, P, T);
view(net);
% Получаем в “m” все максимальные элементы по столбцам, а в “i”
% их строковые индексы
[m, i] = max(T);
% Получаем в j индексы строк максимальных элементов
[m, j] = max(Y);
N =  length(Y);
k = 0;
```

```
% Сравниваем эти индексы, они должны совпасть, если они не
% совпадают, то увеличиваем k на 1, т.е. k будет содержать количество
% ответов ИНС, которые не совпадают с метками
if find(i - j),
        k = length(find(i-j));
end
% Высчитываем это количество в процентах
fprintf('Correct classified samples: %.if%% samples\n', 100*(N-k)/N);
figure;
% Строим верхний график на канве
subplot(211);
plot(T');
title('Targets');
ylim([-2 2]);
grid on;
% Строим нижний график на канве
subplot(212);
plot(Y');
title('Network response');
xlabel('# sample');
ylim([-2 2]);
grid on;
span = -1:.01:2;
[P1, P2] = meshgrid(span, span);
pp = [P1(:) P2(:)]';
aa = net(pp);
figure(1);
% Наносим на сетку все ответы первого нейрона для
% всей выборки, они должны занять ¼ от площади всех классов
m = mesh(P1, P2, reshape(aa(1, :), length(span), length(span))-5);
set(m, 'facecolor', [1 0.2 .7], 'linestyle', 'none');
hold on;
% Тоже самое делаем для других нейронов
m = mesh(P1, P2, reshape(aa(2, :), length(span), length(span))-5);
set(m, 'facecolor', [1 1.0 0.5], 'linestyle', 'none');
m = mesh(P1, P2, reshape(aa(3, :), length(span), length(span))-5);
set(m, 'facecolor', [.4 1.0 0.9], 'linestyle', 'none');
m = mesh(P1, P2, reshape(aa(4, :), length(span), length(span))-5);
set(m, 'facecolor', [.3 .4 0.5], 'linestyle', 'none');
view(2);
```

Создаём 4 класса, каждый представлен 100 паттернами, рисунок 3.34.

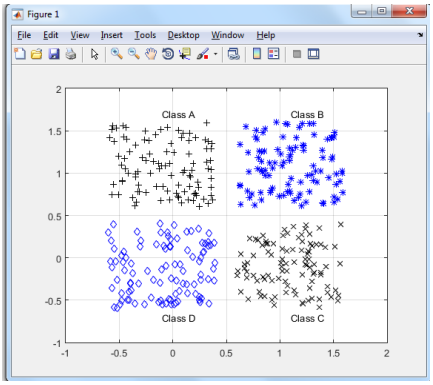


Рисунок 3.34 – 4 класса, которые нужно разделить

Далее создаём сеть со структурой 2-4-3-4 и обучаем её, рисунки 3.35, 3.36, 3.37.

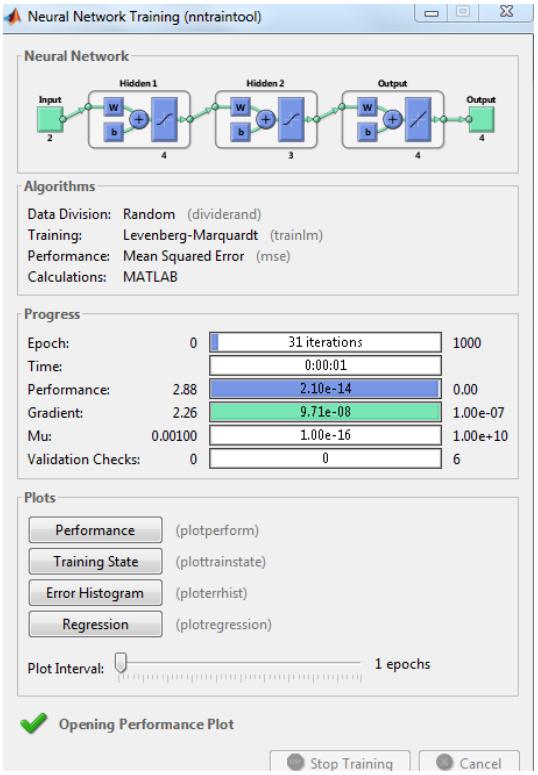


Рисунок 3.35 – Обучение сети, потребовалось 31 итерация

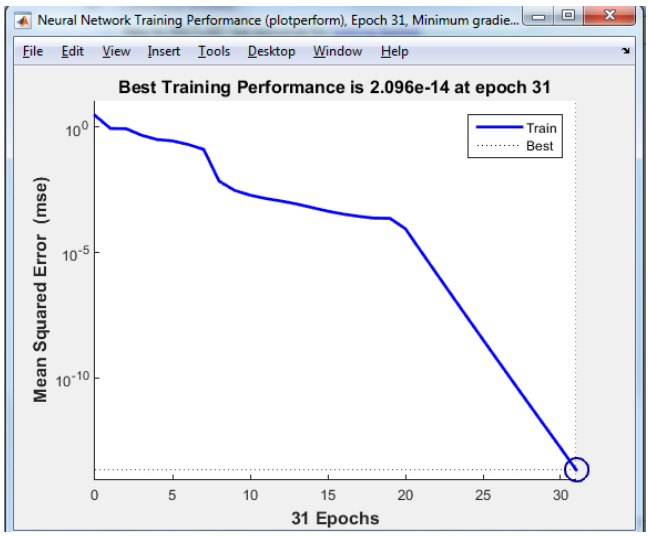


Рисунок 3.36 – Ошибка обучения очень хорошая, ушла ниже 10^{-10}

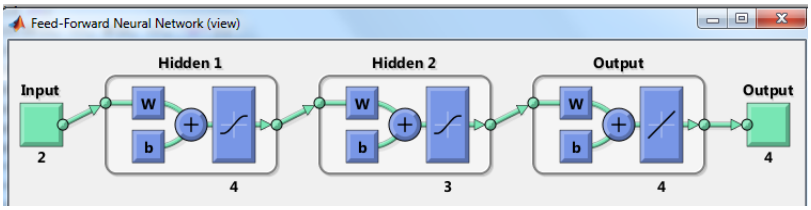


Рисунок 3.37 – Общая структура сети, которая использовалась

Разумеется, предложенная структура не единственная, на самом деле хватило бы сети с одним скрытым слоем и четырьмя нелинейными нейронами на выходном слое.

Метки и ответы сети изображены на рисунке 3.38, видно, что графики полностью совпадают, т.е. сеть точно моделирует все реакции на входные значения.

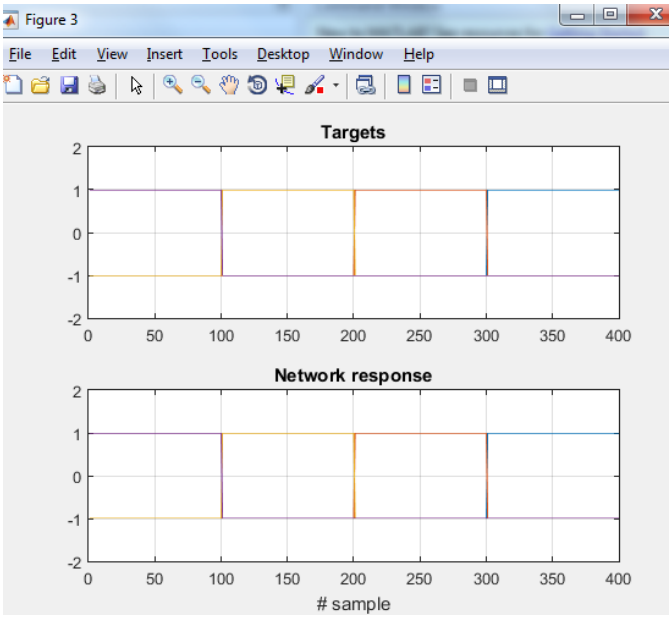


Рисунок 3.38 – График ожидаемых и реальных ответов ИНС

После построения границ решения можно видеть, что каждый класс отделён от другого, рисунок 3.39.

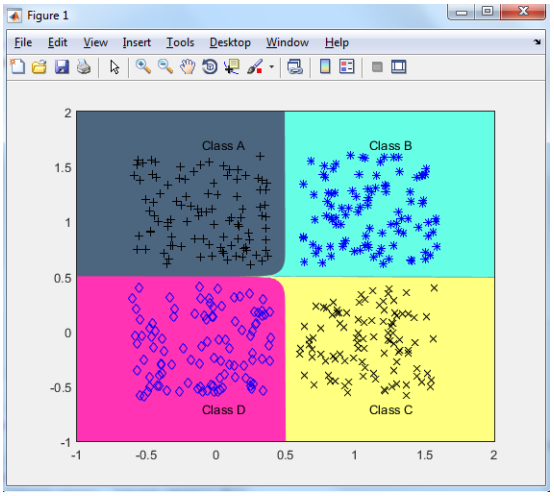


Рисунок 3.39 – Границы решения для задачи по разделению четырёх классов с помощью сети прямого распространения

Если сравнить это решение с тем, которое было достигнуто с помощью персептрона, то станет очевидным, что возможности по отделению одних классов от других у ИНС прямого распространения значительно выше, чем у персептрона, т.к. такие сети позволяют строить более сложные границы.

Аппаратура и материалы. 64-разрядный (x64) персональный компьютер, процессор с тактовой частотой 1 ГГц и выше, оперативная память 1 Гб и выше, свободное дисковое пространство не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система Windows 7 и выше, Matlab (R2013) и выше.

Указание по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

Методика и порядок выполнения работы

В индивидуальном варианте (таблица 3.1) дано расположение двух классов (A и B) по «углам» квадрата (по типу как на рисунке 3.39). Каждый «угол» представлен 200 точками. Требуется отделить класс A от B с помощью персептрона и полносвязанной

многослойной сети прямого распространения.

Таблица 3.1 – Варианты заданий

Номер варианта	Условие задачи
1	$\begin{pmatrix} B & A \\ B & B \end{pmatrix}$
2	$\begin{pmatrix} A & B \\ B & B \end{pmatrix}$
3	$\begin{pmatrix} B & B \\ A & B \end{pmatrix}$
4	$\begin{pmatrix} B & B \\ B & A \end{pmatrix}$
5	$\begin{pmatrix} A & A \\ B & B \end{pmatrix}$
6	$\begin{pmatrix} A & B \\ A & B \end{pmatrix}$
7	$\begin{pmatrix} B & B \\ A & A \end{pmatrix}$
8	$\begin{pmatrix} B & A \\ B & A \end{pmatrix}$
9	$\begin{pmatrix} A & B \\ B & A \end{pmatrix}$
10	$\begin{pmatrix} B & A \\ A & B \end{pmatrix}$
11	$\begin{pmatrix} A & A \\ B & A \end{pmatrix}$
12	$\begin{pmatrix} B & A \\ A & A \end{pmatrix}$
13	$\begin{pmatrix} A & B \\ A & A \end{pmatrix}$
14	$\begin{pmatrix} A & A \\ A & B \end{pmatrix}$

Содержание отчета и его форма

Отчёт по лабораторной работе должен содержать следующую информацию:

1. Название лабораторной работы и её номер.
2. ФИО студента и группу.

3. Формулировка индивидуального задания.
4. Документ отчёта с Print Prtscr диалоговых окон по шагам для своего варианта по подобию того, что описано в теоретической части. Обязательно должна присутствовать раскрашенная форма с финальным разделением для сети и цветные линии, отделяющие один класс от другого для персептрона. Должен присутствовать ступенчатый график, показывающий качество работы сети.
5. Ответы на контрольные вопросы.

Вопросы для защиты работы

1) Как кодируются классы для персептронов или сетей прямого распространения?

2) Опишите последовательность действий для нанесения входных значений на канву, чтобы визуально было видно расположение классов относительно друг друга.

3) Опишите последовательность действий для построения на канве границ решения задачи классификации.

4) Приведите пример сети прямого распространения с одним скрытым слоем, с помощью которой можно решить задачу о разделении 4-х классов в рассмотренной лабораторной работе.

5) Почему с помощью сетей прямого распространения можно лучшесправляться с задачами классификации, чем с помощью персептронов?

Лабораторная работа 3. Пространственная фильтрация изображений

Цель: изучить методы пространственной фильтрации

Формируемые компетенции: ПК-11

Теоретическая часть

Методы обработки изображений с точки зрения реализации делятся на два основных класса – локальные и глобальные. Каждый из них имеет свои преимущества и недостатки. Преимущество методов при глобальной реализации заключается в простоте их исполнения и быстродействии. Локальные методы владеют более широкими функциональными возможностями, в частности, они могут учитывать характеристикилокальных областей, т.е. быть адаптивными.

Реализация локальных методов состоит из четырех основных этапов:

1. формирование локальной окрестности и ее центральной точки;
2. проведение операции на пикселями данной локальной окрестности;
3. присвоение результата операции (п. 2) центральной точке окрестности;
4. повторение операций, описанных в п.п. 1 – 3, для каждой точки изображения.

Далее приведем примеры применения некоторых видов пространственных фильтров.

Сначала считаем исходное изображение. Для лучшего понимания сути приведем реализацию метода для одной цветовой составляющей.

```
L = imread('G:\РАБОТА\2018\Digital Image Processing\Лабораторный практикум\К лабе 3\TheWoman.jpg');
L=L(:,:,1);
figure, imshow(L);
```

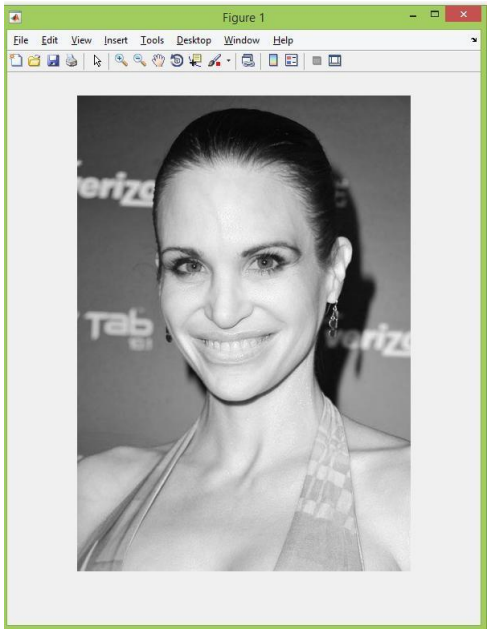


Рисунок 3.1 – Исходное изображение в оттенках серого

Испортим исходное изображение шумом типа «соль и перец».

```
L=imnoise(L,'salt & pepper',0.01);
figure, imshow(L);
```

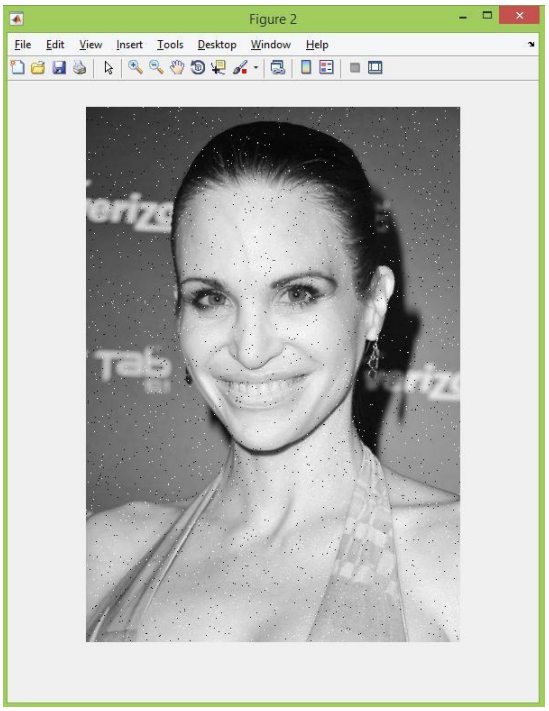


Рисунок 3.2 – Изображение с шумом «соль и перец»

Далее применим к этому зашумленному изображению различные виды пространственных фильтров и проанализируем результат обработки.

Прежде чем приступать к реализации локальных методов, необходимо расширить границы исходного изображения. Размеры расширения зависят от размеров маски фильтра. Если проводить фильтрацию с помощью встроенной функции **imfilter**, то параметры расширения можно задавать с помощью опций:

- **P** – границы изображения расширяются значением P. По умолчанию значение P=0;
- **replicate** – размер изображения увеличивается повторением величин на его боковых границах;
- **symmetric** – размер изображения увеличивается путем зеркального отражения через границы;
- **circular** – размер изображения увеличивается периодическим повторением двумерной функции;

В этом примере будем использовать не встроенную функцию **imfilter** и опции расширения, а приведем детальное описание работы нескольких пространственных фильтров. Размер изображения увеличим путем зеркального отражения через границы.

```
L=double(L)./255;

% инициализируем некоторые константы для работы алгорима
q=1.5;
n=3;
m=n;
[N M]=size(L);
n1=fix(n/2); % при n=m=3, n1=m1=1
m1=fix(m/2);
a=L(1,1); % значение левого верхнего угла
b=L(1,M); % значение правого верхнего угла
c=L(N,1); % значение нижнего левого угла
d=L(N,M); % значение правого нижнего угла

% при размерах локальной окрестности n=m=3
% L1, L3, L6, L8 получат значения этих переменных
for i=1:n1;
  for j=1:m1;
    L1(i,j)=a;
```

```
    L3(i,j)=b;
    L6(i,j)=c;
    L8(i,j)=d;
  end;
end;


L2=L(1,1:M);   %верхняя строка
L02=L2;
for i=1:n1-1; % цикл не выполнится
        L2=[L2;L02];
end;

L7=L(N,1:M);    % нижняя строка
L07=L7;
for i=1:n1-1;
        L7=[L7;L07];
end;

L4=L(1:N,1);     % левый столбец
L4=L4';     % превращаем его действительно в столбец
L04=L4;
for i=1:m1-1;
        L4=[L4;L04];
end;
L4=L4';

L5=L(1:N,M); % правый столбец
L5=L5';
L05=L5;
for i=1:m1-1;
        L5=[L5;L05];
end;
L5=L5';

% нижний код выполняется для того, чтобы получить расширенную
% матрицу Lr на 2 строки и 2 столбца больше, чем предыдущая
L1=[L1;L4];
L1=[L1;L6];
L1=L1';
L2=[L2;L];
L2=[L2;L7];
L2=L2';
L3=[L3;L5];
L3=[L3;L8];
```

```
L3=L3';
L1=[L1;L2];
L1=[L1;L3];
Lr=L1';
clear L1; clear L2; clear L3; clear L4;
clear L5;  clear L6;  clear L7;  clear L8;
```

Приведем программную реализацию непосредственно самих методов.

```
% основной код программы

for i=1+n1 : N+n1; % от второй строки до предпоследней
      for j=1+m1 : M+m1; % от второго столбца до предпоследнего
            if j==1+m1; % если столбец предельный, то
              D=0;
              for a=-n1:n1; % формирование окрестности
                  for b=-m1:m1;
                        D(n1+1+a,m1+1+b)=Lr(i+a,j+b);
                  end;
              end;
            end;

            if j>1+m1; % если столбец обычный
                  for a=-n1:n1;
                        D(n1+1+a,m+1)=Lr(i+a,j+m1);
                  end;
                  D=D(1:n,2:m+1);
            end;

            % после формирования окрестности пользуемся разными
            % методами
            Lout1(i,j)=sum(sum(D.^(q+1)))/sum(sum(D.^q+eps));
            %Контргармоническое среднее
            Lout2(i,j)=sum(sum(D))./(m*n);
            %Арифметическое среднее
            Lout3(i,j)=prod(prod(D)).^(1/(m*n));
            %Геометрическое среднее
            Lout4(i,j)=(m*n)/sum(sum(1./(D+eps)));
            %Гармоническое среднее
            Lout5(i,j)=median(median(D));
            %медиана
            Lout6(i,j)=max(D(:));          %max
            Lout7(i,j)=min(D(:));          %min
            Lout8(i,j)=0.5*(max(D(:))+min(D(:)));
            %срединная точка
```

```
            d=6;
            Dus=sort(D(:));
            Dus=Dus(d/2+1:end-d/2);
            Lout9(i,j)=mean(Dus);  %a–усеченное среднее
        end;
end;


% Приведения изображений к исходным размерам
Lout1=Lout1(n1+1:N+n1,m1+1:M+m1);
Lout2=Lout2(n1+1:N+n1,m1+1:M+m1);
Lout3=Lout3(n1+1:N+n1,m1+1:M+m1);
Lout4=Lout4(n1+1:N+n1,m1+1:M+m1);
Lout5=Lout5(n1+1:N+n1,m1+1:M+m1);
Lout6=Lout6(n1+1:N+n1,m1+1:M+m1);
Lout7=Lout7(n1+1:N+n1,m1+1:M+m1);
Lout8=Lout8(n1+1:N+n1,m1+1:M+m1);
Lout9=Lout9(n1+1:N+n1,m1+1:M+m1);

figure, imshow(Lout1);title('Контргармоническое среднее');
figure, imshow(Lout2);title('Арифметическое среднее');
figure, imshow(Lout3);title('Геометрическое среднее');
figure, imshow(Lout4);title('Гармоническое среднее');
figure, imshow(Lout5);title('Медиана');
figure, imshow(Lout6);title('Максимум по окрестности');
figure, imshow(Lout7);title('Минимум по окрестности');
figure, imshow(Lout8);title('Срединная точка');
figure, imshow(Lout9);title('a-усеченное среднее');
```

Результаты работы фильтров представлены на рисунках 3.3 – 3.11. Для более подробного изучения фильтрации можно обратиться к [1] главе 3.4: «Основы пространственной фильтрации».

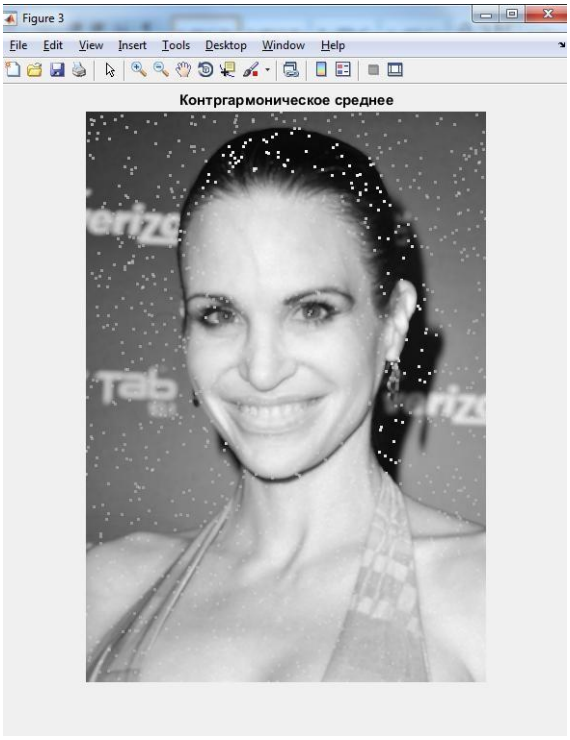


Рисунок 3.3 – Контргармоническое среднее

Как видно контргармоническое среднее не очень хорошо справилось с задачей. При варьировании параметра q, результат не особенно улучшиться.

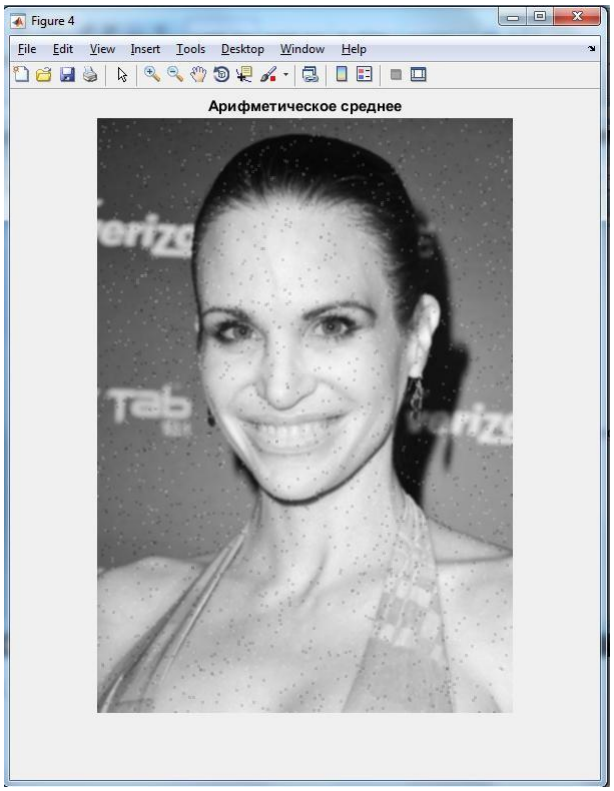


Рисунок 3.4 – Арифметическое среднее

Арифметическое среднее значительно лучше справилось с работой, если увеличить размер окна с 3 до 9, то результат будет ещѐ лучше. Недостаток этого фильтра состоит в том, что его применение приводит не только к устранению шумовой составляющей, но и размывает границы объектов, т.е. понижает качество визуального восприятия такихизображений.

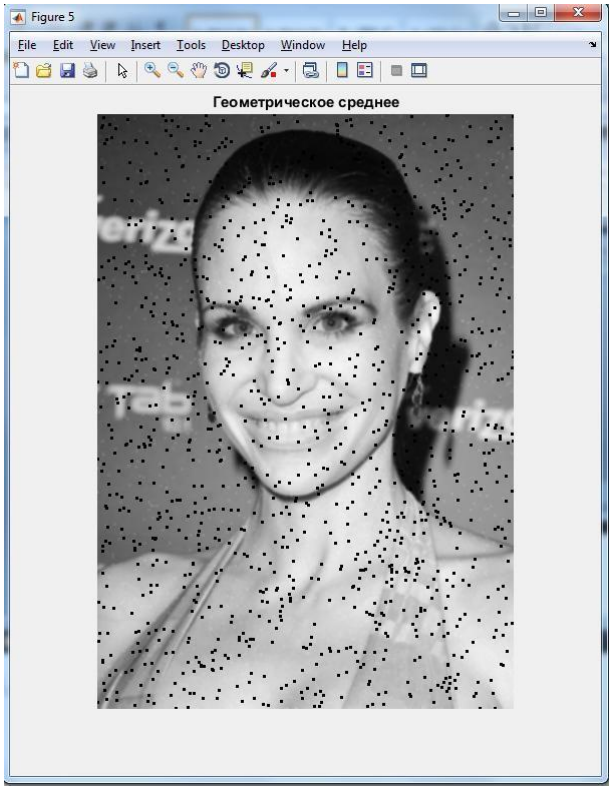


Рисунок 3.5 – Геометрическое среднее

Рисунок 3.6 – Гармоническое среднее

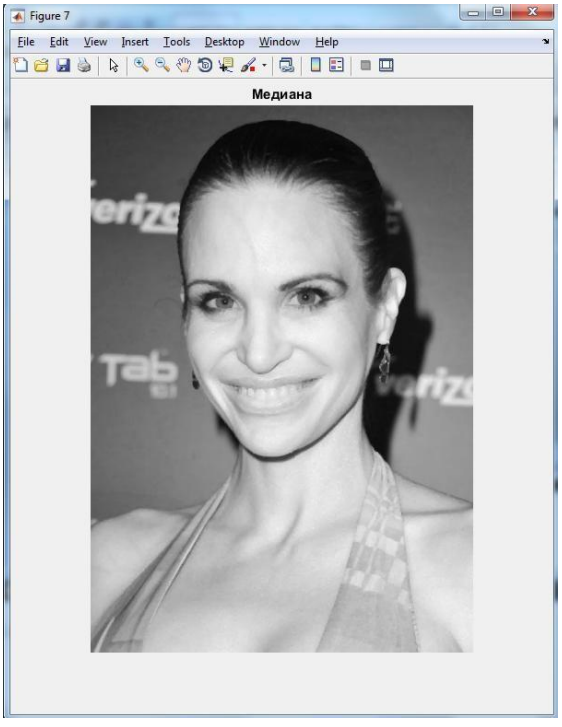


Рисунок 3.7 – Медиана

Для устранения данного вида шума («соль и перец») медианная фильтрация является одним из наиболее простых и эффективных методов. Размер локальной апертуры влияет на результат обработки, но не в такой

степени как в других рассмотренных нами методах. Одно из основных преимуществ медианной фильтрации состоит в том, что она приводит к устранению импульсных выбросов, не размывая границ объектов.

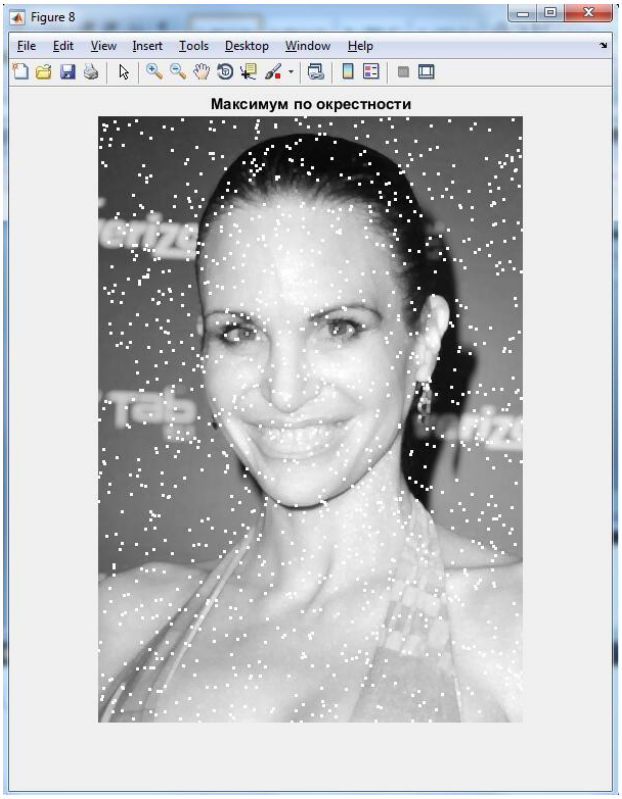


Рисунок 3.8 – Максимум по окрестности

Рисунок 3.9 – Минимум по окрестности

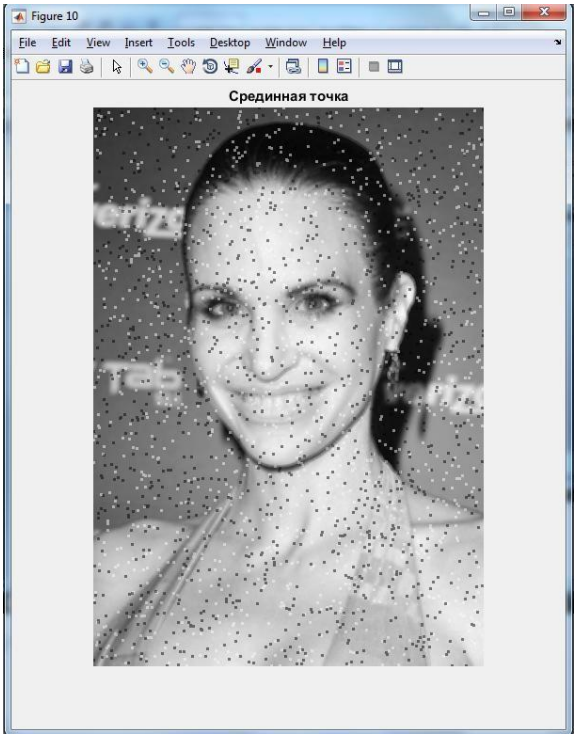


Рисунок 3.10 – Срединная точка

Рисунок 3.11 – a-усеченное среднее

Также очень качественный фильтр для устранения этого типа шума.

Полный код программы приводится ниже.

```
L = imread('G:\РАБОТА\2018\Digital Image Processing\Лабораторный практикум\К лабе 3\TheWoman.jpg');
L=L(:,:,1);
figure, imshow(L);
L=imnoise(L,'salt & pepper',0.01);
figure, imshow(L);
L=double(L)./255;

% инициализируем некоторые константы для работы алгорима
q=1.5;
n=3;
m=n;
[N M]=size(L);
n1=fix(n/2); % при n=m=3, n1=m1=1
m1=fix(m/2);
a=L(1,1); % значение левого верхнего угла
b=L(1,M); % значение правого верхнего угла
c=L(N,1); % значение нижнего левого угла
d=L(N,M); % значение правого нижнего угла

% при размерах локальной окрестности n=m=3
% L1, L3, L6, L8 получат значения этих переменных
for i=1:n1;
  for j=1:m1;
    L1(i,j)=a;
    L3(i,j)=b;
    L6(i,j)=c;
    L8(i,j)=d;
  end;
end;


L2=L(1,1:M);   %верхняя строка
L02=L2;
for i=1:n1-1; % цикл не выполнится
        L2=[L2;L02];
end;

L7=L(N,1:M);    % нижняя строка
```

```
L07=L7;
for i=1:n1-1;
        L7=[L7;L07];
end;

L4=L(1:N,1);     % левый столбец
L4=L4';     % превращаем его действительно в столбец
L04=L4;
for i=1:m1-1;
        L4=[L4;L04];
end;
L4=L4';

L5=L(1:N,M); % правый столбец
L5=L5';
L05=L5;
for i=1:m1-1;
        L5=[L5;L05];
end;
L5=L5';

% нижний код выполняется для того, чтобы получить расширенную
% матрицу Lr на 2 строки и 2 столбца больше, чем предыдущая
L1=[L1;L4];
L1=[L1;L6];
L1=L1';
L2=[L2;L];
L2=[L2;L7];
L2=L2';
L3=[L3;L5];
L3=[L3;L8];
L3=L3';
L1=[L1;L2];
L1=[L1;L3];
Lr=L1';
clear L1; clear L2; clear L3; clear L4;
clear L5;  clear L6;  clear L7;  clear L8;

% основной код программы

for i=1+n1 : N+n1; % от второй строки до предпоследней
        for j=1+m1 : M+m1; % от второго столбца до предпоследнего
                if j==1+m1; % если столбец предельный, то
                   D=0;
                   for a=-n1:n1; % формирование окрестности
```

```
                for b=-m1:m1;
                    D(n1+1+a,m1+1+b)=Lr(i+a,j+b);
                end;
            end;
        end;

        if j>1+m1; % если столбец обычный
            for a=-n1:n1;
                D(n1+1+a,m+1)=Lr(i+a,j+m1);
            end;
            D=D(1:n,2:m+1);
        end;

        % после формирования окрестности пользуемся разными
        % методами
        Lout1(i,j)=sum(sum(D.^(q+1)))/sum(sum(D.^q+eps));
        %Контргармоническое среднее
        Lout2(i,j)=sum(sum(D))./(m*n);
        %Арифметическое среднее
        Lout3(i,j)=prod(prod(D)).^(1/(m*n));
        %Геометрическое среднее
        Lout4(i,j)=(m*n)/sum(sum(1./(D+eps)));
        %Гармоническое среднее
        Lout5(i,j)=median(median(D));
        %медиана
        Lout6(i,j)=max(D(:));          %max
        Lout7(i,j)=min(D(:));          %min
        Lout8(i,j)=0.5*(max(D(:))+min(D(:)));
        %срединная точка
         d=6;
        Dus=sort(D(:));
        Dus=Dus(d/2+1:end-d/2);
        Lout9(i,j)=mean(Dus);  %a–усеченное среднее
    end;
end;


% приведения изображений к исходным размерам
Lout1=Lout1(n1+1:N+n1,m1+1:M+m1);
Lout2=Lout2(n1+1:N+n1,m1+1:M+m1);
Lout3=Lout3(n1+1:N+n1,m1+1:M+m1);
Lout4=Lout4(n1+1:N+n1,m1+1:M+m1);
Lout5=Lout5(n1+1:N+n1,m1+1:M+m1);
Lout6=Lout6(n1+1:N+n1,m1+1:M+m1);
Lout7=Lout7(n1+1:N+n1,m1+1:M+m1);
Lout8=Lout8(n1+1:N+n1,m1+1:M+m1);
```

```
Lout9=Lout9(n1+1:N+n1,m1+1:M+m1);

figure, imshow(Lout1);title('Контргармоническое среднее');
figure, imshow(Lout2);title('Арифметическое среднее');
figure, imshow(Lout3);title('Геометрическое среднее');
figure, imshow(Lout4);title('Гармоническое среднее'); figure,
imshow(Lout5);title('Медиана');
figure, imshow(Lout6);title('Максимум по окрестности');
figure, imshow(Lout7);title('Минимум по окрестности');
figure, imshow(Lout8);title('Срединная точка');
figure, imshow(Lout9);title('a-усеченное среднее');
```

Практическая часть

Индивидуальное задание приведено в таблице 3.1. Используя указанный фильтр, любую картинку и шум «соль & перец» более крупного размера, чем в примере, варьируя параметры постараться добиться максимального результата и объяснить почему. При объяснении привести формулу фильтра и показать, как разные параметры влияют на результат. Изображение расширять не кустарным способом, а с использованием**imfilter()**.

Таблица 3.1 – Задание по вариантам

1	Арифметическое среднее
2	Геометрическое среднее
3	Гармоническое среднее
4	Медиана
5	Максимум по окрестности
6	Минимум по окрестности
7	Срединная точка
8	А-усеченное среднее
9	Контргармоническое среднее

Лабораторная работа № 5

Пример создания и обучения нейронных сетей для задач классификации в среде Matlab. Часть 1

Цель и содержание работы: создать и обучить несколько нейронных сетей, решающих задачи классификации на примере задачи ирисов Фишера и двух спиралей.

Задачи:

- разобрать решение задачи об ирисах Фишера;
- научиться обучать сети с разной архитектурой и отбирать из них наилучшие на примере задачи об ирисах;
- разобрать решение задачи двух спиралей;
- научиться подбирать ряд параметров для архитектуры нейронных сетей на основе задаче о двух спиралях;
- научиться создавать сети с помощью разных функций Matlab и обучать их с помощью разных алгоритмов на основе задачи о двух спиралях;
- разобрать код обратного распространения ошибки для задачи о двух спиралях.

Теоретическое обоснование

В предыдущей лабораторной работе рассматривались очень лёгкие задачи классификации, причём тестовое и валидационное множество отсутствовали. В этой лабораторной работе будут рассмотрены два реальных примера задачи классификации.

Первая задача – это задача ирисов Фишера. Ирисы Фишера состоят из данных о 150 экземплярах ириса, по 50 экземпляров из трёх видов – Ирис щетинистый (Iris setosa), Ирис виргинский (Iris virginica), Ирис разноцветный (Iris versicolor), рисунок 5.1.

Рисунок 5.1 – Слева направо: Iris versicolor, Iris virginica, Iris setosa

Каждый экземпляр того или иного ириса характеризовался четырьмя числами, которые выражали определённые характеристики цветков: длина наружной доли околоцветника (sepal length), ширина наружной доли околоцветника (sepal width), длина

внутренней доли околоцветника (petal length), ширина внутренней доли околоцветника(petal width); всё в сантиметрах. На основании этого набора данных требуется построить правило классификации, определяющее вид растения по данным измерений. Это задача многоклассовой классификации, т.к. имеются три вида ириса.

Все данные для данной задачи приведены в таблице 5.1. Именно в подобном виде записывается большинство задач классификации. Стоит обратить внимание, что данные приведены в вещественном формате, где разделителем служит запятая, а не точка. Это сделано для того, чтобы Excel правильно интерпретировал вещественные числа, т.к. в этой среде разделитель – запятая, а точка обычно используется для записи даты.

Таблица 5.1 – Данные для задачи ирисов Фишера

№	Sep	Sepa	Petal	Petal	Species	№	Sepa	Sepa	Petal	Petal	Species
1	5,1	3,5	1,4	0,2	*I, setosa*	76	6,6	3,0	4,4	1,4	*I. versicol*
2	4,9	3,0	1,4	0,2	*I, setosa*	77	6,8	2,8	4,8	1,4	*I. versicol*
3	4,7	3,2	1,3	0,2	*I, setosa*	78	6,7	3,0	5,0	1,7	*I. versicol*
4	4,6	3,1	1,5	0,2	*I, setosa*	79	6,0	2,9	4,5	1,5	*I. versicol*
5	5,0	3,6	1,4	0,2	*I, setosa*	80	5,7	2,6	3,5	1,0	*I. versicol*
6	5,4	3,9	1,7	0,4	*I, setosa*	81	5,5	2,4	3,8	1,1	*I. versicol*
7	4,6	3,4	1,4	0,3	*I, setosa*	82	5,5	2,4	3,7	1,0	*I. versicol*
8	5,0	3,4	1,5	0,2	*I, setosa*	83	5,8	2,7	3,9	1,2	*I. versicol*
9	4,4	2,9	1,4	0,2	*I, setosa*	84	6,0	2,7	5,1	1,6	*I. versicol*
10	4,9	3,1	1,5	0,1	*I, setosa*	85	5,4	3,0	4,5	1,5	*I. versicol*
11	5,4	3,7	1,5	0,2	*I, setosa*	86	6,0	3,4	4,5	1,6	*I. versicol*
12	4,8	3,4	1,6	0,2	*I, setosa*	87	6,7	3,1	4,7	1,5	*I. versicol*
13	4,8	3,0	1,4	0,1	*I, setosa*	88	6,3	2,3	4,4	1,3	*I. versicol*
14	4,3	3,0	1,1	0,1	*I, setosa*	89	5,6	3,0	4,1	1,3	*I. versicol*
15	5,8	4,0	1,2	0,2	*I, setosa*	90	5,5	2,5	4,0	1,3	*I. versicol*
16	5,7	4,4	1,5	0,4	*I, setosa*	91	5,5	2,6	4,4	1,2	*I. versicol*
17	5,4	3,9	1,3	0,4	*I, setosa*	92	6,1	3,0	4,6	1,4	*I. versicol*
18	5,1	3,5	1,4	0,3	*I, setosa*	93	5,8	2,6	4,0	1,2	*I. versicol*
19	5,7	3,8	1,7	0,3	*I, setosa*	94	5,0	2,3	3,3	1,0	*I. versicol*
20	5,1	3,8	1,5	0,3	*I, setosa*	95	5,6	2,7	4,2	1,3	*I. versicol*
21	5,4	3,4	1,7	0,2	*I, setosa*	96	5,7	3,0	4,2	1,2	*I. versicol*
22	5,1	3,7	1,5	0,4	*I, setosa*	97	5,7	2,9	4,2	1,3	*I. versicol*
23	4,6	3,6	1,0	0,2	*I, setosa*	98	6,2	2,9	4,3	1,3	*I. versicol*
24	5,1	3,3	1,7	0,5	*I, setosa*	99	5,1	2,5	3,0	1,1	*I. versicol*
25	4,8	3,4	1,9	0,2	*I, setosa*	100	5,7	2,8	4,1	1,3	*I. versicol*
26	5,0	3,0	1,6	0,2	*I, setosa*	101	6,3	3,3	6,0	2,5	*I. virginic*
27	5,0	3,4	1,6	0,4	*I, setosa*	102	5,8	2,7	5,1	1,9	*I. virginic*
28	5,2	3,5	1,5	0,2	*I, setosa*	103	7,1	3,0	5,9	2,1	*I. virginic*
29	5,2	3,4	1,4	0,2	*I, setosa*	104	6,3	2,9	5,6	1,8	*I. virginic*
30	4,7	3,2	1,6	0,2	*I, setosa*	105	6,5	3,0	5,8	2,2	*I. virginic*
31	4,8	3,1	1,6	0,2	*I, setosa*	106	7,6	3,0	6,6	2,1	*I. virginic*
32	5,4	3,4	1,5	0,4	*I, setosa*	107	4,9	2,5	4,5	1,7	*I. virginic*
33	5,2	4,1	1,5	0,1	*I, setosa*	108	7,3	2,9	6,3	1,8	*I. virginic*
34	5,5	4,2	1,4	0,2	*I, setosa*	109	6,7	2,5	5,8	1,8	*I. virginic*
35	4,9	3,1	1,5	0,2	*I, setosa*	110	7,2	3,6	6,1	2,5	*I. virginic*

36	5,0	3,2	1,2	0,2	*I, setosa*	111	6,5	3,2	5,1	2,0	*I. virginic*
37	5,5	3,5	1,3	0,2	*I, setosa*	112	6,4	2,7	5,3	1,9	*I. virginic*
38	4,9	3,6	1,4	0,1	*I, setosa*	113	6,8	3,0	5,5	2,1	*I. virginic*
39	4,4	3,0	1,3	0,2	*I, setosa*	114	5,7	2,5	5,0	2,0	*I. virginic*
40	5,1	3,4	1,5	0,2	*I, setosa*	115	5,8	2,8	5,1	2,4	*I. virginic*
41	5,0	3,5	1,3	0,3	*I, setosa*	116	6,4	3,2	5,3	2,3	*I. virginic*
42	4,5	2,3	1,3	0,3	*I, setosa*	117	6,5	3,0	5,5	1,8	*I. virginic*
43	4,4	3,2	1,3	0,2	*I, setosa*	118	7,7	3,8	6,7	2,2	*I. virginic*
44	5,0	3,5	1,6	0,6	*I, setosa*	119	7,7	2,6	6,9	2,3	*I. virginic*
45	5,1	3,8	1,9	0,4	*I, setosa*	120	6,0	2,2	5,0	1,5	*I. virginic*
46	4,8	3,0	1,4	0,3	*I, setosa*	121	6,9	3,2	5,7	2,3	*I. virginic*
47	5,1	3,8	1,6	0,2	*I, setosa*	122	5,6	2,8	4,9	2,0	*I. virginic*
48	4,6	3,2	1,4	0,2	*I, setosa*	123	7,7	2,8	6,7	2,0	*I. virginic*
49	5,3	3,7	1,5	0,2	*I, setosa*	124	6,3	2,7	4,9	1,8	*I. virginic*
50	5,0	3,3	1,4	0,2	*I, setosa*	125	6,7	3,3	5,7	2,1	*I. virginic*
51	7,0	3,2	4,7	1,4	*I, versicol*	126	7,2	3,2	6,0	1,8	*I. virginic*
52	6,4	3,2	4,5	1,5	*I, versicol*	127	6,2	2,8	4,8	1,8	*I. virginic*
53	6,9	3,1	4,9	1,5	*I, versicol*	128	6,1	3,0	4,9	1,8	*I. virginic*
54	5,5	2,3	4,0	1,3	*I, versicol*	129	6,4	2,8	5,6	2,1	*I. virginic*
55	6,5	2,8	4,6	1,5	*I, versicol*	130	7,2	3,0	5,8	1,6	*I. virginic*
56	5,7	2,8	4,5	1,3	*I, versicol*	131	7,4	2,8	6,1	1,9	*I. virginic*
57	6,3	3,3	4,7	1,6	*I, versicol*	132	7,9	3,8	6,4	2,0	*I. virginic*
58	4,9	2,4	3,3	1,0	*I, versicol*	133	6,4	2,8	5,6	2,2	*I. virginic*
59	6,6	2,9	4,6	1,3	*I, versicol*	134	6,3	2,8	5,1	1,5	*I. virginic*
60	5,2	2,7	3,9	1,4	*I, versicol*	135	6,1	2,6	5,6	1,4	*I. virginic*
61	5,0	2,0	3,5	1,0	*I, versicol*	136	7,7	3,0	6,1	2,3	*I. virginic*
62	5,9	3,0	4,2	1,5	*I, versicol*	137	6,3	3,4	5,6	2,4	*I. virginic*
63	6,0	2,2	4,0	1,0	*I, versicol*	138	6,4	3,1	5,5	1,8	*I. virginic*
64	6,1	2,9	4,7	1,4	*I, versicol*	139	6,0	3,0	4,8	1,8	*I. virginic*
65	5,6	2,9	3,6	1,3	*I, versicol*	140	6,9	3,1	5,4	2,1	*I. virginic*
66	6,7	3,1	4,4	1,4	*I, versicol*	141	6,7	3,1	5,6	2,4	*I. virginic*
67	5,6	3,0	4,5	1,5	*I, versicol*	142	6,9	3,1	5,1	2,3	*I. virginic*
68	5,8	2,7	4,1	1,0	*I, versicol*	143	5,8	2,7	5,1	1,9	*I. virginic*
69	6,2	2,2	4,5	1,5	*I, versicol*	144	6,8	3,2	5,9	2,3	*I. virginic*
70	5,6	2,5	3,9	1,1	*I, versicol*	145	6,7	3,3	5,7	2,5	*I. virginic*
71	5,9	3,2	4,8	1,8	*I, versicol*	146	6,7	3,0	5,2	2,3	*I. virginic*
72	6,1	2,8	4,0	1,3	*I, versicol*	147	6,3	2,5	5,0	1,9	*I. virginic*
73	6,3	2,5	4,9	1,5	*I, versicol*	148	6,5	3,0	5,2	2,0	*I. virginic*
74	6,1	2,8	4,7	1,2	*I, versicol*	149	6,2	3,4	5,4	2,3	*I. virginic*
75	6,4	2,9	4,3	1,3	*I, versicol*	150	5,9	3,0	5,1	1,8	*I. virginic*

Данная выборка по ирисам является одной из классических задач классификации. Чтобы оценить сложность разделимости классов, можно отобразить точки в пространстве признаков. Так как каждый цветок характеризуется четырьмя числами, то искомая точка лежит в 4-х мерном пространстве, отобразить нельзя, но можно отобразить от двух и от трёх параметров. На рисунке 5.2 приведено расположение классов в зависимости от двух параметров, на рисунке 5.3 – от трёх параметров.

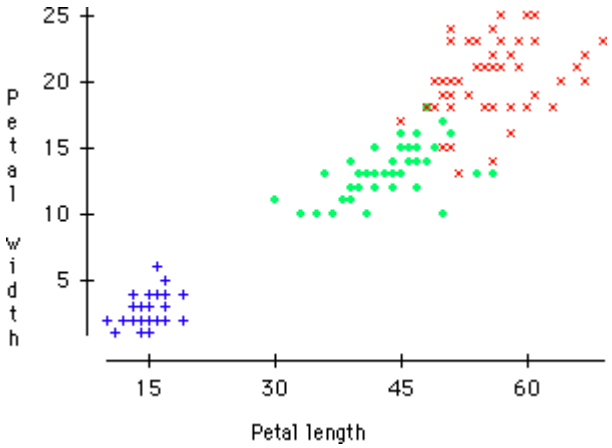


Рисунок 5.2 – Расположение экземпляров классов ирисов Фишера в зависимости от двух параметров: petal width и petal length

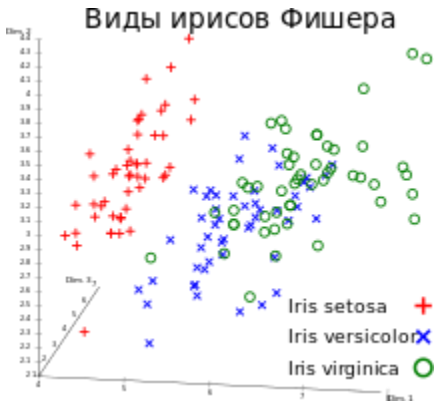


Рисунок 5.3 – Расположение экземпляров классов ирисов Фишера в зависимости от трёх параметров: petal width, petal length, sepal width

Из рисунков видно, что Iris setosa (на рисунке 5.2 – синие крестики, на рисунке 5.3 – красные крестики) линейно отделим от двух других классов, которые не слишком сильно смешаны между собой. Т.е. задача не слишком сложная, каждый класс имеет ярко выраженный собственный центр в пространстве признаков.

Вторая задача – это задача о спиралях, которые нужно разделить. Обычно имеется в виду два класса, экземпляр каждого класса – это точка в двумерном пространстве, каждый класс – это одна из непересекающихся спиралей, рисунок 5.4. Соответственно нужно отделить одну спираль от другой.

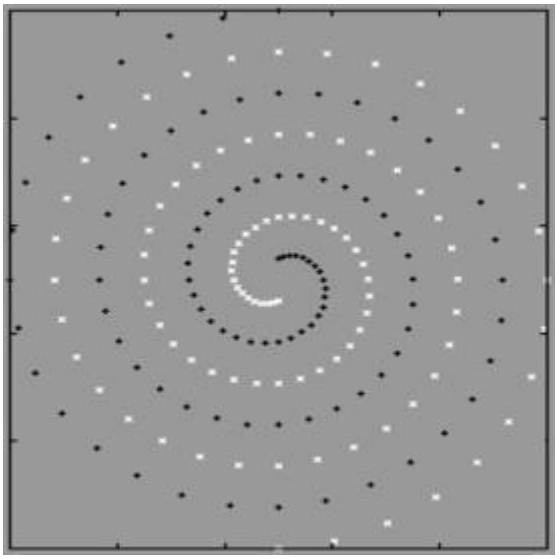


Рисунок 5.4 – Две непересекающиеся спирали

Эта задача – пример синтетических (сгенерированных человеком) данных. Есть разные варианты этой задачи: 3 спирали, 5 спиралей и т.д.

Пример пяти спиралей приведён на рисунке 5.5.

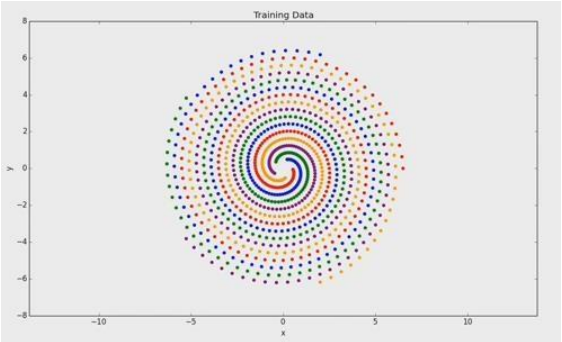


Рисунок 5.5 – Пять спиралей

Соответственно решение для подобных задач будет представлять из себя N спиралевидных поверхностей из сигмойд, на вершине каждой лежит нужный класс, а в ложбине все остальные классы. Пример такой поверхности для пяти спиралей показан на рисунке 5.6 (поверхность отображается в данном случае через ответы сети, покрашенные в разный цвет).

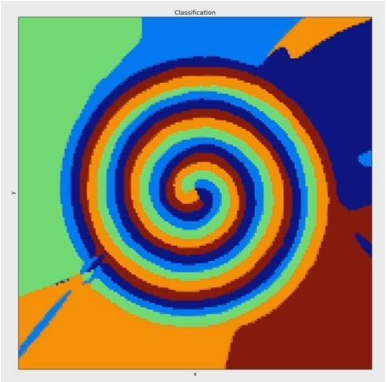


Рисунок 5.6 – Разделение пяти спиралей

На рисунке 5.7 показан пример поверхности отклика для двух спиралей, синий цвет – ложбина для первой спирали, красный цвет – возвышенность для второй спирали.

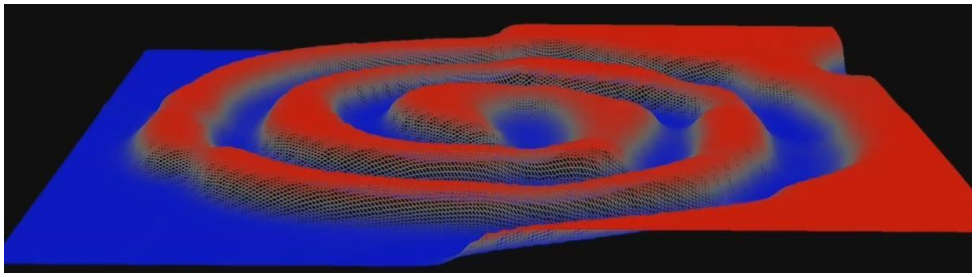


Рисунок 5.7 – Поверхность отклика для задачи о разделении двух спиралей

Для того, чтобы детально разобраться как происходит построение поверхности отклика смотрите приложение 1.

В отличие от задач первого класса, которые встречаются в реальной «природе», такие задачи для ИНС сложны и созданы главным образом для тестирования мощности той или иной сети (Так, для пяти спиралей требуется больше 16000 эпох обучения).

Задача о двух спиралях появилась впервые в [1]. В статье также приведён код на C по генерированию этого набора данных.

В таблице 5.2 приведены данные для задачи двух спиралей.

Таблица 5.2 – Данные для задачи о разделении двух спиралей

№	x	y	Class	№	x	y	Class
1	6,5	0	1	98	3,5	0,00008	-1
2	-6,5	0	-1	99	-3,37143	-0,6707	1
3	6,3138	1,2559	1	100	3,37143	0,6707	-1
4	-6,3138	-1,2559	-1	101	-3,11806	-1,29163	1
5	5,88973	2,43961	1	102	3,11806	1,29163	-1
6	-5,88973	-2,43961	-1	103	-2,7542	-1,84039	1
7	5,24865	3,50704	1	104	2,7542	1,84039	-1
8	-5,24865	-3,50704	-1	105	-2,29804	-2,29815	1

9	4,41941	4,41943	1	106	2,29804	2,29815	-1
10	-4,41941	-4,41943	-1	107	-1,77082	-2,65035	1
11	3,43758	5,14473	1	108	1,77082	2,65035	-1
12	-3,43758	-5,14473	-1	109	-1,19581	-2,88715	1
13	2,34392	5,65877	1	110	1,19581	2,88715	-1
14	-2,34392	-5,65877	-1	111	-0,59739	-3,00367	1
15	1,18272	5,94601	1	112	0,59739	3,00367	-1
16	-1,18272	-5,94601	-1	113	0,00008	-3	1
17	-0,00002	6	1	114	-0,00008	3	-1
18	0,00002	-6	-1	115	0,57315	-2,88104	1
19	-1,15837	5,82341	1	116	-0,57315	2,88104	-1
20	1,15837	-5,82341	-1	117	1,10029	-2,65612	1
21	-2,24829	5,42778	1	118	-1,10029	2,65612	-1
22	2,24829	-5,42778	-1	119	1,5626	-2,33847	1
23	-3,22928	4,8329	1	120	-1,5626	2,33847	-1
24	3,22928	-4,8329	-1	121	1,9446	-1,94449	1
25	-4,06589	4,06584	1	122	-1,9446	1,94449	-1
26	4,06589	-4,06584	-1	123	2,23462	-1,49303	1
27	-4,729	3,15978	1	124	-2,23462	1,49303	-1
28	4,729	-3,15978	-1	125	2,42521	-1,00447	1
29	-5,19684	2,15256	1	126	-2,42521	1,00447	-1
30	5,19684	-2,15256	-1	127	2,51328	-0,49985	1
31	-5,45563	1,08515	1	128	-2,51328	0,49985	-1
32	5,45563	-1,08515	-1	129	2,5	0,00007	1
33	-5,5	-0,00004	1	130	-2,5	-0,00007	-1
34	5,5	0,00004	-1	131	2,39065	0,4756	1
35	-5,33301	-1,06085	1	132	-2,39065	-0,4756	-1
36	5,33301	1,06085	-1	133	2,19419	0,90894	1
37	-4,96584	-2,05696	1	134	-2,19419	-0,90894	-1
38	4,96584	2,05696	-1	135	1,92273	1,28482	1
39	-4,41716	-2,95151	1	136	-1,92273	-1,28482	-1
40	4,41716	2,95151	-1	137	1,59094	1,59104	1
41	-3,71228	-3,71234	1	138	-1,59094	-1,59104	-1
42	3,71228	3,71234	-1	139	1,21525	1,81888	1
43	-2,88198	-4,31328	1	140	-1,21525	-1,81888	-1
44	2,88198	4,31328	-1	141	0,81314	1,96327	1
45	-1,9612	-4,7349	1	142	-0,81314	-1,96327	-1
46	1,9612	4,7349	-1	143	0,40231	2,02288	1
47	-0,98759	-4,96524	1	144	-0,40231	-2,02288	-1
48	0,98759	4,96524	-1	145	-0,00007	2	1
49	0,00006	-5	1	146	0,00007	-2	-1
50	-0,00006	5	-1	147	-0,37805	1,90026	1
51	0,96331	-4,84262	1	148	0,37805	-1,90026	-1
52	-0,96331	4,84262	-1	149	-0,71759	1,73225	1
53	1,86564	-4,50389	1	150	0,71759	-1,73225	-1
54	-1,86564	4,50389	-1	151	-1,00702	1,507	1
55	2,67373	-4,00141	1	152	1,00702	-1,507	-1
56	-2,67373	4,00141	-1	153	-1,23748	1,23739	1
57	3,3588	-3,35871	1	154	1,23748	-1,23739	-1
58	-3,3588	3,35871	-1	155	-1,40314	0,93748	1
59	3,89755	-2,60418	1	156	1,40314	-0,93748	-1

60	-3,89755	2,60418	-1	157	-1,50133	0,62181	1
61	4,27297	-1,76985	1	158	1,50133	-0,62181	-1
62	-4,27297	1,76985	-1	159	-1,53249	0,30477	1
63	4,47485	-0,89004	1	160	1,53249	-0,30477	-1
64	-4,47485	0,89004	-1	161	-1,5	-0,00006	1
65	4,5	0,00007	1	162	1,5	0,00006	-1
66	-4,5	-0,00007	-1	163	-1,40987	-0,28049	1
67	4,35222	0,86578	1	164	1,40987	0,28049	-1
68	-4,35222	-0,86578	-1	165	-1,27031	-0,52624	1
69	4,04195	1,6743	1	166	1,27031	0,52624	-1
70	-4,04195	-1,6743	-1	167	-1,09128	-0,72923	1
71	3,58567	2,39595	1	168	1,09128	0,72923	-1
72	-3,58567	-2,39595	-1	169	-0,88385	-0,88392	1
73	3,00515	3,00525	1	170	0,88385	0,88392	-1
74	-3,00515	-3,00525	-1	171	-0,6597	-0,9874	1
75	2,32639	3,48182	1	172	0,6597	0,9874	-1
76	-2,32639	-3,48182	-1	173	-0,43048	-1,03938	1
77	1,5785	3,81103	1	174	0,43048	1,03938	-1
78	-1,5785	-3,81103	-1	175	-0,20724	-1,04209	1
79	0,79248	3,98445	1	176	0,20724	1,04209	-1
80	-0,79248	-3,98445	-1	177	0,00004	-1	1
81	-0,00007	4	1	178	-0,00004	1	-1
82	0,00007	-4	-1	179	0,18293	-0,91948	1
83	-0,76824	3,86183	1	180	-0,18293	0,91948	-1
84	0,76824	-3,86183	-1	181	0,33488	-0,80838	1
85	-1,48297	3,58	1	182	-0,33488	0,80838	-1
86	1,48297	-3,58	-1	183	0,45143	-0,67555	1
87	-2,11817	3,16994	1	184	-0,45143	0,67555	-1
88	2,11817	-3,16994	-1	185	0,53035	-0,53031	1
89	-2,6517	2,6516	1	186	-0,53035	0,53031	-1
90	2,6517	-2,6516	-1	187	0,57165	-0,38193	1
91	-3,06609	2,0486	1	188	-0,57165	0,38193	-1
92	3,06609	-2,0486	-1	189	0,57744	-0,23915	1
93	-3,34909	1,38716	1	190	-0,57744	0,23915	-1
94	3,34909	-1,38716	-1	191	0,5517	-0,10971	1
95	-3,49406	0,69493	1	192	-0,5517	0,10971	-1
96	3,49406	-0,69493	-1	193	0,5	0,00002	1
97	-3,5	-0,00008	1	194	-0,5	-0,00002	-1

Рассмотрим на примере решения задачи ирисов Фишера общий алгоритм подбора нейронной сети к ранее неизвестной задаче.

Не существует надёжных математических формул, с помощью которых можно однозначно подобрать оптимальную структуру ИНС к задаче (а те, что есть – для простых сетей и не всегда дают качественный результат). Поэтому желательно отталкиваться от уже готовых решений этой или похожей задач. Допустим, получить начальную структуру сети для задачи об ирисах можно с помощью команды nnstart, и выбора данной выборки из стандартных примеров Matlab (в разделе классификации). Но предположим, что нам не

удалось получить никаких идей о том, какая должна быть сеть. Тогда единственный путь методом перебора: обучить некое количество сетей и из них выбрать сеть с оптимальной структурой.

Общий план работы будет такой:

1) Разбиваем искомую выборку на три подмножества: обучающее, тестовое, валидационное (тот случай, когда это подмножество будет полезным).

2) Перебираем 25^2 сетей общей структуры 4-i-j-3, где i, j = 1..25. Два скрытых слоя – это конечно излишество для данной задачи, но они приведены для демонстрации более полного перебора. В переборе используем для контроля от переобучения только валидационную выборку. Запоминаем сеть с наилучшими i, j (таких сетей может быть несколько, поэтому запоминаем ещё и с наименьшими i и j). «Наилучшесть» сети определяется по количеству распознанных паттернов на валидационной выборке: чем больше распознали, тем лучше.

3) Далее объединяем обучающее и валидационное множество в новое обучающее, и учим нашу выбранную сеть ($4\text{-}i_{const}\text{-}j_{const}\text{-}3$). В качестве контроля используем тестовое множество, которое теперь формально будет новым валидационным.

Как видно, тестовое множество не используется в процессе подбора структуры сети, это сделано для того, чтобы это множество не стало артефактом обучения, а осталось объективной мерой оценки качества обучения. Если же структура сети известна сразу или количество возможных структур, из которых происходит выбор, будет небольшим, то можно валидационную выборку не использовать, а сразу бить все данные на обучающую и тестовую выборки.

Листинг программы с комментариями приведён ниже. Программа составлена без использования какой-либо модульности (процедур, функций, библиотек-модулей).

```
% загрузка выборки
load fisheriris;
% рисуем паттерны классов от sepal length и sepal width
gscatter(meas(:, 1), meas(:, 2), species, 'rgb', 'osd');
xlabel('Sepal length');
ylabel('Sepal width');

% бьём искомую выборку на три части: тестовое, валидационное и
% обучающее множество
colIndx = 1:4;
trainSeto = meas(1:35, colIndx);
trainVers = meas(51:85, colIndx);
trainVirg = meas(101:135, colIndx);
valSeto = meas(36:43, colIndx);
valVers = meas(86:93, colIndx);
valVirg = meas(136:143, colIndx);
```

```
testSeto = meas(44:50, colIndx);
testVers = meas(94:100, colIndx);
testVirg = meas(144:150, colIndx);
% формируем ответы учителя
a = [-1 -1 +1]';
b = [-1 +1 -1]';
c = [+1 -1 -1]';
% Заново объединяем все множества в одно, но паттерны уже
% располагаются в нужном нам порядке и оно транспонировано
initSet = [trainSeto' trainVers' trainVirg' valSeto' valVers' valVirg' testSeto' testVers'
testVirg'];
T = [repmat(a, 1, length(trainSeto)) repmat(b, 1, length(trainVers)) repmat(c, 1,
length(trainVirg)) repmat(a, 1, length(valSeto)) repmat(b, 1, length(valVers)) repmat(c, 1,
length(valVirg)) repmat(a, 1, length(testSeto)) repmat(b, 1, length(testVers)) repmat(c, 1,
length(testVirg))];
% оформляем ответы и транспонируем три подмножества
trainSet = [trainSeto' trainVers' trainVirg'];
trainT = [repmat(a, 1, length(trainSeto)) repmat(b, 1, length(trainVers)) repmat(c, 1,
length(trainVirg))];
valSet = [valSeto' valVers' valVirg'];
valT = [repmat(a, 1, length(valSeto)) repmat(b, 1, length(valVers)) repmat(c, 1,
length(valVirg))];
testSet = [testSeto' testVers' testVirg'];
testT = [repmat(a, 1, length(testSeto)) repmat(b, 1, length(testVers)) repmat(c, 1,
length(testVirg))];
% инициализируем некоторые начальные переменные перед обучением
size = 25;
trainCor = zeros(size, size);
valCor = zeros(size, size);
MAX = 0;
max_i = 0;
max_j = 0;
% перебираем возможные структуры сетей и проводим обучение
for i = 1:size,
    for j = 1:size,
        net = newff(initSet, T, [i j], {'tansig' 'tansig' 'tansig'}, 'trainbfg');
        net = init(net);
         % бить будем через диапазоны индексов
        net.divideFcn = 'divideind';
        % для обучающего множества
        net.divideParam.trainInd = 1:105;
        % для валидационного
        net.divideParam.valInd = 106:129;
        % максимальное количество ошибок на валидационном
        % множестве, имеется ввиду, что ошибка обучения падает,
        % а ошибка на валидационном множестве растёт
        net.trainParam.max_fail = 2;
         % обучаем сеть
        [net, tr] = train(net, initSet, T);
         % получаем ответы сети на обучающей части
        Y = sim(net, trainSet);
         % получаем ответы сети на валидационной части
```

```
        Z = sim(net, valSet);
          % высчитываем процент правильных ответов
        col = 0;
        for k = 1:length(trainSet)
          % процент правильных ответов определяется через евклидово
          % расстояние между ответом учителя и ответом сети,
          % величина расхождения – не более 0.01
            if norm(trainT(k)-Y(k)) < 0.01
              col = col + 1;
            end
        end
        trainCor(i, j) = 100 * col / length(trainT);

        col = 0;
        for k = 1:length(valT)
            if norm(valT(k)-Z(k)) < 0.01
              col = col + 1;
            end
        end
        valCor(i, j) = 100 * col / length(valT);
         % определяем максимальное количество ответов, причём
         % сохраняем сеть с минимальной архитектурой (в условии
         % используем “>”, а не “>=”
        if valCor(i, j) > MAX
          MAX = valCor(i, j);
          max_i = i;
          max_j = j;
        end
    end
end

figure
% рисуем получившиеся матрицы
surf(1:size, 1:size, trainCor);
% в 3-D
view(3)
% в 2-D
view(2)
figure
surf(1:size, 1:size, valCor);
view(2);
% учим выбранную сеть до тех пор, пока количество правильных
% ответов на всех ирисах не станет больше 90%
Q = 10;
while (Q < 90)
        net = newff(initSet, T, [max_i max_j], {'tansig' 'tansig' 'tansig'}, 'trainbfg');
        net = init(net);
        net.divideFcn = 'divideind';
          % объединяем обучающее и валидационное множество
        net.divideParam.trainInd = 1:129;
          % новое валидационное множество теперь то, что было
          % зарезервировано под тестовое
```

```
        net.divideParam.valInd = 130:150;
        net.trainParam.max_fail = 3;
        [net, tr] = train(net, initSet, T);
        Z1 = sim(net, initSet);
        col = 0;
        for k = 1:length(T)
           if norm(T(k)-Z1(k)) < 0.01
                col = col + 1;
           end
        end
        Q = 100 * col / length(T);
end
```

Сначала выведем распределение классов, результат показан на рисунке 5.8.

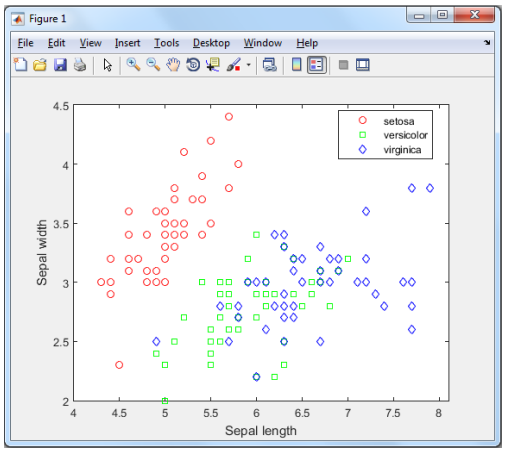


Рисунок 5.8 – Распределение паттернов классов в зависимости от двух параметров: sepal width и sepal length

Затем разобьём исходное множество на три выборки: валидационное, тестовое, обучающее. Каждый класс представлен 50 паттернами, 35 из них оставим для обучающей выборки, 8 – для валидационной и 7 – для тестовой.

Стоит отметить, что разбитие выборки на три множества – серьёзное дело. В учебной задаче можно обойтись простыми диапазонами не углубляясь в то, какие данные попадут в эти диапазоны, в реальных же задачах так бить нельзя. В одни диапазоны могут попасть крупные выбросы (нетипичные значения), в другие – усреднённые данные. В результате может получиться, что, допустим, валидационная и тестовая выборки будут составлены некорректно.

Когда мы загрузили выборку (**load fisheriris**), то массив species стал содержать названия ирисов, но ИНС работает только с числами, поэтому нам необходимо создать ответы учителя. Классов три, поэтому и выходов нейронов – три. Правильный нейрон кодируется +1, неправильный -1.

При создании сети с помощью **newff()** необходимо обозначить, что выходной нейрон

будет обладать нелинейной функцией активации, иначе по умолчанию, среда создаст сеть с нейронами, имеющими на выходе функцию **purelin**, рисунок 5.9.

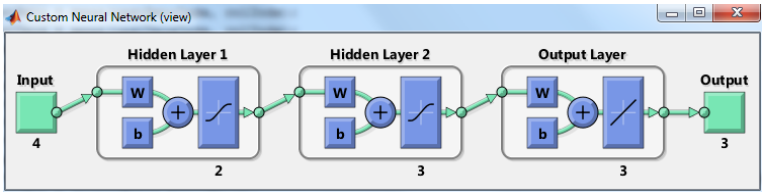


Рисунок 5.9 – Сеть, которая создастся по умолчанию, если в **newff()** прямо не указать, что нужны функции **tansig**

Процент правильно распознанных паттернов из обучающей выборки приведён на рисунке 5.10 (массив trainCor).

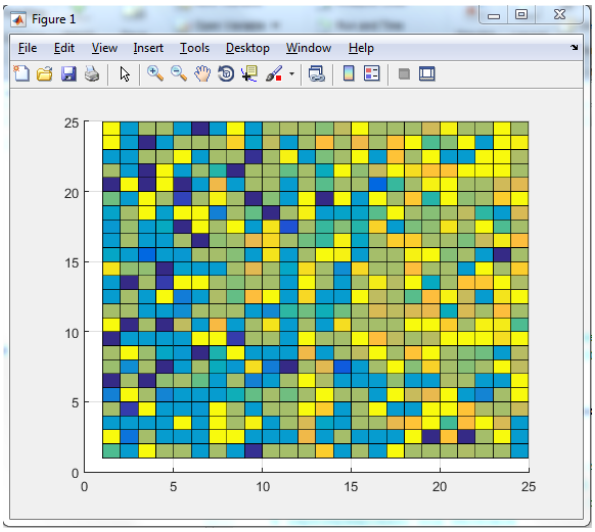


Рисунок 5.10 – Массив trainCor, желтые цвета соответствуют высокому проценту распознанных паттернов, синие - низкому

На рисунке 5.11 приведён массив valCor, именно по нему определяется первый максимально правильный ответ. В данном случае max = 100, max_i = 1, max_j = 3.

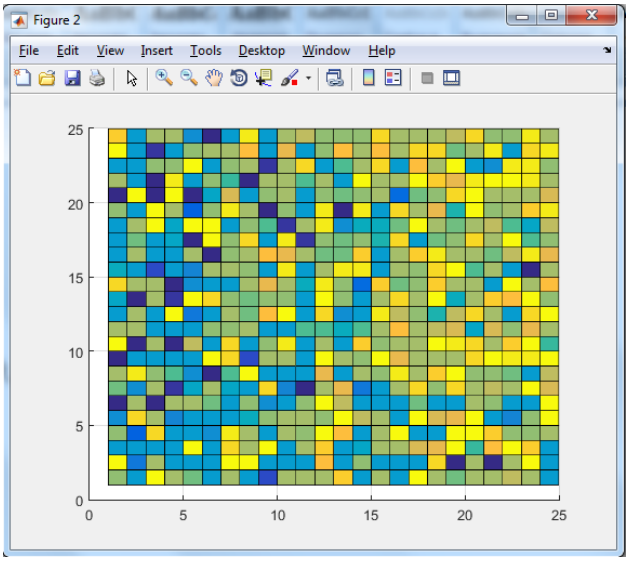


Рисунок 5.11 – Массив valCor

Можно наблюдать высокую корреляцию между двумя массивами, что говорит о хорошем разбиении искомого множества на валидационную и обучающую выборки. Также видно, что правый верхний угол матрицы значительно желтее, чем нижний левый, что говорит о том, что с ростом скрытых слоёв качество решения задачи улучшается.

Пример обучения сети со структурой 4-25-25-3 приведён на рисунке 5.12.

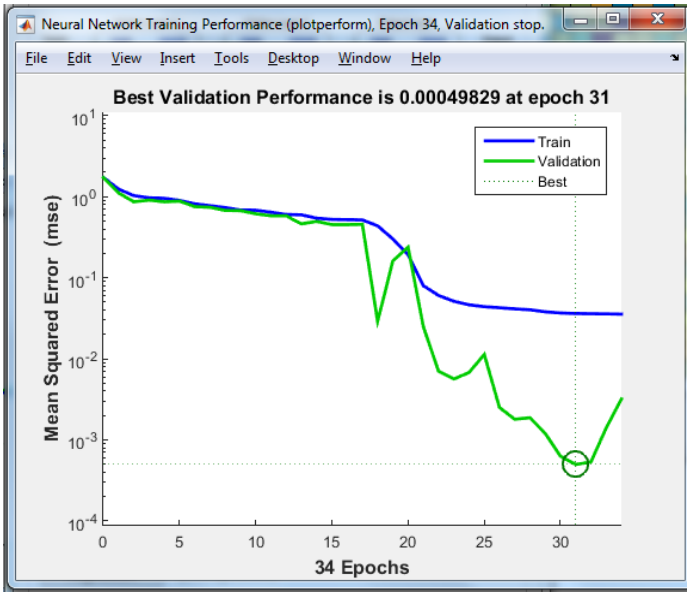


Рисунок 5.12 – Видно, что на 31 эпохе на валидационной выборки был достигнут лучший результат, потом произошёл рост ошибки, а так как значение max_fail установлено в 2, то очень быстро обучение было прервано, чтобы сеть не переобучилась и не превратилась в память

После выбора нужной структуры (4-1-3-3) необходимо обучить данную сеть. В коде использовался критерий оценки количества правильных результатов на всей

первоначальной выборки, но также можно использовать критерий оценки только на тестовой выборке. Стоит обратить внимание также на то, что первоначальная обучающая и валидационная выборки были объединены в новую обучающую выборку, а искомая тестовая выборка стала новой валидационной.

Далее рассмотрим решение задачи о спирали. Эта задача считается сложной, т.к. даже не разбивая искомую выборку на тестовое и обучающее множество тяжело заставить сеть построить спиральную поверхность отклика. Решим эту задачу с помощью сетей разной структуры и разных алгоритмов обучения.

Как видно из рисунка 5.7 поверхность отклика должна быть спиральной, причём один класс должен лежать на возвышенности этой поверхности, а другой – в низменности. Первый скрытый слой нейронов моделирует сигмойды, а второй объединяет их в более сложные поверхности. Становится очевидным, что для данной задачи достаточно сети со структурой 2-N-1, где собственно выходной нейрон и объединит сигмойды (сориентирует их так в пространстве), чтобы они объединились в спираль. Необходимо теперь выбрать значение N. Как уже отмечалось, точных формул нет, есть примерные формулы. Для сети с одним скрытым слоем формула имеет вид:

$$N = \sqrt{I * O}, \tag{5.1}$$

где N – количество нейронов в скрытом слое, I, O – количество нейронов во входном и выходном слоях соответственно.

Для сети с двумя скрытыми слоями будет так:

$$r = \sqrt[3]{\frac{I}{O}}, \tag{5.2}$$

$$\begin{cases} N_1 = O * r^2 \\ N_2 = O * r \end{cases}, \tag{5.3}$$

где N_1 и N_2 – количество нейронов в первом и втором слоях соответственно.

Как нетрудно убедиться, формула (5.1) для задачи о спирали не работает, т.к. $N = \lfloor \sqrt{2} \rfloor = 1$, что, разумеется, не даст решения задачи. Поэтому остаётся либо применять постепенный рост сети и смотреть на результат, либо искать в литературе примерныйразмер. Из [3] становится ясно, что $N \geq 40$. Учитывая, что мы отнимем часть выборки на тестовое множество, которое не будет участвовать в обучении, т.е. усложним и без того сложную задачу, возьмём N=60.

Ниже приведён код решения данной задачи.

% Загружаем данные из Excel

```
initSet = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'B:C');
T = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'D:D');
% Отображаем данные на графике
gscatter(initSet(:, 1), initSet(:, 2), T, 'br', 'xo');
xlabel('X');
ylabel('Y');
% Транспонируем множество и метки к нему
initSet = initSet';
T = T';
% Создаём сеть с одним скрытым слоем, на нём и на выходном
% слое устанавливаем ф.а. как гиперболический тангенс
% способ обучения trainlm – Левенберга-Марквардта
net = newff(initSet, T, [60], {'tansig' 'tansig'}, 'trainlm');
% Максимальное количество эпох
net.trainParam.epochs = 5000;
net = init(net);
% Способ разбиения на тестовое и обучающее множество – случайный
net.divideFcn = 'dividerand';
% Обучающее множество 85% = 165 паттернов
net.divideParam.trainRatio = 0.85;
% Тестовое множество 15% = 29 паттернов
net.divideParam.testRatio = 0.15;
net.divideParam.valRatio = 0;
% Минимальный уровень нормы для вектора градиента
net.trainParam.min_grad = 1e-9;
[net, tr, Y, E] = train(net, initSet, T);
% Определяем квадратную область вокруг спирали, тут
% по оси X и Y будет от -12 до 12, т.к. изначально вся спираль
% умещалась от -8 до 8 по X и от -6 до 6 по Y.
span = -12:.05:12;
[P1, P2] = meshgrid(span, span);
% вектор для ответов сети на этом квадрате
pp = [P1(:) P2(:)]';
% получаем ответы сети
aa = net(pp);
% возвращаемся к исходному рисунку со спиралью
figure(1)
% включаем режим добавления новой информации на этот рисунок,
% без него точки будут стёрты нашей раскраской
hold on;
% рисуем сетку
grid on;
% рисуем и закрашиваем сетку.
% -5 тут необходим для того, что без него только половина меток
% отобразится поверх раскраски, нам нужно понизить значения
% сетки до отрицательных величин, и тогда вся спираль будет
% видна поверх новой раскраски рисунка
mesh(P1, P2, reshape(aa, length(span), length(span))-5);
colormap copper
```

На примере этой задачи рассмотрим, как загружать данные из Excel. До этого момента данные либо генерировались Matlab, либо, как в случае с задачей об ирисах Фишера, извлекались из стандартных баз данных Matlab. Обычно же данные содержатся в табличной форме во внешнем файле. Для унификации будем всегда полагать, что этот внешний файл – файл Excel.

Сначала необходимо перенести данные из таблицы 5.2 в Excel, помня, что в Excel разделитель для вещественных чисел – это запятая, а не точка. В ячейках необходимо установить числовой формат. Пример части данных, уже перенесённых в Excel, приведён на рисунке 5.13. В данном случае данные располагаются по столбцам.

	A	B	C	D
1	№	X	Y	Class
2	1	6,5	0	1
3	2	-6,5	0	-1
4	3	6,3138	1,2559	1
5	4	-6,3138	-1,2559	-1
6	5	5,88973	2,43961	1
7	6	-5,88973	-2,43961	-1
8	7	5,24865	3,50704	1
9	8	-5,24865	-3,50704	-1
10	9	4,41941	4,41943	1
11	10	-4,41941	-4,41943	-1
12	11	3,43758	5,14473	1

Рисунок 5.13 – Пример данных в файле Excel

К этой лабораторной работе прилагается файл «DataForSpiral.xlsx», который содержит всю искомую выборку.

Сама же подгрузка столбцов будет осуществляться с помощью функции **xlsread()**, где первый параметр – это путь к файлу, а третий – диапазон по столбцам. Эта функция перегружена, поэтому, чтобы узнать другие способы её вызова можно обратиться к справке Matlab (F1). Входные данные поместим в initSet, а метки в T. Стоит сказать, что для данной задачи на выходе достаточно одного нейрона, принимающего значения от -1 до 1, хотя можно было бы и закодировать избыточно (разряженно) с помощью двух нейронов: [-1 +1]
– для первого класса, [+1 -1] – для второго класса.

Вывод точек осуществляем с помощью функции **gscatter()**, результат показан на рисунке 5.14.

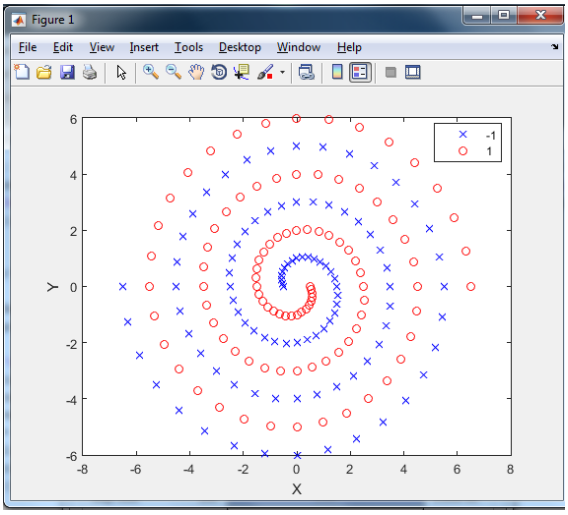


Рисунок 5.14 – Вывод точек спиралей для двух классов

Далее создаём сеть с помощью функции **newff(initSet, T, [60], {'tansig' 'tansig'}, 'trainlm')**. Мы задаём количество скрытых нейронов, функции активации и способ обучения. С помощью **net.trainParam.min_grad = 1e-9** мы устанавливаем минимальное значение нормы вектора градиента, достигнув которого, процесс обучения остановится. Дело в том, что алгоритм Левенберга-Марквардта (trainlm) быстро достигнет локального минимума, поэтому это значение можно и уменьшить, но тогда сеть может переобучиться, растеряв часть своих навыков.

Результат закрашивания квадратной области, в пределах которой находятся точки спирали, показан на рисунке 5.15.

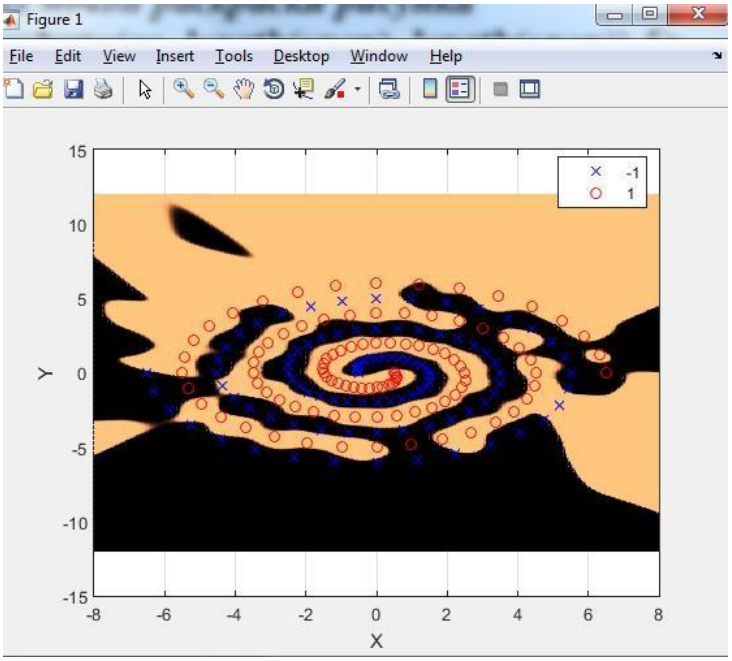


Рисунок 5.15 – Результат обучения сети

Видно, что не все представители классов двух спиралей попали на свой цвет, что и не удивительно, ведь в процессе обучения использовалась не вся спираль, а только её часть (напомним, что задачу о спиралях часто используют для тестирования алгоритмов обучения: насколько хорошо будет построена в процессе достижения локального или глобального минимума на поверхности ошибок искомая спиральная поверхность отклика. Бить на обучающее и тестовое множество не обязательно, задача и так сложна).

Поведение сети на тесте и на обучающем множестве показано на рисунке 5.16.

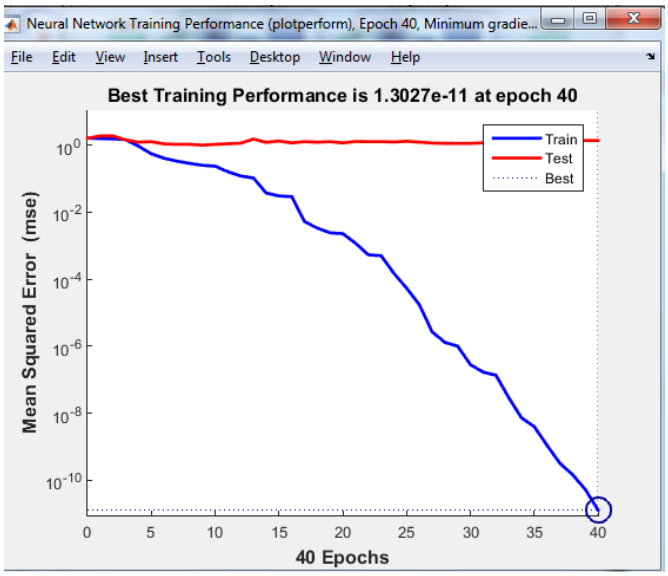


Рисунок 5.16 – Ошибка обучения и обобщения для 40 эпох

Из графиков видно, что снижать норму вектора градиента для продолжения обучения опасно, т.к. красный график (ошибка обобщения) потихоньку начинает ползти вверх.

В данной задаче тестовое множество специально не выделялось в коде, но индексы (29 штук), сгенерированные средой хранятся в **tr.testInd**, поэтому для получения конкретных ответов сети на тестовом множестве, можно написать **Q = sim(net, initSet(tr.testInd))**. Можно сравнить эти значения с ответами учителя: **T(tr.testInd)**. Из результатов сравнения становится видно, что даже несмотря на малую ошибку для обучающего множества сеть с большим трудом моделирует данную спиральную зависимость для тестового множества. Добиться для данной задачи необходимой поверхности отклика значительно проще, чем нужного ответа для тестового множества. С чем это связано?

В реальных задачах экземпляры классов обычно располагаются кучно друг по отношению к другу. Они образуют кластеры в пространстве признаков. Связано это с тем, что за каждым классом стоит некий инвариант, который в той или иной степени

проявляется в каждом экземпляре. Поэтому, если обучающая и тестовая выборка составлены репрезентативно, то низкая ошибка на обучающем множестве в целом должна приводить к низкой ошибке на тестовом множестве, т.к. гиперплоскости должны быть настроены и отделять одни кластеры от других в процессе обучения. В данной же задаче поверхность кластеров не выпуклая, более того, кластер одного класса глубоко заходит в кластер другого класса. В таких условиях угадать какому кластеру принадлежит экземпляр очень сложно. Этот пример демонстрирует ту особенность, что если зависимость сложная или в данных много хаоса (не шума), то хорошая ошибка обучения вообще ничего не скажет о том, как поведёт себя сеть на тестовом множестве. Вот почему очень сложно строить модели для предсказаний на финансовых рынках или модели для задачи классификации вроде рассмотренной.

Далее сравним различные алгоритмы обучения на примере задачи о спирали. Разбивать на тестовую и обучающую выборки уже не будем. Задача упрощается: смоделировать поверхность отклика для сети (рисунок 5.7), соответственно критерием качества будет моделируемая поверхность, а также количество эпох, которое потребовалось на это.

Подробную справку по алгоритмам обучения можно получить через команду **help nntrain**.

Рассмотрим следующие алгоритмы: **trainlm** (алгоритм Левенберга-Марквардта)**, trainbfg** (метод BFGS)**, traingd** (обычный алгоритм обратного распространения)**, traingdm** (алгоритм обратного распространения с моментом)**, trainrp** (алгоритм RPROP).

Для более подробного ознакомления рекомендуется [4–6].

Обучения ИНС – это поиск минимума на поверхности ошибок. Все методы обучения можно разбить на два класса: локальные и глобальные. Глобальные пытаются найти глобальный минимум, самый известный класс из таких методов – это генетические алгоритмы. Локальные методы ориентированы на пошаговый спуск из некоторой точки на поверхности ошибок к локальному минимуму. Локальные методы можно разбить на три класса в зависимости от того какая информация ими используется для спуска: методы нулевого порядка (не используют производные), методы первого порядка (используют первые производные, - обычно для получения вектора градиента), методы второго порядка (используют информацию от вторых производных (матрица Гесса, матрица Якоби)). Вцелом зависимость такая: чем выше порядок метода, тем он считается эффективнее для спуска, но в тоже время требует больше вычислений и памяти для хранения дополнительных структур (матриц). Если сети большие, то придётся отказаться от методов второго порядка, т.к. потребуется слишком много памяти и времени на обучение.

Глобальные методы часто используются для преднастройки сети перед использованием локальных методов.

Алгоритмы **trainlm** и **trainbfg** – алгоритмы второго порядка, все остальные – первого порядка. Между рассматриваемыми алгоритмами можно выделить следующуювзаимосвязь, таблица 5.3.

Таблица 5.3 – Взаимосвязь между рассматриваемыми алгоритмами

№	Тип алгоритма	Скорость, эффективность	Память
1	Trainlm	Самый быстрый, во многих функциях стоит по умолчанию	Расходует больше всех памяти
2	Trainbfg	Более медленный, чем Trainlm	Расходует меньше памяти, чем Trainlm
3	Trainrp	Более медленный, чем Trainbfg	Расходует меньше памяти, чем Trainbfg
4	Traingdm	Очень медленный	Расходует меньше памяти, чем Trainrp
5	Traingd	Очень медленный	Расходует меньше памяти, чем Trainrp, отличий от Traingdm практически нет

Листинг по применению алгоритма trainlm приведён ниже. Стоит обратить внимание, что здесь использована практически полная форма вызова функции **newff()**, а также установлен параметр **net.trainParam.epochs = 60.** Дело в том, что если не устанавливать этот параметр, то алгоритм остановит свою работу, когда модуль вектора градиента станет близким к нулю, но, если при этом изучить график ошибки обучения (синий цвет), то окажется, что при использовании этого алгоритма минимум достигается очень быстро (в среднем 50-60 эпох), а остальные силы (порядка 150 эпох) идут на
«топтание на месте» (приближение к асимптоте). В результате этого процесса сеть всё больше становится похожа на память и всё больше теряет обобщающие способности, поэтому имеет смысл остановить этот процесс. Получившаяся поверхность отклика представлена на рисунке 5.17.

Переменная **pref** содержит результирующую ошибку обучения (разницу между метками учителя и реальными ответами сети).

```
initSet = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'B:C');
T = xlsread('C:\Users\Computer Grand\Desktop\Для
```

```
студентов\Лабы\5\DataForSpiral.xlsx', 1, 'D:D');
gscatter(initSet(:, 1), initSet(:, 2), T, 'br', 'xo');
xlabel('X');
ylabel('Y');
initSet = initSet';
T = T';
% Вместо learngd можно применить learngdm, вместо mse
% msereg – с регуляризацией, crossentropy – кросс-энтропия
net = newff(initSet, T, [60], {'tansig' 'tansig'}, 'trainlm', 'learngd', 'mse', {}, {}, 'dividerand');
net.divideParam.trainRatio = 1;
net.divideParam.testRatio = 0;
net.divideParam.valRatio = 0;
net.trainParam.epochs = 60;
net = init(net);
[net, tr] = train(net, initSet, T);
% один из способов получения ответов сети
y = net(initSet);
% высчитываем разницу между ожидаемыми и реальными ответами
% PREF ≈ 0.0014
perf = perform(net, T, y);
span = -12:.05:12;
[P1, P2] = meshgrid(span, span);
pp = [P1(:) P2(:)]';
aa = net(pp);
figure(1)
hold  on;
grid on;
mesh(P1, P2, reshape(aa, length(span), length(span))-5);
colormap copper
```

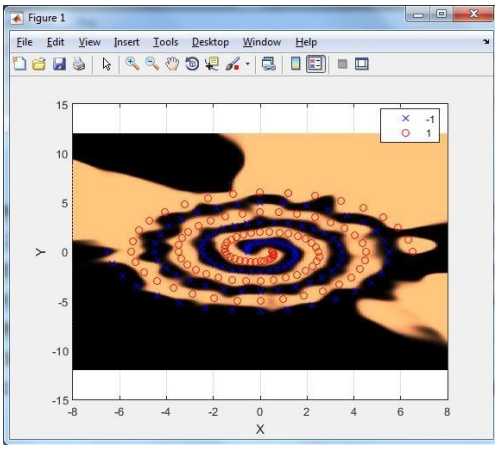


Рисунок 5.17– Поверхность отклика после применения алгоритма trainlm

Ниже приведён листинг для использования алгоритм **trainbfg**. Как видно, было изменено количество эпох, которые примерно потребуются для нахождения решения. Это количество выросло. Количество нейронов в скрытом слое также было снижено до 50.

Найденное решение показано на рисунке 5.18. Изменение ошибки обучения

показано на рисунке 5.19.

```
initSet = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'B:C');
T = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'D:D');
gscatter(initSet(:, 1), initSet(:, 2), T, 'br', 'xo');
xlabel('X');
ylabel('Y');
initSet = initSet';
T = T';
net = newff(initSet, T, [50], {'tansig' 'tansig'}, 'trainbfg', 'learngdm', 'mse', {}, {}, 'dividerand');
net.divideParam.trainRatio = 1;
net.divideParam.testRatio = 0;
net.divideParam.valRatio = 0;
net.trainParam.epochs = 250;
net = init(net);
[net, tr] = train(net, initSet, T);
y = net(initSet);
% PREF == 7.4596e-08;
perf = perform(net, T, y);
span = -12:.05:12;
[P1, P2] = meshgrid(span, span);
pp = [P1(:) P2(:)]';
aa = net(pp);
figure(1)
hold  on;
grid on;
mesh(P1, P2, reshape(aa, length(span), length(span))-5);
colormap copper
```

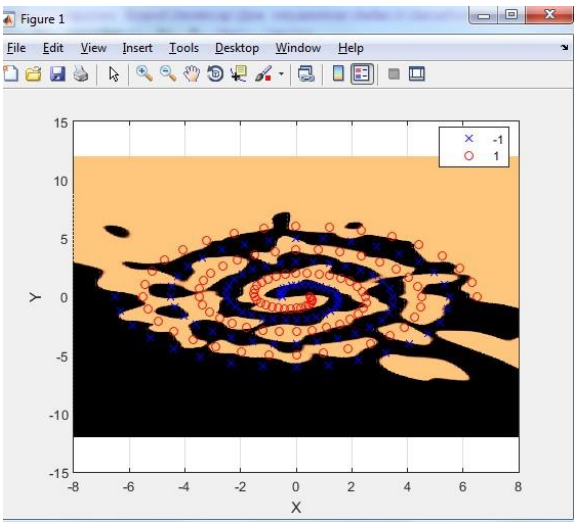


Рисунок 5.18 – Найденное решение с использованием алгоритм trainbfg

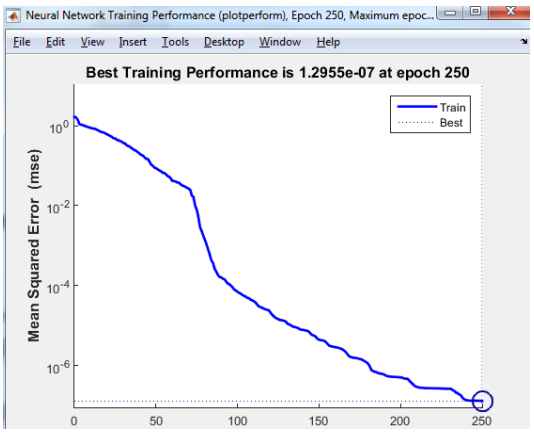


Рисунок 5.19 – Изменение величины ошибки обучения за 250 эпох

Далее рассмотрим применение алгоритма **trainrp**, но вместо функции **newff()** будем использовать **feedforwardnet()**. В пакете matlab очень часто существует иерархия функций, которые выполняют, по сути, одно и тоже, но с разной степенью детализации для пользователя. Для создания сетей прямого распространения можно выделить условно следующую иерархию: network()→newff()→feedforwardnet()→patternnet().

Ниже приведён листинг для обучения с помощью алгоритма trainrp (RPROP). Стоит обратить внимание, что количество эпох уже исчисляется тысячами. Сеть создавали с помощью функции **feedforwardnet()**, которая получает в качестве входных параметров количество нейронов в скрытом слое и алгоритм обучения. У данного алгоритма обычно настраивают такие параметры обучения как **net.trainParam.delt_inc, net.trainParam.delt_dec, net.trainParam.delta0, net.trainParam.deltamax, net.trainParam.lr**, однако, учитывая, что для осмысленного регулирования этих параметров необходимо знание тонкостей работы алгоритмы, мы их оставили настроенными по умолчанию. Для ознакомления с этими параметрами можно использовать команду **help trainrp**. Полученное решение показано на рисунке 5.20, а изменение значения ошибки обучения на рисунке 5.21.

```
initSet = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'B:C');
T = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'D:D');
gscatter(initSet(:, 1), initSet(:, 2), T, 'br', 'xo');
xlabel('X');
ylabel('Y');
initSet = initSet';
T = T';
net = feedforwardnet(60, 'trainrp');
net = configure(net, initSet, T);
net.layers{2}.transferFcn = 'tansig';
net.divideParam.trainRatio = 1;
```

```
net.divideParam.testRatio = 0;
net.divideParam.valRatio = 0;
net.trainParam.epochs = 6000;
net = init(net);
[net, tr] = train(net, initSet, T);
y = net(initSet);
perf = perform(net, T, y);
% PREF == 1.7276e-05.
span = -12:.05:12;
[P1, P2] = meshgrid(span, span);
pp = [P1(:) P2(:)]';
aa = net(pp);
figure(1)
hold  on;
grid on;
mesh(P1, P2, reshape(aa, length(span), length(span))-5);
colormap copper
```

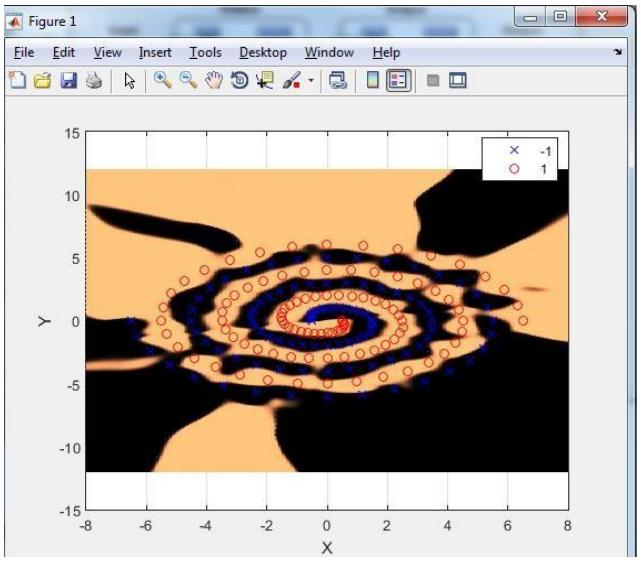


Рисунок 5.20 – Найденное решение с использованием алгоритма trainrp

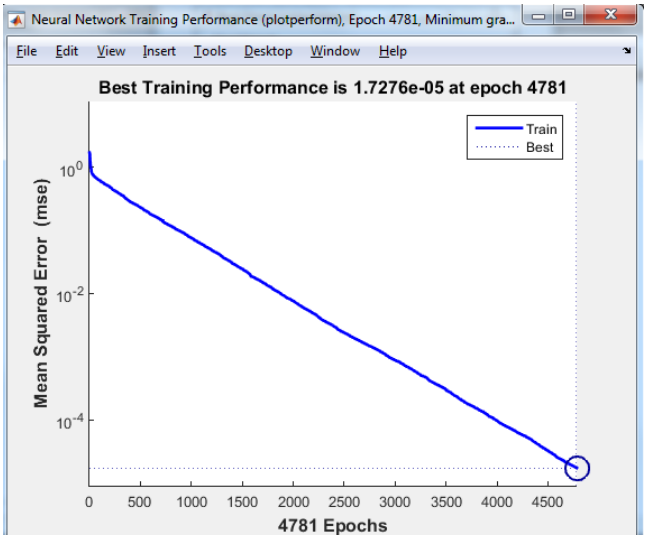


Рисунок 5.21 – Изменение величины ошибки обучения за 5000 эпох

Далее рассмотрим сеть с двумя скрытыми слоями для решения задачи о спиралях.

Формулы (5.2) и (5.3) тут также не подойдут для выбора размеров скрытых слоёв. Ориентируясь на различные источники выберем структуру 2-10-10-1.

Листинг обучения такой сети приведён ниже.

```
initSet = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'B:C');
T = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'D:D');
gscatter(initSet(:, 1), initSet(:, 2), T, 'br', 'xo');
xlabel('X');
ylabel('Y');
initSet = initSet';
T = T';
net = feedforwardnet([10 10], 'trainlm');
net = configure(net, initSet, T);
net.layers{3}.transferFcn = 'tansig';
net.divideParam.trainRatio = 0.85;
net.divideParam.testRatio = 0.15;
net.divideParam.valRatio = 0;
net.trainParam.epochs = 6000;
net.trainParam.min_grad = 1e-10;
net = init(net);
[net, tr] = train(net, initSet, T);
y = net(initSet);
perf = perform(net, T, y);
span = -12:.05:12;
[P1, P2] = meshgrid(span, span);
pp = [P1(:) P2(:)]';
aa = net(pp);
figure(1)
hold  on;
grid on;
mesh(P1, P2, reshape(aa, length(span), length(span))-5);
colormap copper
Q = sim(net, initSet(tr.testInd))
T(tr.testInd)
```

Подсчитав количество совпадений между векторами Q и T найдём, что качество работы сети выросло на неизвестных примерах с 28% до 50%.

Теперь создадим пользовательскую сеть, которая решает задачу о спирали. Структура для этой сети позаимствована из [1], рисунок 5.22.

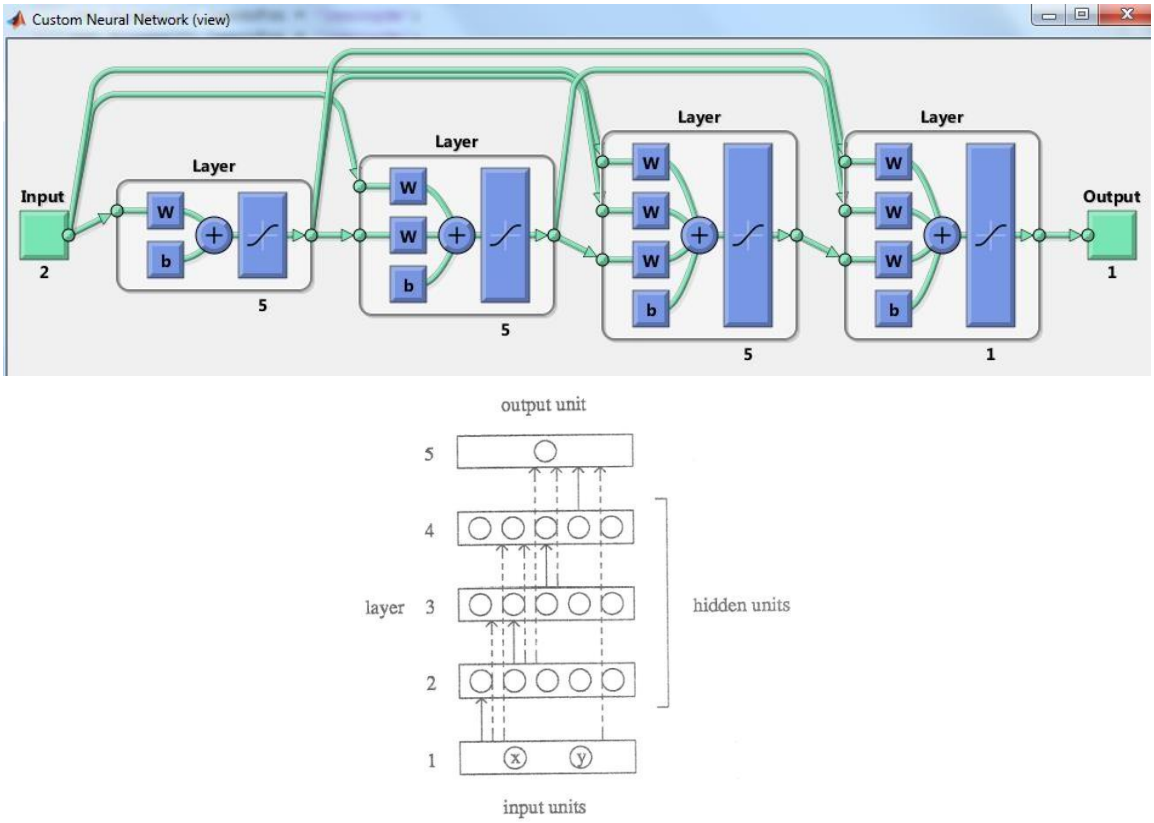


Рисунок 5.22 – Сверху структура предлагаемой сети в Matlab, снизу она же в виде обобщённой схемы

Как пишут авторы статьи, такая структура была выбрана потому, что они допустили, что каждый синапс-вес может с лёгкостью запомнить-хранить 1.5 бит информации (судя по всему предполагается, что вещественное число может в частности принять три значения: два концевых, допустим, 0 и 1, и одно промежуточное, допустим, 0.5. Таким образом, этоне два бита информации, но и не один). Всего в выборке 194 паттерна и для их хранения им потребуется 194 / 1.5 = 130 весов. Далее они подбирали сеть с несколькими слоями, чтобы она содержала примерно 130 весов. Сеть на рисунке 5.22 имеет 136 параметров. Можно это число получить сразу (после создания сети набрать **net.numWeightElemetns**) или подсчитать вручную: первый слой имеет два нейрона и связан с тремя другими слоями ((5+5)*3), второй слой связан с тремя слоями, два из них по 5 нейронов и один содержит один нейрон, сам же этот первый скрытый слой имеет тоже 5 нейронов (25+25+5), третий слой связан с двумя слоями (25+5), и для четвёртого слоя – 5. Также учитываем количество смещений для всех скрытых слоёв и выходного слоя – 16. Получаем 120+16=136. Что касается связей через слой, то они нужны, чтобы частично скомпенсировать затухание градиента. Если их убрать и оставить только связи от последовательных слоёв, то невязка, распространяемая с выходного слоя, дойдя до входного, совсем затухнет (приметмаленькие значения), и обучение будет неэффективным. Поэтому связи через один-два слоя

– вынужденная мера. Стоит отметить, что вообще с затуханием градиента борются через введение общих весов (сверточные сети).

Листинг программы с подробными комментариями приведён ниже. Стоит обратить внимание на применение функции **network()**. В лабораторной работе 4 объяснялись её параметры.

```
initSet = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'B:C');
T = xlsread('C:\Users\Computer Grand\Desktop\Для студентов\Лабы\5\DataForSpiral.xlsx', 1, 'D:D');
gscatter(initSet(:, 1), initSet(:, 2), T, 'br', 'xo');
xlabel('X');
ylabel('Y');
initSet = initSet';
T = T';
% создание пользовательской сети
net = network(1, 4, [1; 1; 1; 1], [1; 1; 1; 0], [0 0 0 0; 1 0 0 0; 1 1 0 0; 1 1 1 0], [0 0 0 1]);
% размеры скрытых слоёв
net.layers{1}.size = 5;
net.layers{2}.size = 5;
net.layers{3}.size = 5;
% функции активации на скрытых слоях
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'tansig';
net.layers{3}.transferFcn = 'tansig';
% функция активации на выходном слое
net.layers{4}.transferFcn = 'tansig';
% способ разделения обучающего множества: случайно
net.divideFcn = 'dividerand';
net.divideParam.trainRatio = 1;
net.divideParam.testRatio = 0;
net.divideParam.valRatio = 0;
% алгоритм обучения RPROP, нужно учесть, что чем сложнее
% структура пользовательской сети, тем более осторожно
% нужно применять мощные алгоритмы, т.к. они могут не сработать
% в силу отсутствия реализации некоторых тонкостей настройки
% этих алгоритмов к пользовательским архитектурам.
% trainrp, traingd, traingdm – как раз простые алгоритмы, которые
% подойдут практически ко всему
net.trainFcn = 'trainrp';
% функция ошибки – mse, ставить, допустим, crossentropy удастся
% не всегда, т.к. многие алгоритмы рассчитаны на mse или msereg
net.performFcn = 'mse';
% максимальное количество эпох, ставим побольше
net.trainParam.epochs = 35000;
net.trainParam.goal = 0;
% начальная скорость обучения
net.trainParam.lr = 0.01;
net.trainParam.max_fail = 4;
% настройки RPROP
net.trainParam.delt_inc = 1.2;
```

```
net.trainParam.delt_dec = 0.5;
net.trainParam.delta0 = 0.07;
net.trainParam.deltamax = 50.0;
% минимальный размер модуля градиента для остановки алгоритма
% для более мощных алгоритмов можно установить 1e-10, для слабых
% от 1e-5 до 1e-7
net.trainParam.min_grad = 1e-7;
% подключаем график изменения ошибки обучения
net.plotFcns = {'plotperform'};
% чтобы на пользовательской архитектуре работал алгоритм
% обратного распространения ошибки, нужно вручную для слоёв
% установить инициализацию весов и правило обновления-обучения
% установка правила обучения для весов
net.adaptFcn = 'adaptwb';
% net.layerWeights{i, j} – это матрица 4x4, показывающая какой слой
% с каким связан, в ней заполнены все элементы ниже главной
% диагонали, кроме элемента {4, 4}, поэтому для них для всех нужно
% установить правило обучения
net.layerWeights{1}.learnFcn = 'learngdm';
net.layerWeights{2}.learnFcn = 'learngdm';
net.layerWeights{3}.learnFcn = 'learngdm';
net.layerWeights{3,2}.learnFcn = 'learngdm';
net.layerWeights{4}.learnFcn = 'learngdm';
net.layerWeights{4,2}.learnFcn = 'learngdm';
net.layerWeights{4,3}.learnFcn = 'learngdm';
% тоже, но для смещений
net.biases{1}.learnFcn = 'learngdm';
net.biases{2}.learnFcn = 'learngdm';
net.biases{3}.learnFcn = 'learngdm';
net.biases{4}.learnFcn = 'learngdm';
% для входного слоя
net.inputWeights{1}.learnFcn = 'learngdm';
% этот параметр сообщает, что при применении configure(), веса
% будут инициализированы с помощью выбранного нами параметра,
% мы выбрали initnw, т.е. инициализация по методу Нгуена-Видроу
net.initFcn = 'initlay';
net.layers{1}.initFcn = 'initnw';
net.layers{2}.initFcn = 'initnw';
net.layers{3}.initFcn = 'initnw';
net.layers{4}.initFcn = 'initnw';
net.biases{1}.initFcn = 'initnw';
net.biases{2}.initFcn = 'initnw';
net.biases{3}.initFcn = 'initnw';
net.biases{4}.initFcn = 'initnw';
% устанавливаем величину момента
net.trainParam.mc = 0.5;
% конфигурируем сеть, т.е. применяем к ней все наши настройки
net = configure(net, initSet, T);
% обучаем сеть
[net, tr, Y, E] = train(net, initSet, T);
y = net(initSet);
% вычисляем производительность
```

```
perf = perform(net, T, y);
span = -12:.05:12;
[P1, P2] = meshgrid(span, span);
pp = [P1(:) P2(:)]';
aa = net(pp);
figure(1)
hold  on;
grid on;
% выводим полученное решение
mesh(P1, P2, reshape(aa, length(span), length(span))-5);
colormap copper
```

На рисунках 5.23–5.25 приведены результаты обучения этой сети.

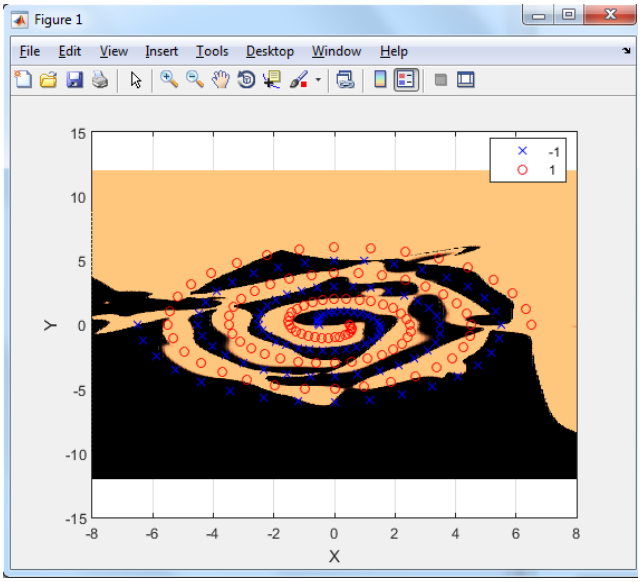


Рисунок 5.23 – Полученная поверхность отклика для пользовательской сети

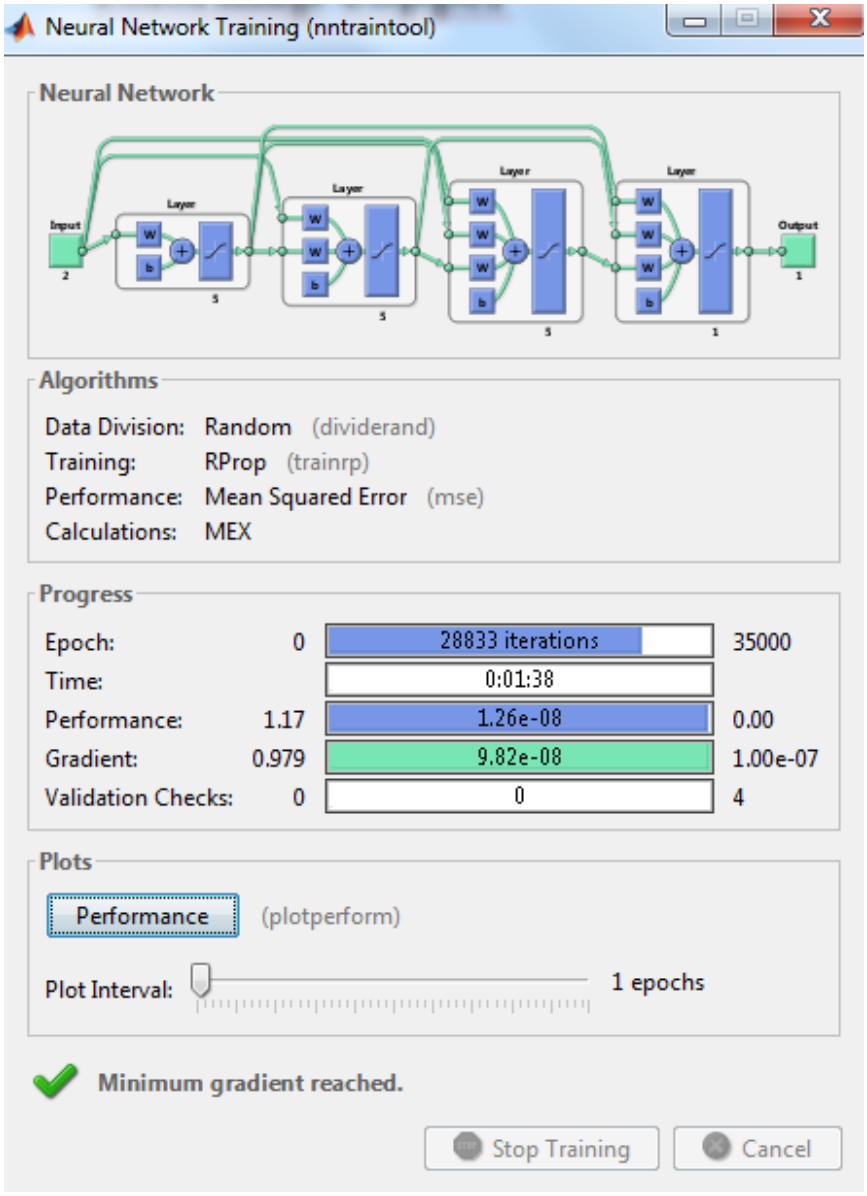


Рисунок 5.24 – Результирующие параметры обучения

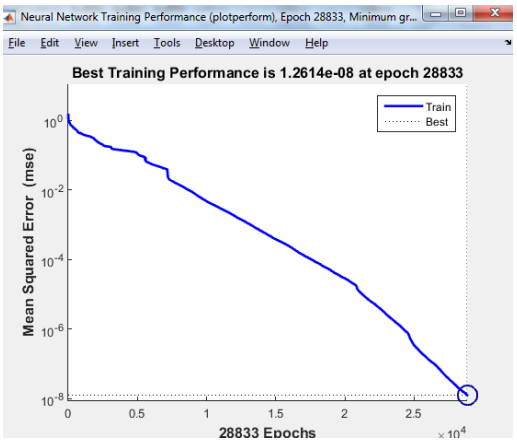


Рисунок 5.25 – Изменение величины ошибки обучения

Изменив **net.divideParam.trainRatio = 0.85, net.divideParam.testRatio = 0.15** можно

ещё раз обучить сеть, подсчитать число совпадений между векторами Q и T (как в первом листинге). Получилось порядка 65%.

Теперь можно сделать общие выводы по всем трём сетям. Будем оценивать ресурсы, обобщающую способность, аппроксимационную способность. Под ресурсами понимается количество синаптических весов, т.е. количество настраиваемых параметров. Так как мы проводили эксперименты с разным количеством нейронов для первой сети с одним скрытым слоем, то для определённости пусть будет 50 нейронов в скрытом слое. Под аппроксимационной способностью подразумевается способность построить необходимую поверхность отклика. Результаты приведены в таблице 5.4.

Таблица 5.4 – Сравнение трёх сетей с различной структурой

Структура сети	Ресурсы net.numWeightElements	Справилась с задачей	Обобщающая способность
2-50-1	201	да	30%
2-10-10-1	151	да	50%
2-5-5-5-1	136	да	65%

Обобщающую способность можно было бы оценить более точно, если провести не один эксперимент, а целую серию и усреднить результаты, подсчитав, допустим, доверительные интервалы. Что видно из этих результатов?

Во-первых, введение промежуточных слоёв позволяет уменьшить количество настраиваемых параметров.

Во-вторых, все три сети без проблем справились с постройки правильной поверхности отклика, т.е. без проблем раскрасили спирали.

В-третьих, то, что они хорошо строят поверхность отклика, не говорит ничего про их обобщающую способность! Первые две сети не справились с заданием вообще, т.к. 50% - это метод «тыка» при двух классах. Это говорит о том, что для сложных задач сложная система нелинейности (много слоёв) подходит лучше, чем простая, т.е. такие сети могут лучше справляться с задачами классификации. Более подробно про доказательства этого результата можно почитать в [7], где рассматриваются неглубокие архитектуры (shallow architecture) и глубокие архитектуры (deep architecture). Стоит также отметить, что для того,чтобы ИНС качественно предсказывали какой спирали может принадлежать точка, нужно увеличить обучающую выборку. Для двух спиралей обучающая выборка должна быть в среднем в районе 400 точек.

В приложении 1 приведён код алгоритма обратного распространения ошибки для

построения разделяющей поверхности к данным, представленным на рисунке 5.26. Это две полуспирали.

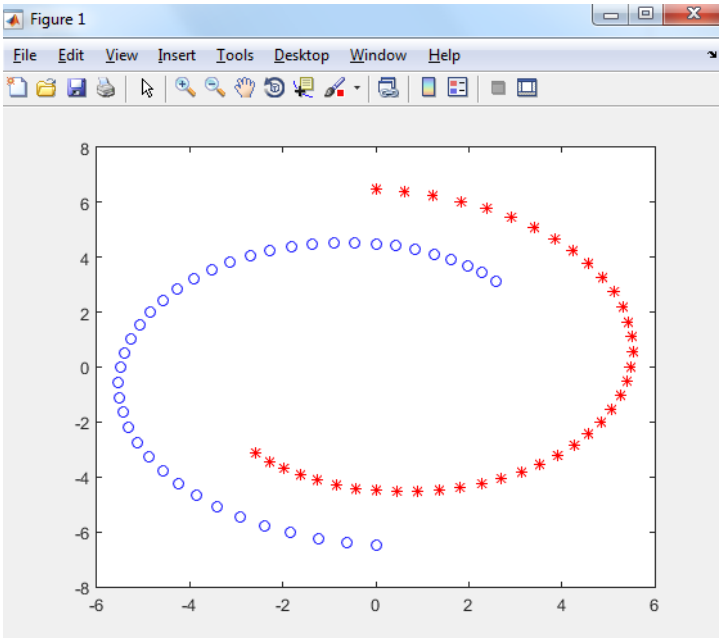


Рисунок 5.26 – Две полуспирали, которые необходимо разделить

В данном лабораторном практикуме везде используются высокоуровневые функции Matlab для обучения и создания нейронных сетей. В приложении 1 показано, как самому запрограммировать обучение сети.

Аппаратура и материалы. 64-разрядный (x64) персональный компьютер, процессор с тактовой частотой 1 ГГц и выше, оперативная память 1 Гб и выше, свободное дисковое пространство не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система Windows 7 и выше, Matlab (R2013) и выше.

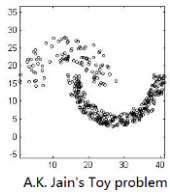
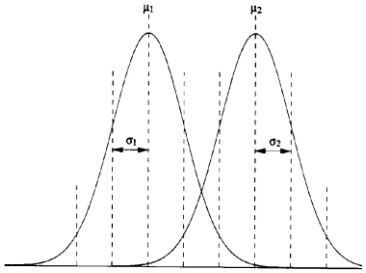
Указание по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

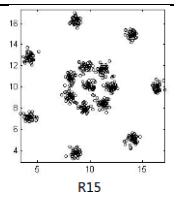
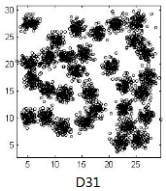
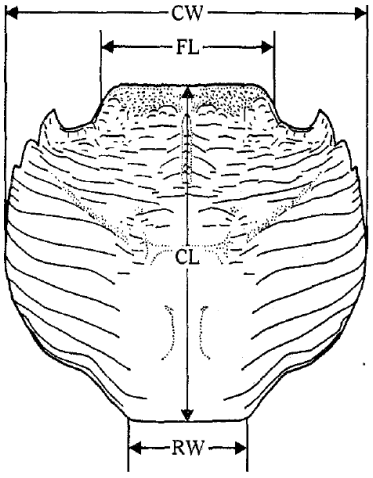
Методика и порядок выполнения работы

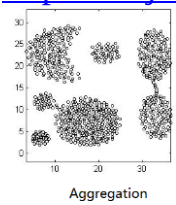
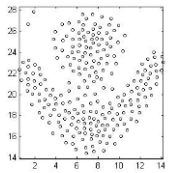
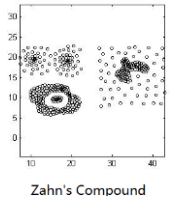
В индивидуальном варианте (таблица 5.5) дано условие задачи на классификацию. Все задачи подобраны так, что предобработки данных в целом не требуется (хотя в

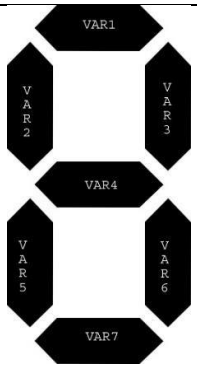
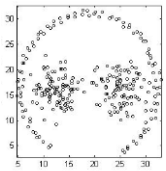
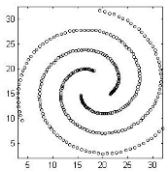
некоторых задачах её можно было бы сделать), приемлемая точность на тестовой выборке везде отмечена как 70%, что в целом не много. Достижение более большой точностираспознавания оценивается в дополнительных баллах.

Таблица 5.5 – Варианты заданий

№	Условие задачи
1	Синтетические данные на классификацию. Постоянный адрес задачи: https://cs.joensuu.fi/sipu/datasets/.  Необходимо перевести выборку в формат Excel. Затем решить задачу классификации: создать нейронную сеть, которая принимая координаты точки, относит её к тому или иному классу. Разбитие на тестовую и обучающую выборку осуществляет сам студент, исходят из задачи. Можно ориентироваться на 80%-85% для обучающей выборки и 15%-20% для тестовой. В выборках обязательно учесть степень представленности классов. Если один класс представлен 20 паттернами, а другой 5, то очевидно, что в обучающей и тестовой выборке окажутся разное количество паттернов из этих классов. Постараться получить классификатор с качеством правильных ответов не ниже 70% для тестовой выборки. Правильным ответом считается ответ сети, при котором евклидово расстояние между идеальным требуемым ответом (учитель) и ответом сети не превышает по модулю 0.1.
2	Задача о двух многомерных перекрывающихся нормальных распределениях.  Даны два двадцатимерных нормальных распределения (7400 паттернов). Одно кодируется классом 0, другое – 1. Класс 1 имеет среднее (a, a, a,…, a), класс 2 – (-a, -a, -a, …, -a), где $a = 2 / \sqrt{20}$. Необходимо на тестовом множестве (можноиспользовать 300 паттернов) добиться правильной классификации в 70%. Ошибка примерно 25% - нормальна.
3	Дана синтетическая выборка, в которой паттерны принадлежат двум кластерам в форме банана. Каждый паттерн задаётся двумя точками и типом класса: -1 или 1. Требуется разбить выборку на два множества: обучающее и тестовое и добиться правильной классификации на тестовом множестве более 70%. В этой задаче полезно для студента вывести данные на канву и оценить степень репрезентативной представленности каждого класса.
4	Синтетические данные на классификацию. Постоянный адрес задачи: https://cs.joensuu.fi/sipu/datasets/.

	 R15 Комментарий к варианту 1.
5	Задача про фонемы. Даны два класса: один описывает назальный звук (класс 0), другой – оральный (класс 1). Класс 0 содержит 3818 паттернов, а класс 1 – 1586 паттернов. Каждый класс описывается 5 фонемами: Aa, Ao, Dcl, Iy, Sh. Фонемы изменяются в следующих вещественных диапазонах: Aa [-1.7, 4.107], Ao [-1.327, 4.378], Dcl [-1.823, 3.199], Iy [-1.581, 2.826], Sh [-1.284, 2.719]. В физический смысл этих фонем и диапазонов вникать не нужно. Необходимо произвести классификацию и добиться на тестовом множестве качества распознавания не ниже 70%.
6	Синтетические данные на классификацию. Постоянный адрес задачи: https://cs.joensuu.fi/sipu/datasets/.  D31 Комментарий к варианту 1.
7	Задача о крабах genus Leptograpsus. Имеется выборка по крабам двух видов. Каждый вид представлен 50 самцами и 50 самками, т.е. вся выборка 200 измерений. Первая колонка – пол. Вторая колонка – индекс от 1 до 50, далее 5 колонок – это измерения тела краба (как в задачах об ирисах Фишера) в мм: FL, RW, CL, CW, BD. Схема измерений представлена ниже.В [8] можно получить подробную информацию об этих животных, а также о смысле эксперимента.  Последняя колонка это вид краба. Выборка бьётся на тестовую и обучающую. Для тестовой выборки подойдёт по 5 экземпляров из каждого класса: 5 самцов типа 1, 5 самок типа 1, 5 самцов типа 2, 5 самок типа 2. Всё остальное – обучающая выборка, хотя студент может предложить своё разбиение. На вход сети подаётся 6 параметров (индексы, разумеется, подавать не надо, они нужны для разбиения на обучающее и тестовое множество, а также для анализа ответов). Требуется создатьи обучить сеть, которая правильно определяет тип краба на тестовой выборке с вероятностью более 70% (чем больше, тем лучше). Что такое правильный ответ написано в варианте №1, а также разобрано в задаче об ирисах Фишера.

8	Синтетические данные на классификацию. Постоянный адрес задачи: https://cs.joensuu.fi/sipu/datasets/.  Комментарий к варианту 1.
9	Синтетические данные на классификацию. Постоянный адрес задачи: https://cs.joensuu.fi/sipu/datasets/.  Комментарий к варианту 1.
10	Задача о классификации стекла. Имеется выборка из 214 записей. Состоит из 10 колонок (+ одна колонка индексов): RI – индекс преломления стекла, далее 8 колонок, которые обозначают содержание в процентах того или иного элемента (Na, Mg, Al, Si, K, Ca, Ba, Fe). Последняя колонка – тип стекла: 1 – window float glass, 2 – window non-float glass, 3 – vehicle glass, 4 – vehicle non-float glass (**данный класс отсутствует в данной выборке**), 5 – containers, 6 – tableware, 7 – vehicle headlamp glass. Из-за отсутствующего одного класса получается 6 типов стекла. Выборку необходимо разбить на две части: обучающую и тестовую. В данной выборке классы представлены различным количеством элементов, поэтому для тестовой каждый класс вносит не одинаковое количество паттернов: 1 – 10 паттернов, 2 – 10, 3 – 3, 5 – 3, 6 – 2, 7 – 4. Всё остальное идёт в обучающую выборку. Добиться классификации с точностью на тестовой выборке более 70%. Дополнительную информацию вместе со статьями, в которых использовалась эта выборка, можно получить по адресу: https://archive.ics.uci.edu/ml/datasets/Glass+Identification/. В Matlab можно набрать команду **nprtool**, нажать **Next**, далее **Load Example Dataset** и выбрать **Types of Glass** и почитать информацию об этой задаче. Однако, стоит учесть, что в Matlab интегрирована упрощенная задача, где на выходе имеем всего 2 класса.
11	Синтетические данные на классификацию. Постоянный адрес задачи: https://cs.joensuu.fi/sipu/datasets/.  Комментарий к варианту 1.
12	Задача о цифрах.

	 Цифра на калькуляторе кодируется переменными VAR1..VAR7, расположенными так, как показано выше. Переменная может принимать значение 0 (ZERO), если часть табло, которая связана с этой переменной не работает, или 1 (ONE), если часть табло работает. В выборке всего 500 записей, каждая запись состоит из 7 значений переменных и класса цифры. Не все переменные правильно кодируют свой класс, т.к. табло неисправно. Необходимо составить классификатор, который на тестовом множестве покажет выше 70% правильных ответов. Можно использовать следующую структуру сети: 7-5-10, количество эпох 500, начальная скорость обучения 0.01. Разбитие на обучающую и тестовую выборки можно производить исходя из соотношений 85%-15%, степень равномерной представленности каждого класса должен сделать сам студент.
13	Синтетические данные на классификацию. Постоянный адрес задачи: https://cs.joensuu.fi/sipu/datasets/.  Path-based1 Комментарий к варианту 1.
14	Синтетические данные на классификацию. Постоянный адрес задачи: https://cs.joensuu.fi/sipu/datasets/.  path-based2: spiral Комментарий к варианту 1.

Содержание отчета и его форма

Отчёт по лабораторной работе должен содержать следующую информацию:

1. Название лабораторной работы и её номер.
2. ФИО студента и группу.
3. Формулировка индивидуального задания.
4. Документ отчёта с Print Prtscr диалоговых окон по шагам для своего варианта по подобию того, что описано в теоретической части. Должна присутствовать информация о количестве классов, количестве паттернов в каждом классе, о размерах тестовой и

обучающей выборки, о представленности каждого класса в тестовой и обучающей выборке, об используемых алгоритмах обучения нейронных сетей, а также результирующие графики и процент правильно распознанных паттернов на тестовой выборке. Необходимо также указать структуру сети и обосновать почему выбрана именная такая сеть.

5. Ответы на контрольные вопросы.

Вопросы для защиты работы

1) Какие алгоритмы обучения нейронных сетей вы знаете, чем они отличаются?
2) Для чего используются в сетях прямого распространения связи через слой?
3) В каких случаях для задачи классификации выгоднее использовать рбф-нейрон, чем сигмоидальный нейрон?
4) Какой ответ сети считать правильным?
5) Чем нужно руководствоваться, разбивая выборку на тестовую и обучающую?
6) Какие функции в Matlab можно использовать для создания сети и в чём их отличие?
7) Какая структура данных содержит информацию об процессе обучения сети и как её получить?

Лабораторная работа 5.
Сегментация изображений

Цель: изучить методы сегментации изображений
Формируемые компетенции: ПК-11

Теоретическая часть

Сегментация методом управляемого водораздела

Довольно часто при анализе изображений возникает задача разделения пикселей изображений на группы по некоторым признакам. Такой процесс разбиения на группы называется сегментацией. Наиболее известными являются два вида сегментации - сегментация по яркости для бинарных изображений и сегментация по цветовым координатам для цветных изображений. Методы сегментации можно рассматривать как формализацию понятия выделяемости объекта из фона или понятий связанных с градиентом яркости. Алгоритмы сегментации характеризуются некоторыми параметрами надежности и достоверности обработки. Они зависят от того, насколько полно учитываются дополнительные характеристики распределения яркостив областях объектов или фона, количество перепадов яркости, форма объектов и др.

Существует много изображений, которые содержат исследуемый объект достаточно однородной яркости на фоне другой яркости. В качестве примера можно привести рукописный текст, ряд медицинских изображений ит.д. Если яркости точек объекта резко отличаются от яркостей точек фона, то решение задачи установления порога является несложной задачей. На практике это не так просто, поскольку исследуемое изображение подвергается воздействию шума и на нем допускается некоторый разброс значений яркости. Известно несколько аналитических подходов к пороговому ограничению по яркости. Один из методов состоит в установлении порога на таком уровне, при котором общая сумма элементов с подпороговой яркостью согласована с априорными вероятностями этих значений яркости.

Аналогичные подходы можно применить для обработки цветных и спектрозональных изображений. Существует также такой вид сегментации

как контурная сегментация. Довольно часто анализ изображений включает такие операции, как получение внешнего контура изображений объектов и запись координат точек этого контура. Известно три основных подхода к представлению границ объекта: аппроксимация кривых, прослеживание контуров и связывание точек перепадов. Для полноты анализа следует отметит, что есть также текстурная сегментация и сегментация формы.

Наиболее простым видом сегментации является пороговая сегментация. Она нашла очень широкое применение в робототехнике. Это объясняется тем, что в этой сфере изображения исследуемых объектов, в своем большинстве, имеют достаточно однородную структуру и резко выделяются их фона. Но кроме этого, для достоверной обработки нужно знать, что изображение состоит из одного объекта и фона, яркости которых находятся в строго известных диапазонах и не пересекаются между собой.

Развитие технологий обработки изображений привело к возникновению новых подходов к решению задач сегментации изображений и применении их при решении многих практических задач.

В данной работе рассмотрим относительно новый подход к решению задачи сегментации изображений - метод водораздела. Коротко объяснимназвание этого метода и в чем его суть.

Предлагается рассматривать изображение как некоторую карту местности, где значения яркостей представляют собой значения высот относительно некоторого уровня. Если эту местность заполнять водой, тогда образуются бассейны. При дальнейшем заполнении водой, эти бассейны объединяются. Места объединения этих бассейнов отмечаются как линииводораздела.

Разделение соприкасающихся предметов на изображении является одной из важных задач обработки изображений. Часто для решения этой задачи используется так называемый метод маркерного водораздела. Припреобразованиях с помощью этого метода нужно определить "водосборные

бассейны" и "линии водораздела" на изображении путем обработки локальных областей в зависимости от их яркостных характеристик.

Метод маркерного водораздела является одним из наиболее эффективных методов сегментации изображений. При реализации этого метода выполняются следующие основные процедуры:

1. Вычисляется функция сегментации. Она касается изображений, где объекты размещены в темных областях и являются трудно различимыми.
2. Вычисление маркеров переднего плана изображений. Они вычисляются на основании анализа связности пикселей каждого объекта.
3. Вычисление фоновых маркеров. Они представляют собой пиксели, которые не являются частями объектов.
4. Модификация функции сегментации на основании значений расположения маркеров фона и маркеров переднего плана.
5. Вычисления на основании модифицированной функции сегментации.

В данном примере среди функций пакета Image Processing Toolbox наиболее часто используются функции **fspecial, imfilter, watershed, label2rgb, imopen, imclose, imreconstruct, imcomplement, imregionalmax, bwareaopen, graythresh и imimposemin**.

Сначала считаем цветное изображение с множеством объектов длякотороых необходимо провести сегментацию, рисунок 7.1.

```
L=imread('H:\РАБОТА\2018\Digital Image Processing\Лабораторный практикум\К лабе 7\mandarins.jpg');
figure, imshow(L);
I=rgb2gray(L);
imshow(I)
```

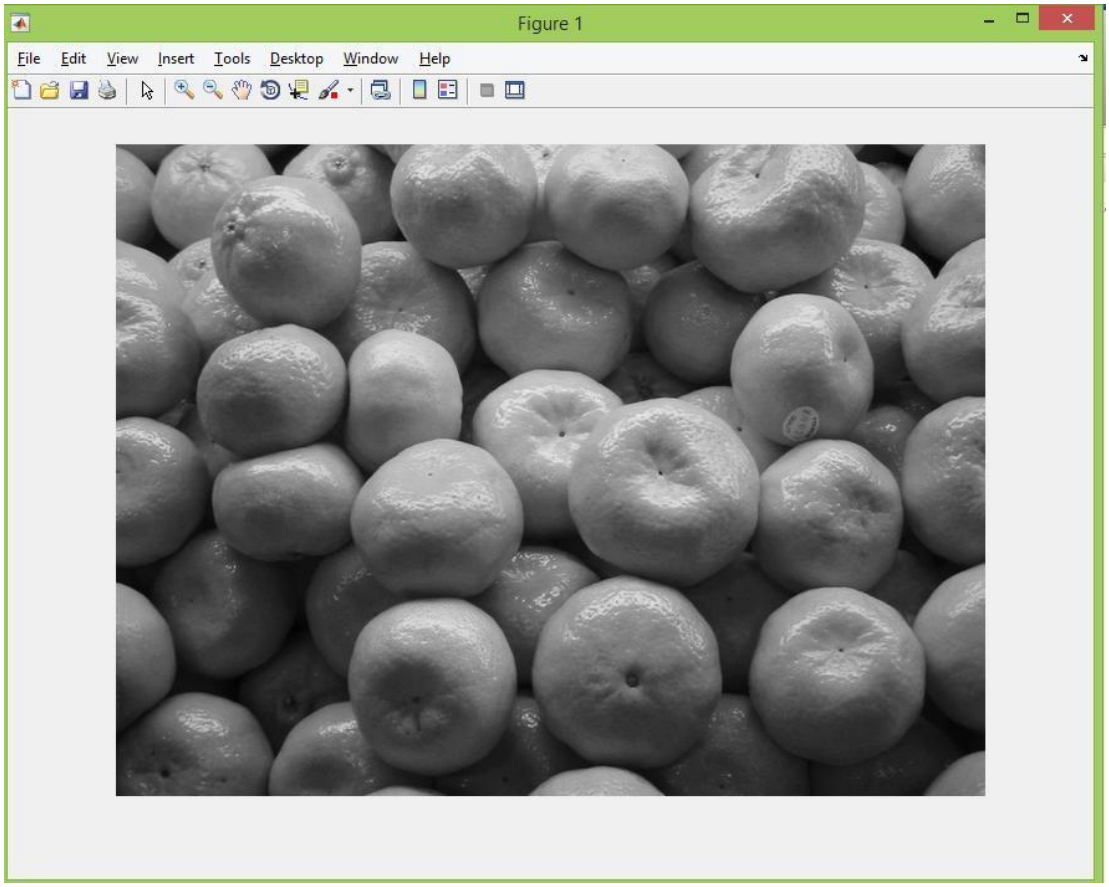


Рисунок 7.1 – Исходная картинка в виде полутонового изображения

```
hy=fspecial('sobel');
hx=hy';
Iy=imfilter(double(I), hy, 'replicate');
Ix=imfilter(double(I), hx, 'replicate');
gradmag=sqrt(Ix.^2+Iy.^2);
figure, imshow(gradmag,[]), title('значение градиента')
```

Для вычисления значения градиента используется оператор Собеля,функция imfilter и другие вычисления. Градиент имеет большие значения на границах объектов и небольшие (в большинстве случаев) вне границ объектов. Результат показан на рисунке 7.2.

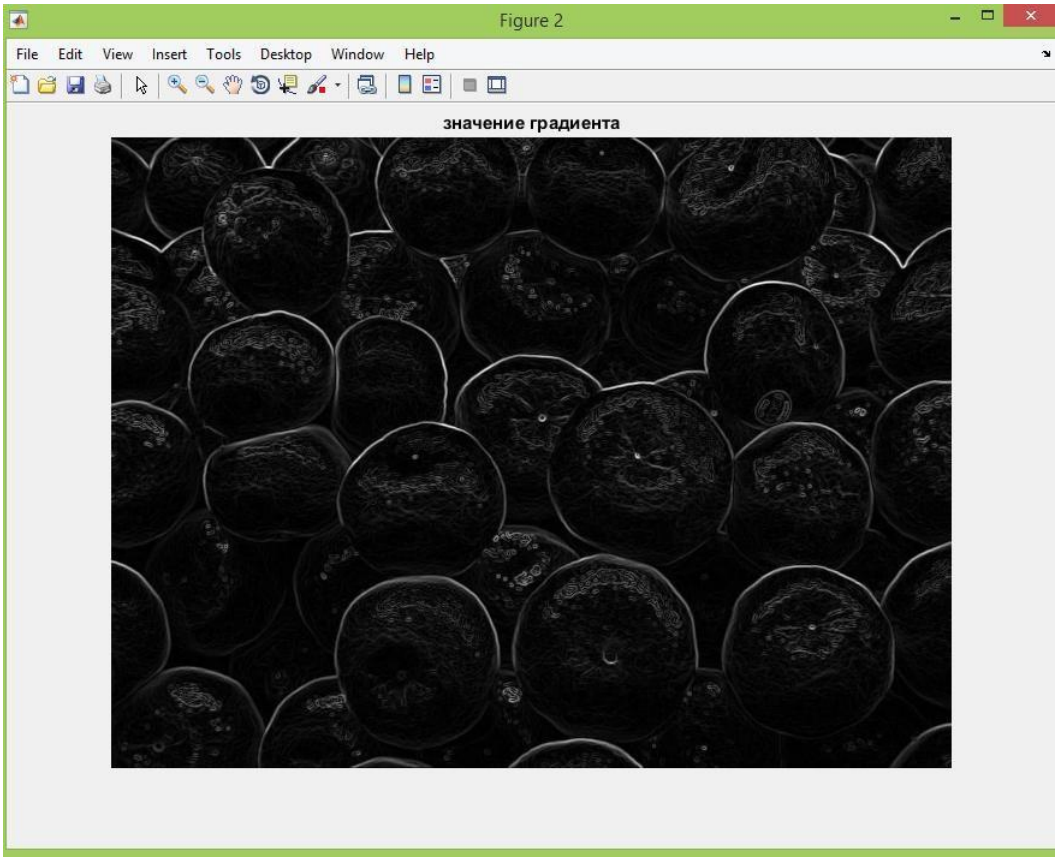


Рисунок 7.2 – Полученные значения градиента

Таким образом, вычислив значения градиента, можно приступить ксегментации изображений с помощью рассматриваемого метода маркерного водораздела (рисунок 7.3).

```
Q=watershed(gradmag);
Lrgb=label2rgb(Q);
figure, imshow(Lrgb), title('Lrgb')
```

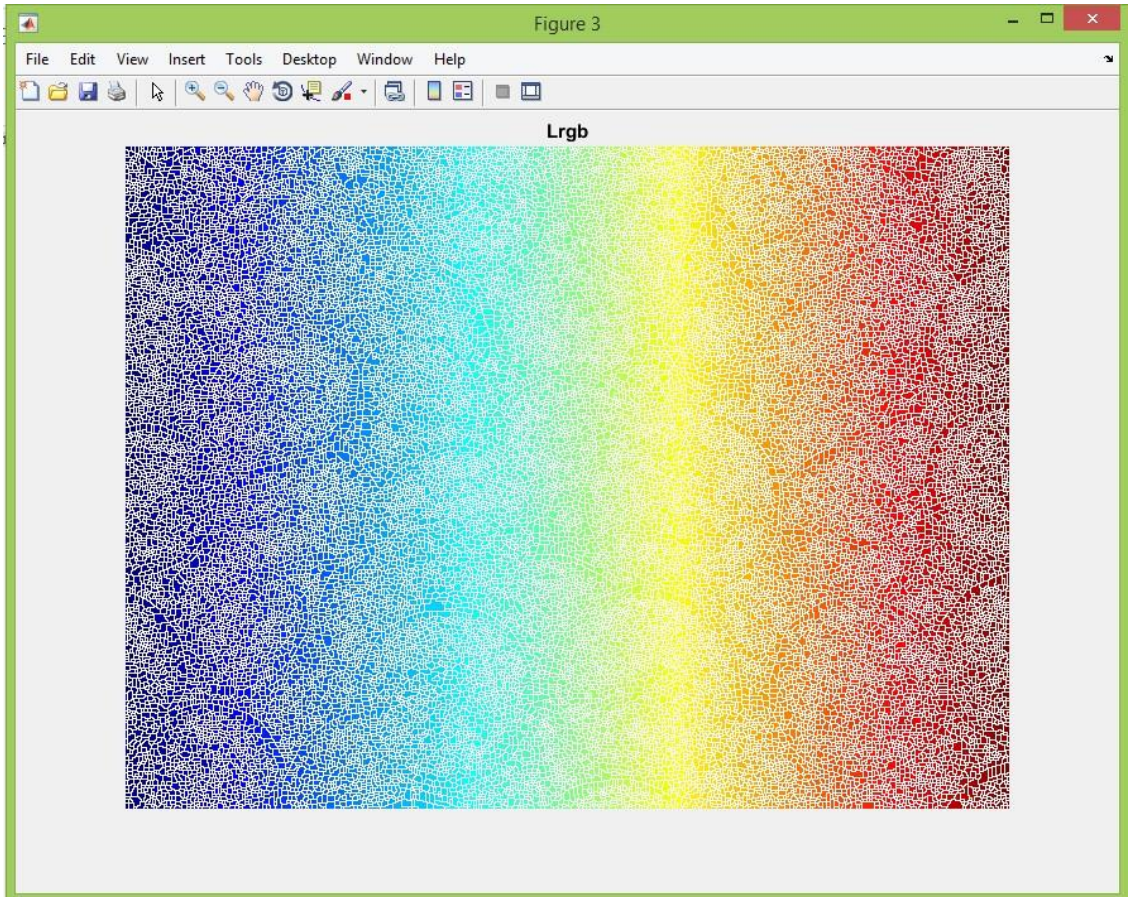


Рисунок 7.3 – Первичная сегментация с помощью маркерного водораздела

Однако, без проведения еще дополнительных вычислений, такаясегментация будет поверхностной.

Для маркировки объектов переднего плана могут использоваться различные процедуры. В этом примере будут использованы морфологические технологии, которые называются "раскрытие через восстановление" и "закрытие через восстановление". Эти операциипозволяют анализировать внутреннюю область объектов изображения с помощью функции **imregionalmax**.

Как было сказано выше, при проведении маркировки объектов переднего плана используются также морфологические операции.Рассмотрим некоторые из них и сравним. Сначала реализуем операцию раскрытия (рисунок 7.4) с использованием функции **imopen**.

```
se=strel('disk', 20);
Io=imopen(I, se);
figure, imshow(Io), title('Io')
```

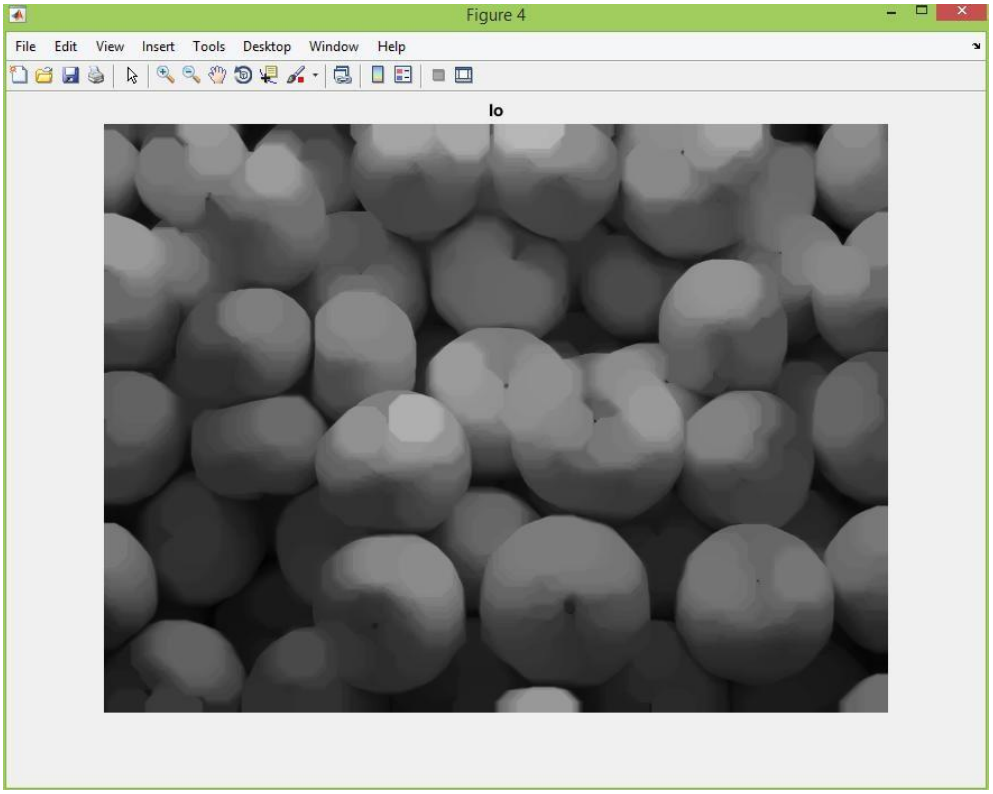


Рисунок 7.4 – Реализация операции раскрытия

Далее вычислим раскрытие с использованием функций **imerode** и **imreconstruct** (рисунок 7.5).

```
Ie=imerode(I, se);
Iobr=imreconstruct(Ie,  I);
figure, imshow(Iobr), title('Iobr')
```

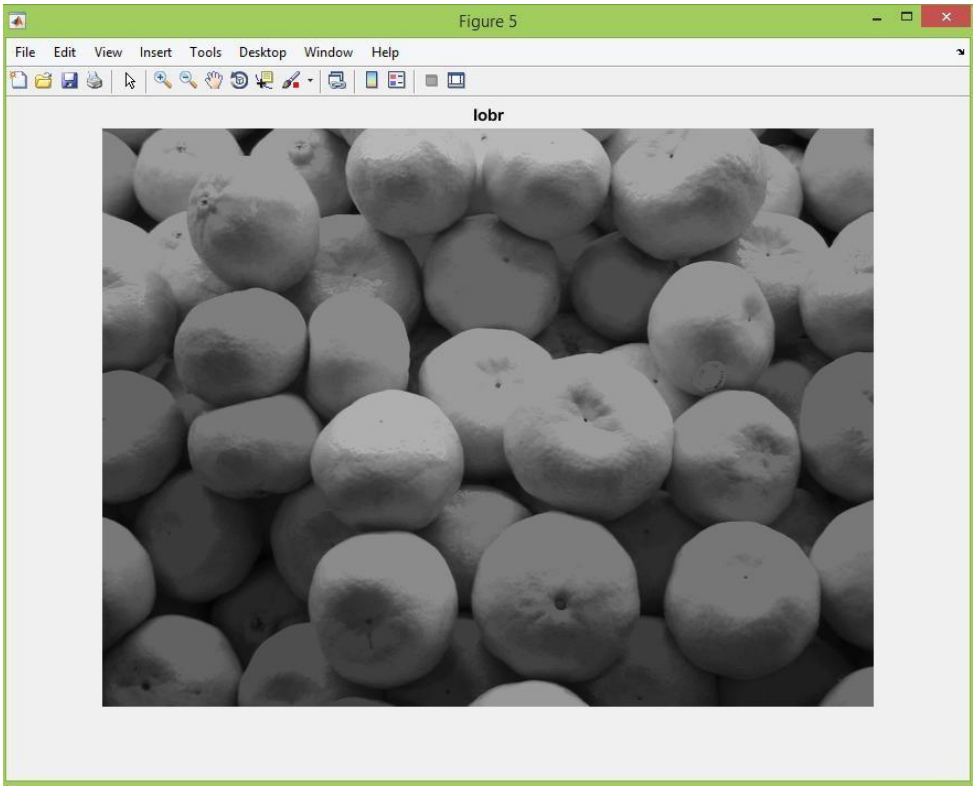


Рисунок 7.5 – Операция раскрытия

Последующие морфологические операции раскрытия и закрытия приведут к перемещению темных пятен и формированию маркеров. Проанализируем операции морфологического закрытия. Для этого сначала используем функцию **imclose** (рисунок 7.6):

```
Ioc=imclose(Io, se);
figure, imshow(Ioc), title('Ioc')
```

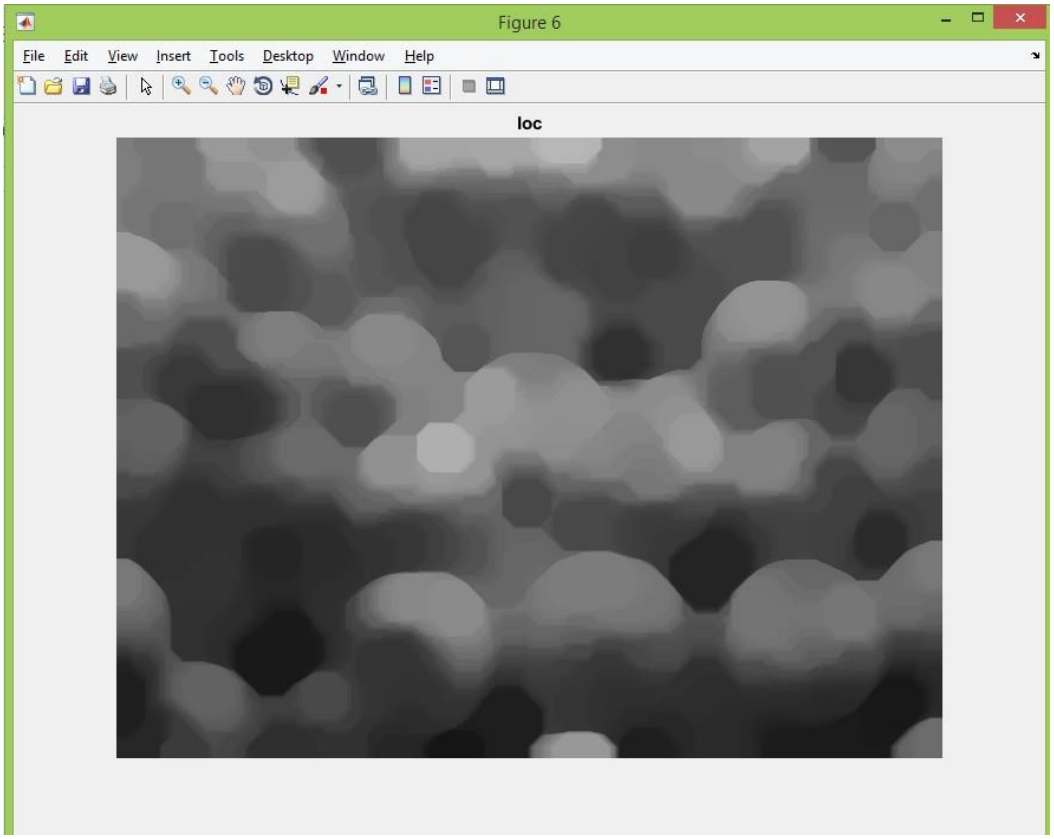


Рисунок 7.6 – Результат закрытия для изображения

Далее используем функцию **imdilate**, которая применяется вместе с функцией **imreconstruct**. Отметим, что для реализации операции **imreconstruct** необходимо провести операцию дополнения изображений (рисунок 7.7).

```
Iobrd=imdilate(Iobr, se);
Iobrcbr=imreconstruct(imcomplement(Iobrd), imcomplement(Iobr));
Iobrcbr=imcomplement(Iobrcbr);
figure, imshow(Iobrcbr), title('Iobrcbr')
```

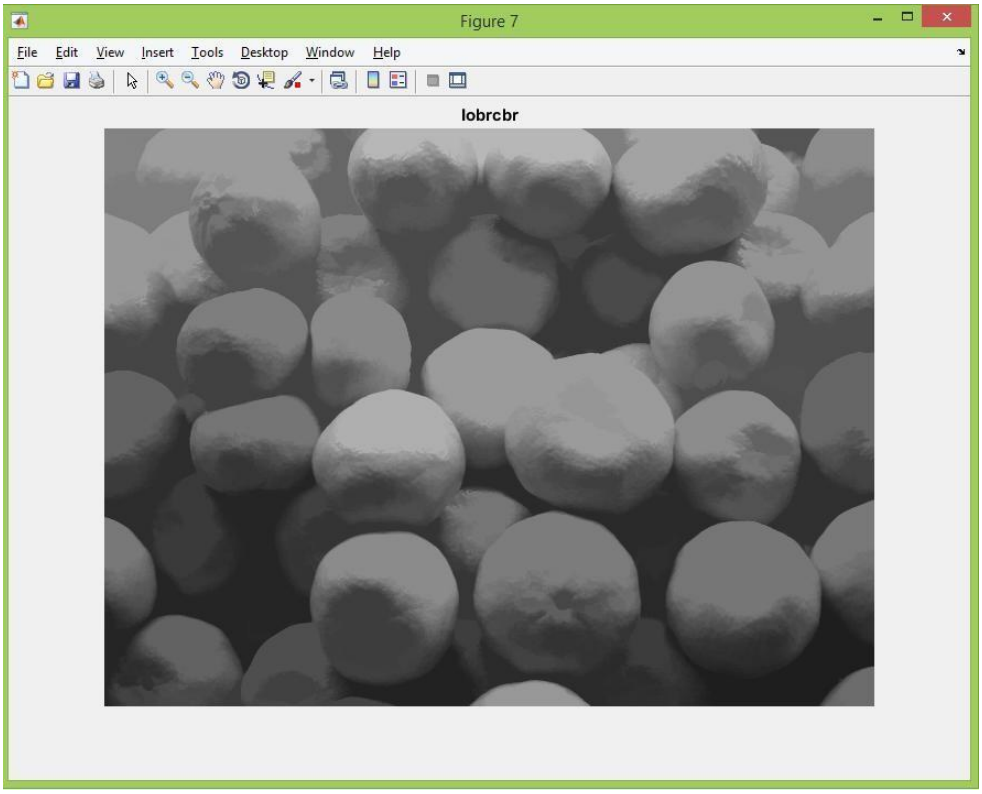


Рисунок 7.7 – Операция дополнения изображения

Сравнительный визуальный анализ **Iobrcbr** и **Ioc** показывает, что представленная реконструкция на основе морфологических операций открытия и закрытия является более эффективной в сравнении с стандартными операциями открытия и закрытия. Вычислим локальные максимумы Iobrcbr и получим маркеры переднего плана (рисунок 7.8).

```
fgm=imregionalmax(Iobrcbr);
figure, imshow(fgm), title('fgm')
```

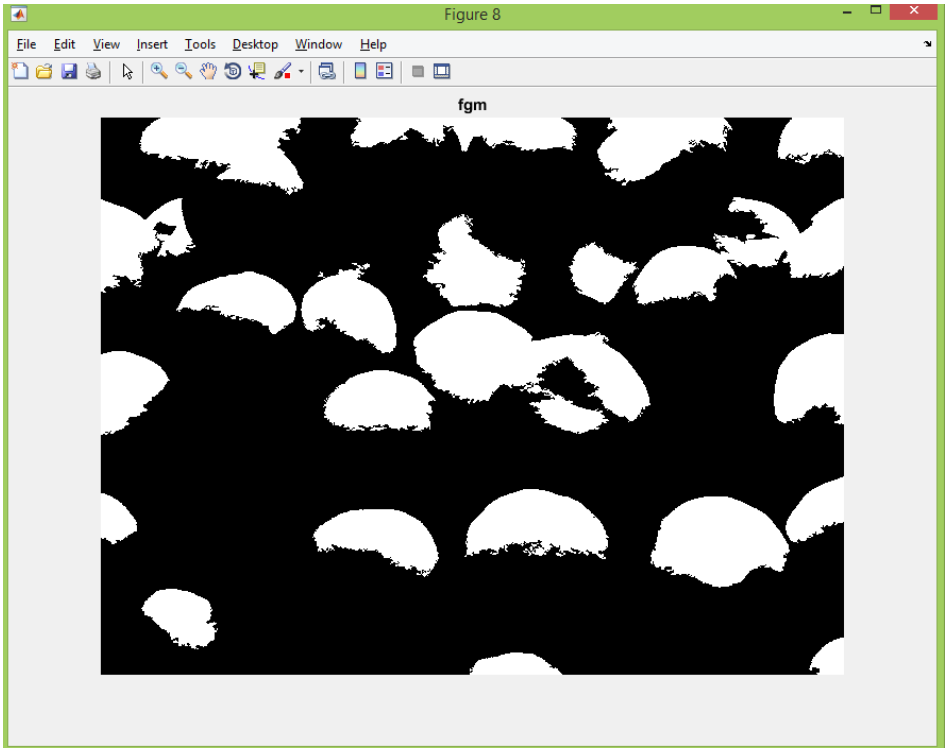


Рисунок 7.8 – Вычисление локальных маркеров

Наложим маркеры переднего плана на исходное изображение (рисунок 7.9).

```
I2=I;
I2(fgm)=255;
figure, imshow(I2), title('fgm, наложенное на исходное изображение')
```

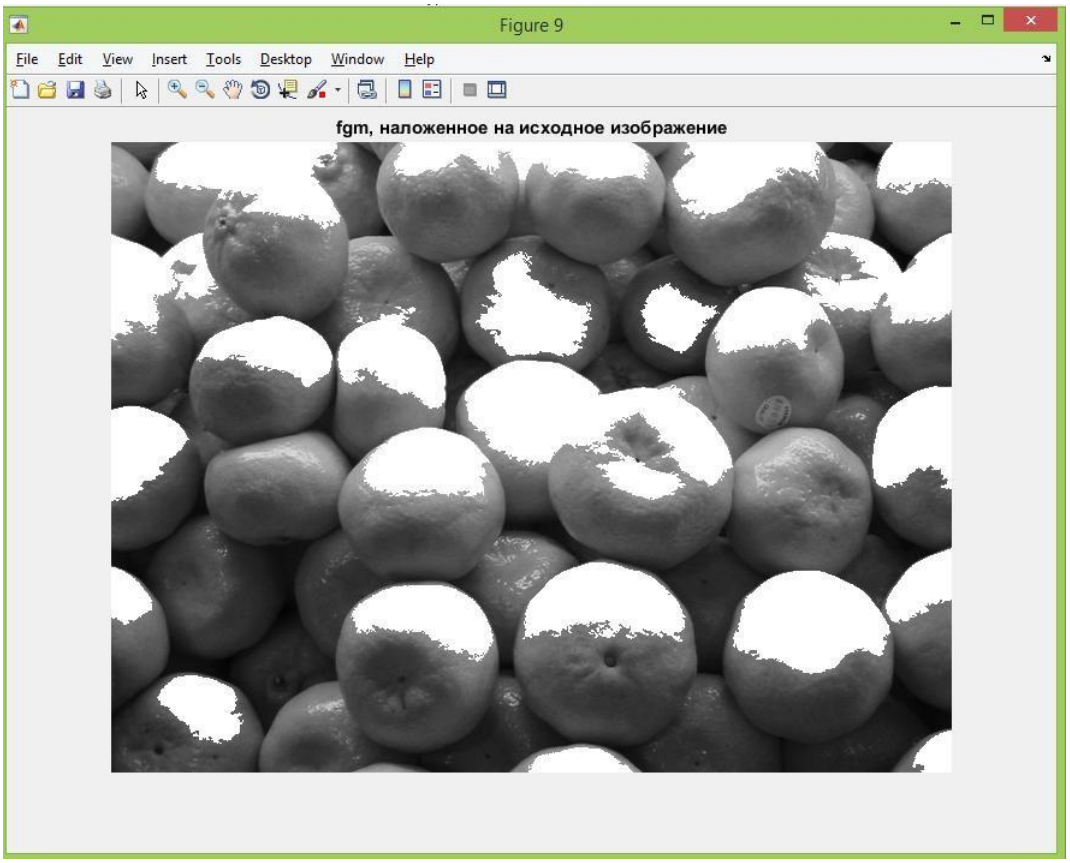


Рисунок 7.9 – Наложение маркеров переднего плана на исходное изображение

Отметим, что при этом некоторые скрытые или закрытые объекты изображения не являются маркированными. Это свойство влияет на формирование результата и многие такие объекты изображения не будут обработаны с точки зрения сегментации. Таким образом, маркеры переднего плана отображают границы только большинства объектов. Представленные таким образом границы подвергаются дальнейшей обработке. В частности, это могут быть морфологические операции.

В результате проведения такой операции (рисунок 7.10) пропадают отдельные изолированные пиксели изображения. Также можно использовать функцию **bwareaopen**, которая позволяет удалять заданное число пикселей.

```
se2=strel(ones(5, 5));
fgm2=imclose(fgm, se2);
fgm3=imerode(fgm2, se2);
fgm4=bwareaopen(fgm3, 20);
I3=I;
```

```
I3(fgm4)=255;
figure, imshow(I3)
title('fgm4, наложенное на исходное изображение')
```

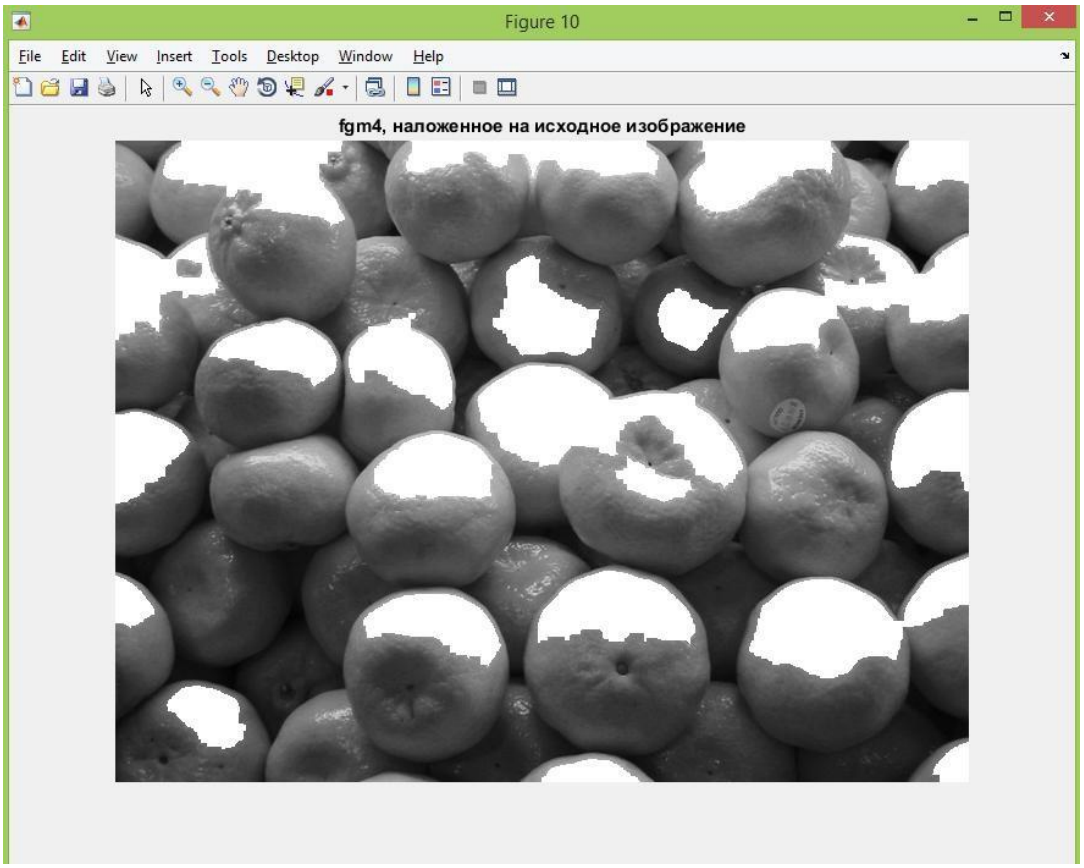


Рисунок 7.10 – Наложение на исходное изображение

Теперь проведем операцию маркирования фона. На изображении **Iobrcbr** темные пиксели относятся к фону. Таким образом, можно применить операцию пороговой обработки изображения, рисунок 7.11.

```
bw=im2bw(Iobrcbr, graythresh(Iobrcbr));
figure, imshow(bw), title('bw')
```

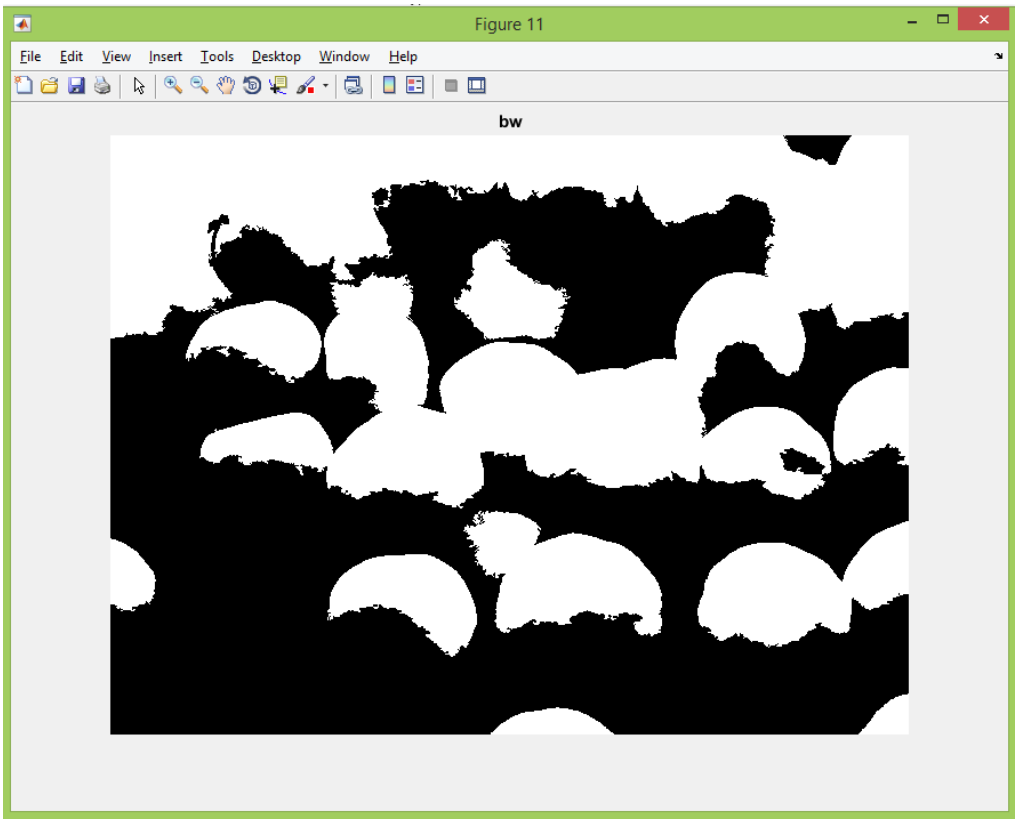


Рисунок 7.11 – Операция пороговой обработки изображения

Пиксели фона являются темными, однако нельзя просто провести морфологические операции над маркерами фона и получить границыобъектов, которые мы сегментируем. Мы хотим "утоньшить" фон таким образом, чтобы получить достоверный скелет изображения или, так называемый, передний план полутонового изображения. Это вычисляется с применением подхода по водоразделу и на основе измерения расстояний (до линий водораздела), рисунок 7.12.

```
D=bwdist(bw);
DL=watershed(D);
bgm=DL==0;
figure, imshow(bgm), title('bgm')
```

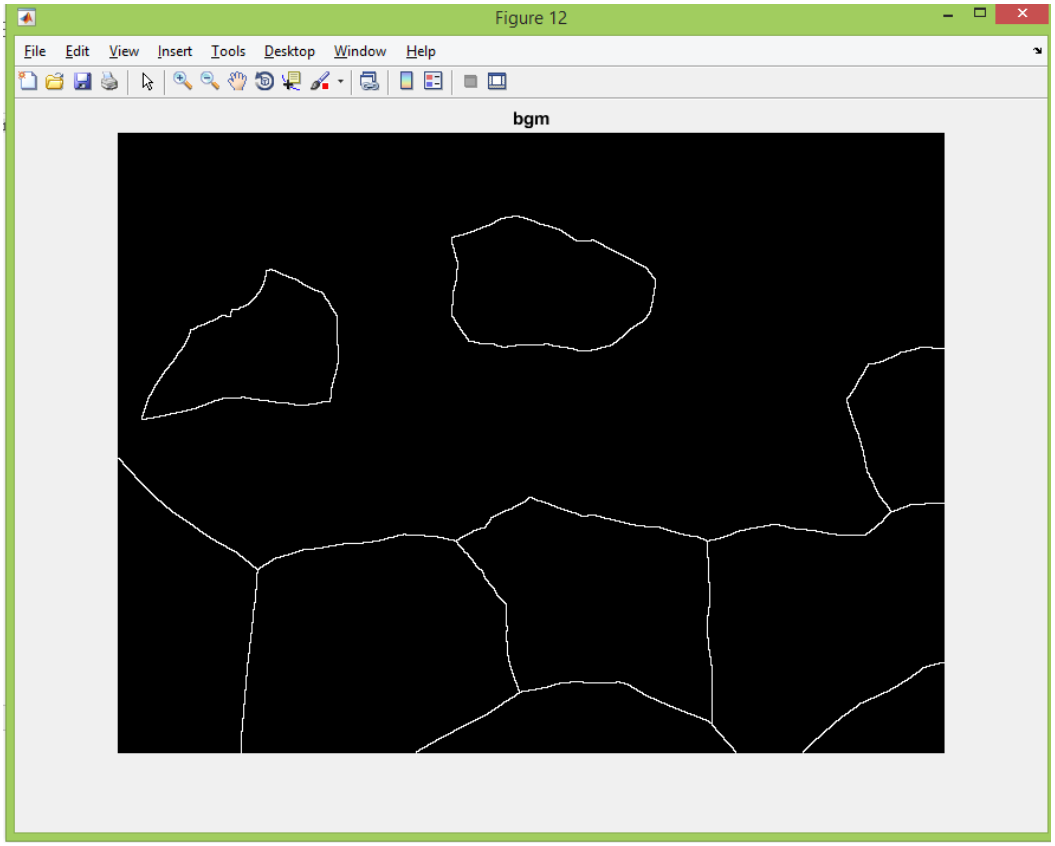


Рисунок 7.12 – Получение скелета изображения

Функция **imimposemin** может применяться для точного определения локальных минимумов изображения. На основании этого функция **imimposemin** также может корректировать значения градиентов на изображении и таким образом уточнять расположение маркеров переднего плана и фона.

И наконец, выполняется операция сегментации на основе водораздела спомощью **watershed()**.

Отобразим на исходном изображении наложенные маркеры переднего плана, маркеры фона и границы сегментированных объектов, рисунок 7.13.

```
gradmag2=imimposemin(gradmag, bgm | fgm4);
L=watershed(gradmag2);
I4=I;
I4(imdilate(L==0, ones(3, 3))|bgm|fgm4)=255;
figure, imshow(I4)
title('Маркеры и границы объектов, наложенные на исходное изображение')
```

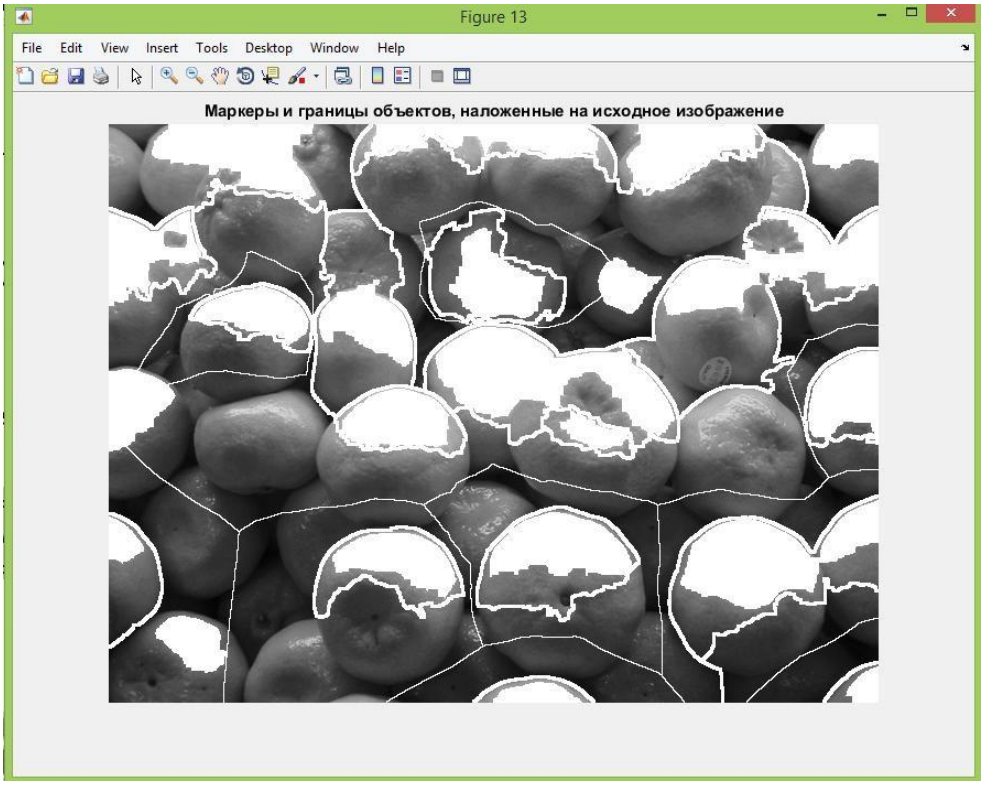


Рисунок 7.13 – Отображение на исходном изображении маркеров переднегоплана, маркеров фона и границы сегментированных объектов

В результате такого отображения можно визуально анализировать месторасположение маркеров переднего плана и фона.

Представляет интерес также отображение результатов обработки с помощью цветного изображения (рисунок 7.14). Матрица, которая генерируется функциями **watershed** и **bwlabel**, может быть конвертирована в truecolor-изображение посредством функции **label2rgb**.

```
Lrgb=label2rgb(L, 'jet', 'w', 'shuffle');
figure, imshow(Lrgb)
title('Lrgb')
```

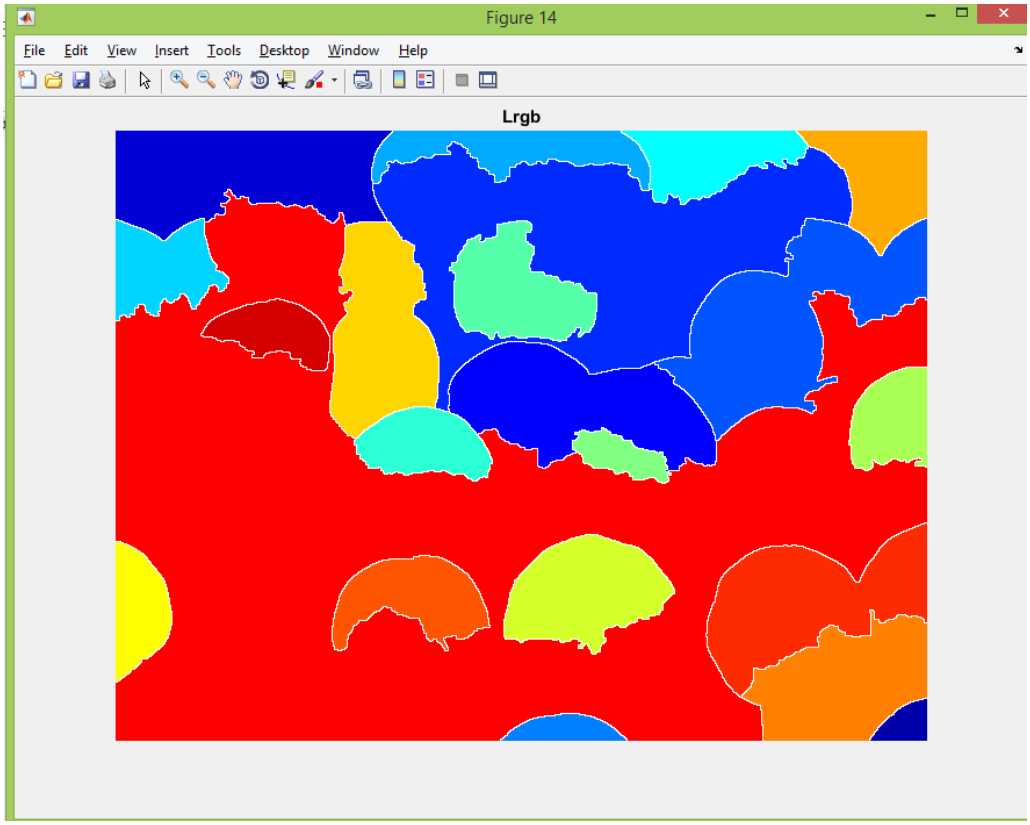


Рисунок 7.14 – Цветное отображение

Также можно использовать полупрозрачный режим для наложения псевдоцветовой матрицы меток поверх исходного изображения, рисунок 7.15.

```
figure, imshow(I), hold on
himage=imshow(Lrgb);
set(himage, 'AlphaData', 0.3);
title('Lrgb, наложенное на исходное изображение в полупрозрачном режиме')
```

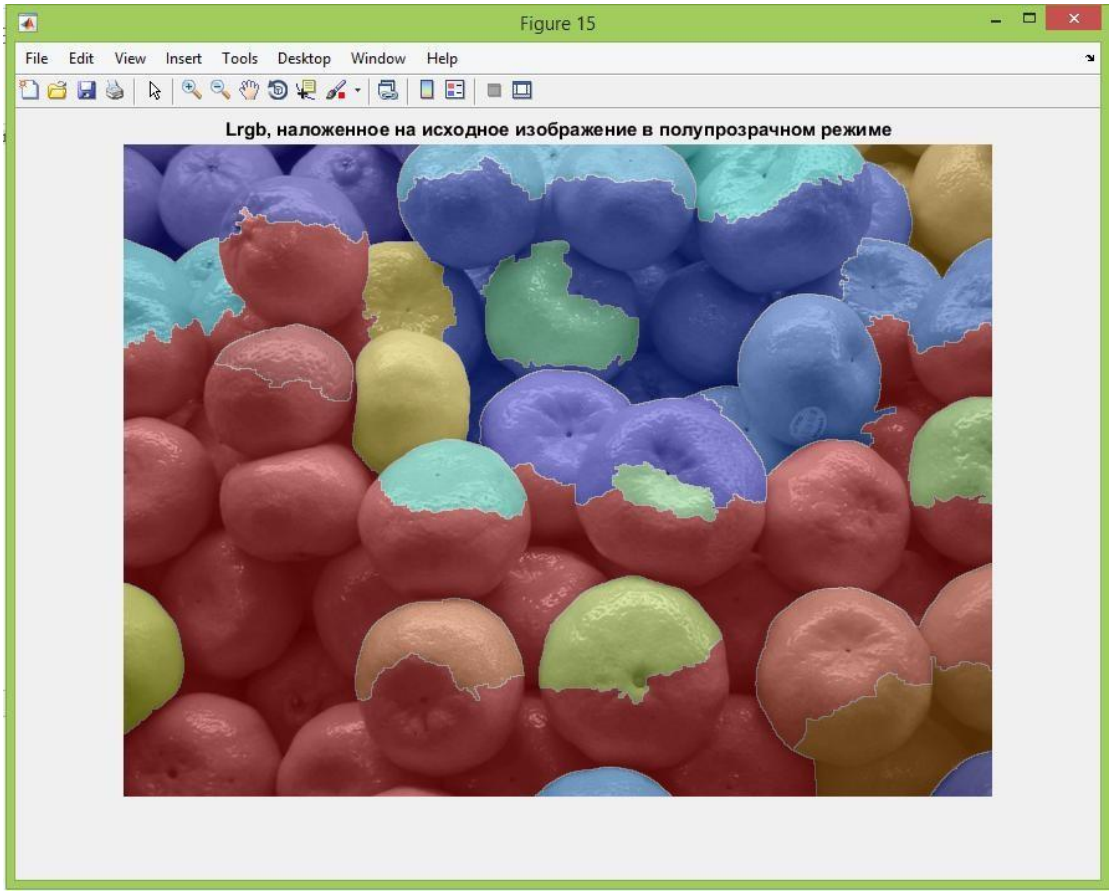


Рисунок 7.15 – Результат наложения на исходное изображение в полупрозрачном режиме

На практике также применяются и другие методы сегментации: на основе k-средних, различные морфологические операторы и т.д. В целом же задача сегментации – это задача из области искусственного интеллекта и одного универсального алгоритма не существует.

Практическая часть

Подберите 15 цветных изображений с различным семантическим содержанием (люди, фрукты и т.д.). Проведите в цикле сегментацию этих изображений методом водораздела, сделайте выводы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Основная литература:
1 Тропченко А.А. Методы вторичной обработки и распознавания изображений [Электронный ресурс] : учебное пособие / А.А. Тропченко, А.Ю. Тропченко. — Электрон. текстовые данные. — СПб. : Университет ИТМО, 2015. — 215 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/67277.html.
2. Стенли, Липпман.; Язык программирования C++ Электронный ресурс : Полное руководство / Липпман Стенли, Лажойе Жози ; пер. А. Слинкин. - Язык программирования C++,2019-04-19. - Саратов : Профобразование, 2017. - 1104 c. - Книга находится в премиум- версии ЭБС IPR BOOKS. - ISBN 978-5-4488-0136-5, экземпляров неограниченно

Дополнительная литература:
1. Основы теории обработки непрерывных контуров изображений : монография / Р.Г. Хафизов, А.А. Роженцов, Д.Г. Хафизов, С.А. Охотников ; Поволжский государственный технологический университет ; под общ. ред. Р. Г. Хафизов. - Йошкар-Ола : ПГТУ, 2015. - 172 с. : ил. - http://biblioclub.ru/. - Библиогр.: с. 132-141. - ISBN 978-5-8158-1606-0, экземпляров неограниченно
2. Рафаэл, Гонсалес. Цифровая обработка изображений Электронный ресурс / Гонсалес Рафаэл, Вудс Ричард ; пер.: Л. И. Рубанов, П. А. Чочиа ; ред. П. А. Чочиа. - Москва : Техносфера, 2012. - 1104 c. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-94836-331-8, экземпляров неограниченно.

Методическая литература

1. Комашинский В.И.Нейронные сети и их применение в системах управления и связи/ Д.А.Смирнов. - М:Горячая линия-Телеком,2010. - 94с. – с. 88

2. Нейронные сети: история развития теории : учеб. пособие для вузов / под ред. А. И. Галушкина, Я. З. Цыпкина. - М. : ИПРЖР, 2015. - 840 с. - (Нейрокомпьютеры и их применение, Кн. 5). - Гриф: Рек. МО. - Прил.: с. 826-834. – ISBN 5-93108-007-4

3. Ширяев, В. И. Финансовые рынки. Нейронные сети, хаос и нелинейная динамика : [учеб. пособие] / В.И. Ширяев. - 3-е изд. - М. : КРАСАНД, 2010. -

232 с. : ил. - На учебнике гриф: Доп.УМО. - Библиогр.: с. 210-221. - ISBN 978-5-396-00273-9, экземпляров 1

Интернет-ресурсы

4. http://algolist.manual.ru/ai/ga/ga1.php
5. http://rain.ifmo.ru/cat/view.php/theory/unsorted/genetic-2005
6. https://basegroup.ru/community/articles/ga-math
7. https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BD%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC
8. http://www.prodav.narod.ru/dsp/index.html - Давыдов А.В. Цифровая обработка сигналов. Тематические лекции: Учебное пособие в электронной форме. – Екатеринбург, УГГУ