Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования
Санкт-Петербургский горный университет

Кафедра автоматизации технологических процессов и производств

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

Методические указания к практическим занятиям для студентов магистратуры направления 15.04.04

САНКТ-ПЕТЕРБУРГ
2019

УДК 66.012-52 (073)

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ: Методические указания к практическим занятиям / Санкт-Петербургский горный университет. Сост. *Н.И. Котелева.* СПб, 2019. 32 с.

Содержат общие теоретические сведения по темам практических занятий, перечень работ и заданий, необходимых к выполнению во время практических работ, пример выполнения заданий и контрольные вопросы по темам практических работ.

Методические указания предназначены для студентов магистратуры направления 15.04.04 «Автоматизация технологических процессов и производств» всех направленностей (профилей) программы.

Научный редактор проф. *В.Ю. Бажин*

Рецензент канд. техн. наук *В.В. Васильев* (ООО «ТОМС инжиниринг»)

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

Методические указания к практическим занятиям для студентов магистратуры направления 15.04.04

Сост. *Н.И. Котелева*

Печатается с оригинал-макета, подготовленного кафедрой автоматизации технологических процессов и производств

Ответственный за выпуск *Н.И. Котелева*

Лицензия ИД № 06517 от 09.01.2002

Подписано к печати 28.10.2019. Формат 60×84/16.
Усл. печ. л. 1,8. Усл.кр.-отт. 1,8. Уч.-изд.л. 1,5. Тираж 50 экз. Заказ 922. С 309.

Санкт-Петербургский горный университет
РИЦ Санкт-Петербургского горного университета
Адрес университета и РИЦ: 199106 Санкт-Петербург, 21-я линия, 2

ВВЕДЕНИЕ

Дисциплина «Интеллектуальные системы» призвана сформировать у студентов базовые знания в области представления и обработки знаний в интеллектуальных системах, методов построения логических, продукционных, сетевых моделей и их использования в интеллектуальных системах различного назначения: экспертных системах, нечетких системах, системах поддержки принятия решений, нейросетевых и генетических алгоритмах, освоение навыков в областях решения задач проектирования и управления на основе методов искусственного интеллекта, разработки программного обеспечения для современных интеллектуальных систем, а также методологией проведения научно-исследовательских работ.

Представленные методические указания к выполнению практических занятий по дисциплине «Интеллектуальные системы» содержат общие теоретические сведения по темам практических занятий, перечень работ и заданий, необходимых к выполнению во время практических работ, пример выполнения заданий и контрольные вопросы по темам практических работ.

ОБЩИЕ СВЕДЕНИЯ ОБ ОРГАНИЗАЦИИ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

Практические занятия – обязательная и неотъемлемая часть учебной работы студента по дисциплине «Интеллектуальные системы».

Целью практических занятий является совершенствование умений и навыков решения практических задач. Главным содержанием этого вида учебных занятий является работа каждого обучающегося по овладению практическими умениями и навыками профессиональной деятельности.

Перечень работ, необходимых к выполнению во время практических занятий, а также срок сдачи отчетов по данным видам работ представлен в специальном документе «График самостоятельных работ студента».

В настоящих методических указаниях приведена информация о выполнении работ по первому разделу дисциплины

«Интеллектуальные системы» - «Нейронные сети в интеллектуальных системах». Данный раздел состоит из двух расчетно-графических работ.

РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА №1

Использование нейронной сети для решения задач регрессии

ЦЕЛЬ РАБОТЫ

Научиться применять теорию нейронной сети для решения задачи регрессии на примере решения математического уравнения.

ОБЩИЕ СВЕДЕНИЯ

Нейронная сеть – это искусственный математический аналог человеческого мозга. Как известно из биологии мозг состоит из нейронов и синапсов. Искусственная нейронная сеть также состоит из совокупности нейронов и связей между ними синапсов. Пример структуры нейронной сети представлен на рисунке 1.1.

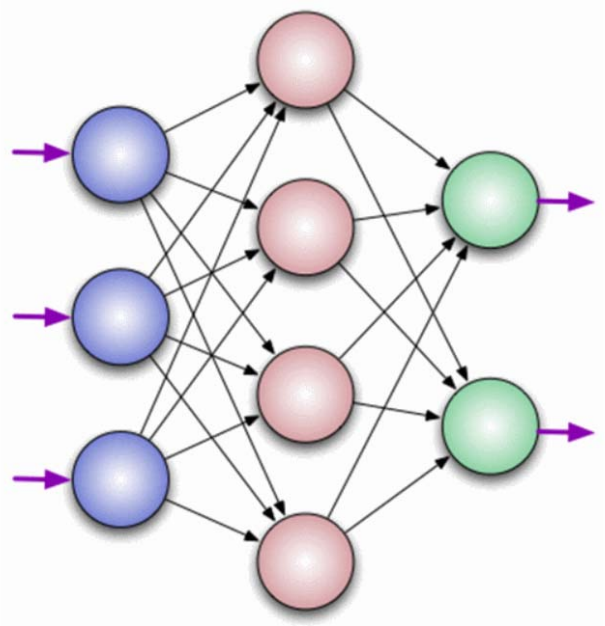


Рис 1.1. Пример структуры нейронной сети.

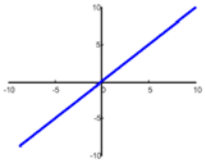
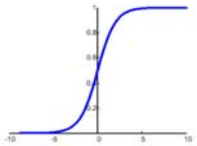
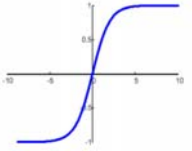
В типовой структуре нейронной сети существует входной слой, скрытый слой и выходной.

На рисунке представлена сеть, состоящая из одного скрытого слоя, содержащего 4 нейрона. Входной слой содержит три нейрона и выходной 2 нейрона. Число нейронов во входном и выходном слоях обычно определяется условиями поставленной задачи. Число нейронов в скрытом слое определяется специальными правилами или экспериментальным путем.

Каждый нейрон может оперировать только с числами в диапазоне [0,1] или [-1,1]. Для приведения значений к данным диапазонам в нейронах используется функция активации (Таблица 1.1). Функция активации нормализует входные данные и приводит их к диапазонам [0,1] или [-1,1]. Функций активации множество, основными из них является Линейная, Сигмоидальная (Логистическая) и Гиперболический тангенс.

Таблица 1.1

Функции активации

Линейная	Сигмоидальная	Гиперболический тангенс
$f(x) = x$	$f(x) = \frac{1}{1 + e^{-x}}$	$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$
		

Процесс создания нейронной сети содержит в себе ряд обязательных этапов, среди которых:

1. Формирование обучающей выборки
2. Формирование тестовой выборки
3. Определение структуры и архитектуры сети
4. Обучение сети
5. Тестирование работоспособности сети
6. Дообучение сети (при необходимости)

Использование ПО Matlab для создания нейронной сети

Программное обеспечение Matlab – специализированное программное обеспечение для решения различных инженерных задач. Данное программное обеспечение содержит специальные пакеты для разработки нейронных сетей и использования нейронных сетей при управлении технологическими процессами. Нейронную сеть в ПО Matlab можно создавать двумя способами – с помощью специального приложения Neural Network toolbox и скриптом с помощью встроенных специальных функций. В данной работе представлены алгоритмы создания нейронной сети двумя, представленными выше вариантами.

Создание нейронной сети с помощью специального приложения Neural Network toolbox

1. Запустите ПО Matlab (Нажмите ярлык Matlab.exe на рабочем столе)
2. File-New script
3. Сформируем обучающие и тестовые выборки

Размер обучающей выборки – 1000 значений, размер тестовой выборки – 300 значений.

Создадим матрицы a, b, c размерностью 1x1300 (1 строка, 1300 столбцов), матрицы заполним случайными числами, распределенными по нормальному закону со средним значением равным нулю и стандартным отклонением равным единице (см. рис. 1.2).

```
a= random ('Normal',0,1,1,1300);
b= random ('Normal',0,1,1,1300);
c= random ('Normal',0,1,1,1300);
```

Рис. 1.2. Задание матриц a, b, c в MATLAB

4. Согласно уравнению $d=a^2+2b+3c$ зададим матрицу d. (см. рис. 1.3)

```
for i=1:1300
    d(1,i)=a(1,i)*a(1,i)+2*b(1,i)+3*c(1,i);
end
```

Рис. 1.3. Задание матрицы d в MATLAB

5. Запустим код на исполнениею В специальной рабочей зоне Workspace можно увидеть созданные нами переменные и их значения. Кликнув мышью на любую из переменных откуроются в специальном окне ее значения.

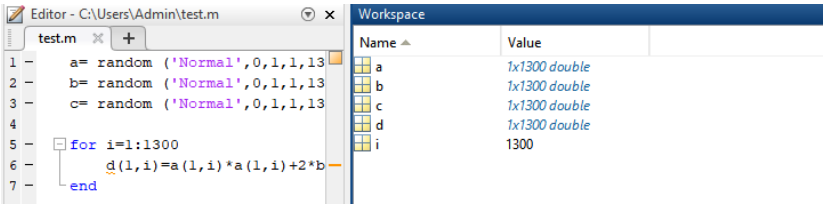


Рис. 1.4. Пример рабочей зоны Workspace

6. Зададим матрицу xvh входных значений, составленную путем объединения матриц a, b, и c (см. рис. 1.5). Таким образом, xvh - обучающая выборка для будущей нейронной сети. Данная обучающая выборка состоит из тройки чисел a,b,c (входы модели), которая однозначно соответствует числу d (выход модели). Всего обучающая выборка содержит 1000 значений, то есть соответствий тройки чисел a,b,c числу d. По данной обучающей выборке инструмент создания нейросетевой модели Neural Network Fitting Tool построит модель нейронной сети, а затем обучит на примере 1000 сэмплов вычислять значение d при заданных числах a,b,c.

```
for i=1:1000
    xvh (1,i)=a(1,i);
    xvh (2,i)=b(1,i);
    xvh (3,i)=c(1,i);
    dh (1,i)=d(1,i);
end
```

Рис. 1.5. Задание обучающей выборки xvh

Тестовая выборка содержит 300 значений. Для ее формирования необходимо ввести код (рисунок 1.6.)

```
for i=1001:1300
    xvt (1,i-1000)=a(1,i);
    xvt (2,i-1000)=b(1,i);
    xvt (3,i-1000)=c(1,i);
    dt (1,i-1000)=d(1,i);
end
```

Рис. 1.6. Задание тестовой выборки xvh

7. Создание нейронной сети

Откройте вкладку Apps и откройте Neural Fitting app. Если его нет в списке введите сверху в строке поиска имя данного приложения (рисунок 1.7).

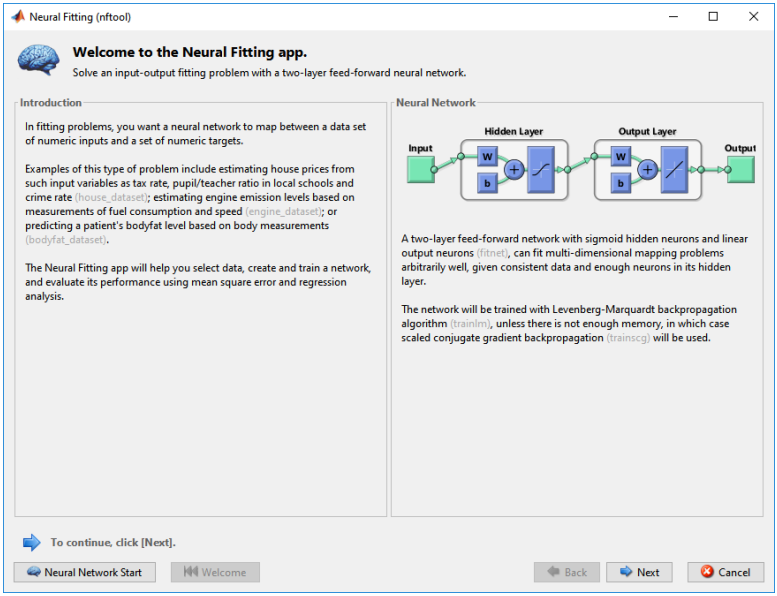


Рис. 1.7. Окно приложения Neural Fitting

Данное приложение позволяет настраивать, обучать и тестировать нейронную сеть. На второй вкладке выбираем вход и выход обучающей выборки (Рисунок 1.8)

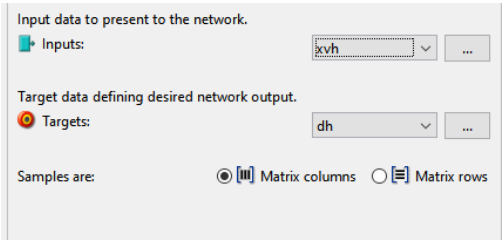


Рис. 1.8. Вход и выход обучающей выборки

В двух последующих вкладках смотрим настройки сети ее структуру и др. параметры, в третьей вкладке нажимаем кнопу

обучить сеть. В конце обучения можно увидеть следующую форму (Рисунок 1.9)

Рис. 1.9. Окно обучения нейронной сети.

8. Создание и обучение нейронной сети прошло успешно, так как выход модели соответствует значению в выборке (см. рис. 1.10).

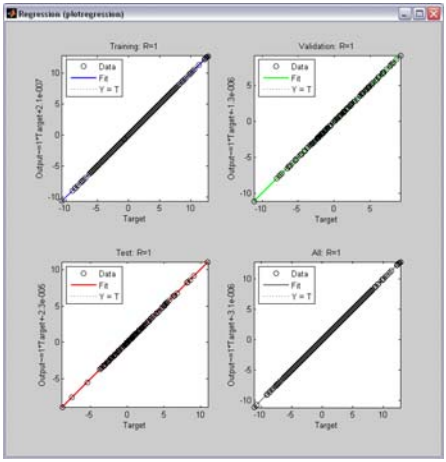


Рис. 1.10. Коэффициенты корреляции обучающей, валидационной и тестовой выборок

9. Протестируем работу сети. Перейдем на следующую вкладку и выберем тестовые выборки (Рисунок 1.11)

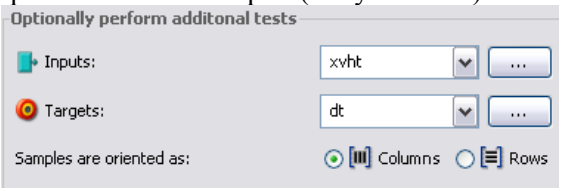


Рис. 1.11. Тестовая выборка

10. Проверка модели по тестовой выборке прошла успешно (см. рис. 1.12).

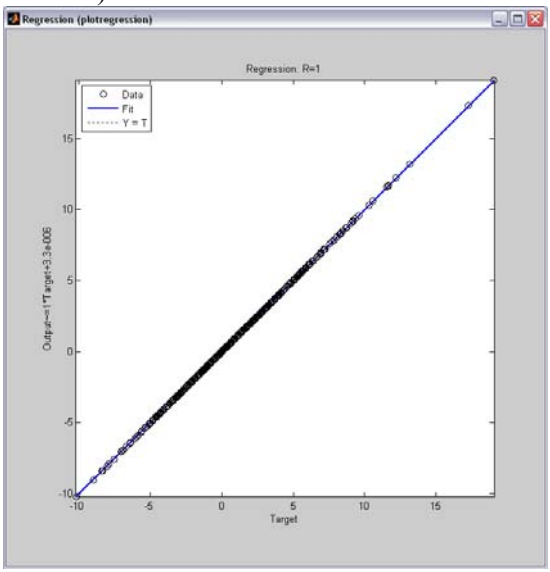
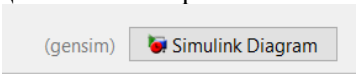


Рис. 1.12. Проверка модели по тестовой выборке

11. Создадим модель нейронной сети, которая может быть интегрирована в различные приложения. Создадим модель на примере среды моделирования работы АСУТП в динамическом режиме Simulink. Для этого в приложении NNF нажмите кнопку Simulink Diagram 

откроется окно с блоком модели нейронной сети (Рисунок 1.13).

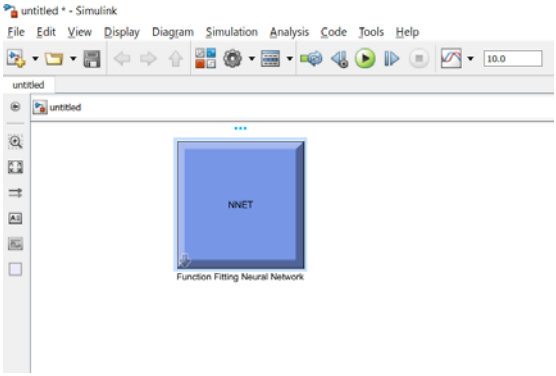


Рис. 1.13. Модель нейронной сети в Simulink.

12. Убедимся на примере, что нейронная сеть действительно правильно рассчитывает число d по заданным a,b,c. Для этого в MATLAB Simulink создадим модель, в которой на дисплей выведем значение по модели (Display) и значение по аналитическому выражению (Display1) (см. рис. 1.14).

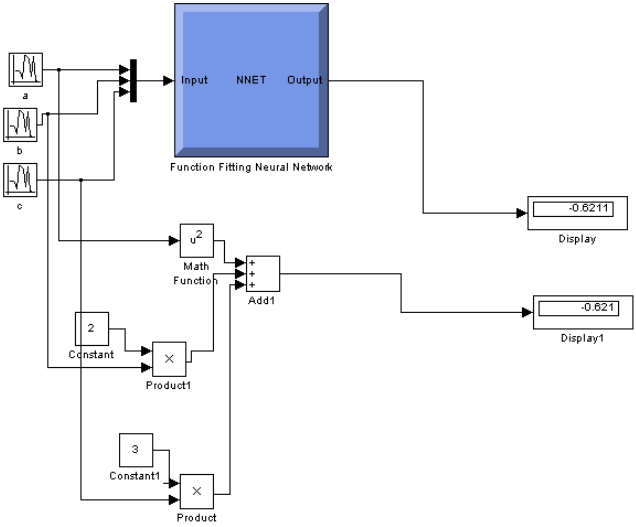


Рис. 1.14. Проверка модели

13. Как видно из рис. 1.14, значения по модели и по выражению совпадают. Таким образом, модель нейронной сети готова.

Создание нейронной сети в ПО Matlab без использования специального пакета инструментов для создания нейронной сети

1. Формирование обучающей и тестовой выборок для работы нейронной сети происходит одинаковым образом вне зависимости от используемых инструментов для создания сети. Поэтому для формирования массивов обучающей и тестовой выборок необходимо выполнить пункты 1-6 из предыдущего раздела.

2. Создаем нейронную сеть (рис. 1.15)

```
net = feedforwardnet(20, 'trainbr');
net = train(net,xvh,dh);
view(net)
y = net(xvh);
perf = perform(net,y,dh);
```

Рис. 1.15. Код создания нейронной сети

Поясним каждую команду, используемую в коде:

- net – название будущей нейронной сети;
- feedforwardnet (hiddenSizes,trainFcn) – команда, создающая нейронную сеть с заданными параметрами: количество нейронов в скрытом слое (hiddenSizes. по умолчанию 10) и функция обучения (trainFcn, по умолчанию = 'trainlm'). В дословном переводе feedforwardnet – сети с прямой связью. Первый уровень данной сети – входной слой. Количество нейронов во входном слое определятся размром входного массива. Каждый последующий слой имеет соединение с предыдущим. Последний уровень сети – выходной слой. Число нейронов в выходном слое определяется размером выходного массива. Сети прямой связи могут использоваться для поиска любого вида зависимости между входом и выходом сети;
- train - эта функция обучает простую нейронную сеть. Для обучения более сложных сетей (например, сверхточных или

LSTM нейронных сетей) необходимо использовать функцию trainNetwork;

- view – покажет структуру настраиваемой сети, определенной на первом шаге. На рисунке 1.16 представлен внешний вид структуры нейронной сети, реализованный в данном примере;
- net – функция, ссылающаяся на первоначально созданный нами объект net (нейронную сеть), позволяющая промоделировать работу нейронной сети. В качестве аргумента необходимо задать вектор входных значений. В данном примере аргументом является обучающая выборка xvh;
- perform – функция, позволяющая выполнить оценку производительности работы нейронной сети. По умолчанию в качестве данной функции используется среднеквадратичная ошибка.

3. Промоделируем работу созданной нейронной сети. В качесва вектора входных значений в данном случае будемиспользовать созданную раннее тестовую выборку.

```
test_prim_1=sim(net,xvt);
```

4. Создадим график расположения точек действительных значений выборки на который методом наложения добавим точки, полученные в результате работы нейронной сети. На рисунке 1.16 представлен фрагмент кода.

```
for i=1:300
    atest(1,i)=xvt(1,i);
    btest(1,i)=xvt(2,i);
    ctest(1,i)=xvt(3,i);
end

figure (1)
h1 = scatter(atest,dt,'o');
hold on
h2 = scatter(atest,test_prim_1,'x');
title('Сравнение обучающей и тестовой выборок')
xlabel('a')
ylabel('d')
```

Рис. 1.16. Фрагмент кода тестирования работы сети

В данном коде первая функция позволяет разбить вектор тестовой выборки отдельно на составляющие a, b и c. Функция figure () позволяет создать новый график, функция scatter позволяет создать точечную диаграмму с параметрами title - имя графика, xlabel – Подписи оси x и ylabel – подписи оси y. Полученный график представлен на рисунке 1.17

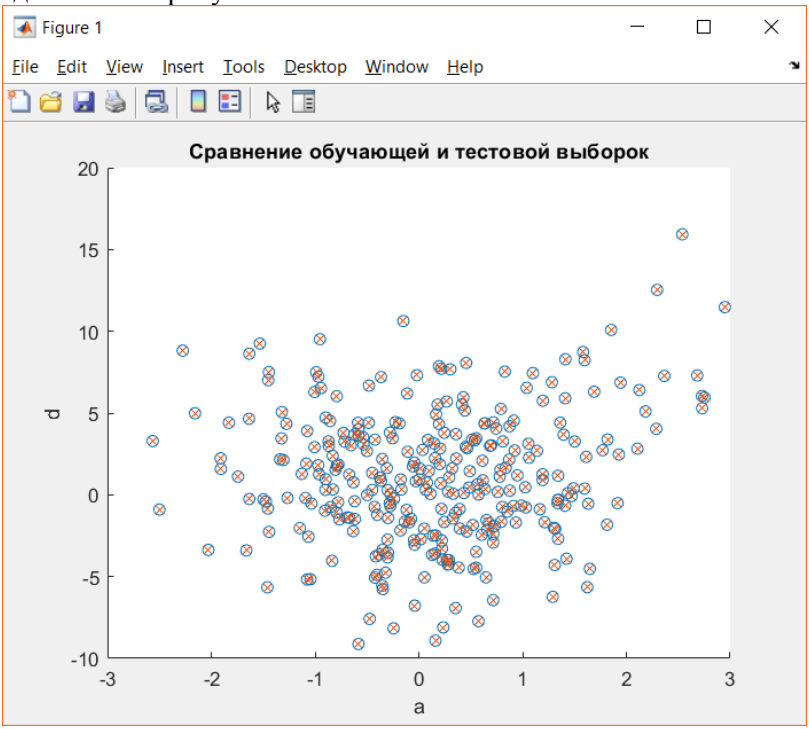


Рис.1.17. Результаты работы нейронной сети

5. Допишите самостоятельно код для отображения результатов по параметрам b и c.

6. Рассчитайте ошибку работы нейронной сети (рисунок 1.18)

```
err1=test_prim_1-dt;
error= sum(err1)/300;
```

Рис. 1.18. Расчет ошибки работы нейронной сети

7. В случае, если ошибка не существенна нейронная сеть готова к использованию. В случае если ошибка существенна необходимо переобучить нейронную сеть. Для этого существуют различные способы – от изменения структуры сети до изменения различных компонент сети, способа обучения и т.п.

Создание GUI интерфейса в ПО Matlab для демонстрации работы нейронной сети

GUI интерфейс – Graphical User Interface – специальное программное приложение, позволяющее пользователю работать с создаваемой нами программой. В данном разделе будет указан способ создания пользовательского интерфейса для решаемой нами задачи.

Разрабатываемый пользовательский интерфейс должен включать в себя:

- поля для ввода коэффициентов A, B и C;
- поля для вывода результатов вычислений по уравнению и по нейронной сети;
- кнопки запуска расчетов и переобучения нейронной сети (для случая, когда коэффициенты A, B или C значительно отличаются от оных из обучающей выборки);
- небольшие элементы оформления, поясняющие поля (например, «Введите A» и тому подобное).

Для создания графического интерфейса пользователя необходимо выполнить следящие действия:

1. Вызовите специальное приложение, позволяющее создать графический интерфейс. Для этого в командной строке задайте команду appdesigner. Данная команда откроет специальное окно для разработки пользовательского интерфейса (рисунок 1.19).

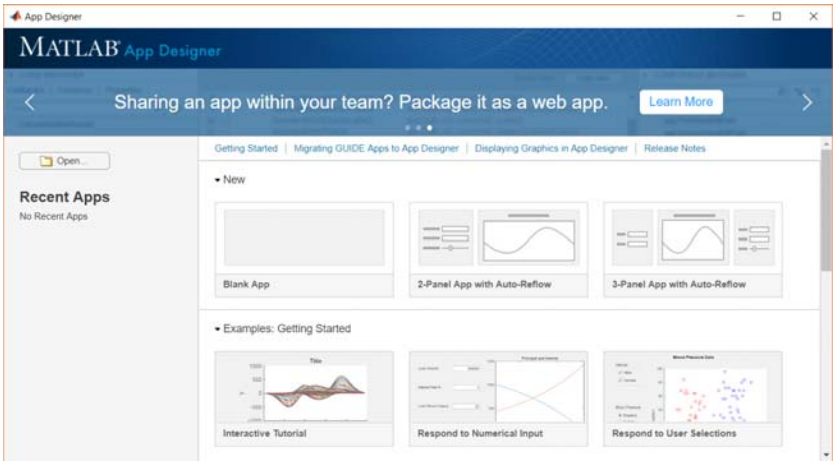


Рис. 1.19. Стартовое окно разработки графического интерфейса GUI

2. В качестве шаблона выберете чистый шаблон (BlankApp).

3. Расставьте необходимые элементы на форму (рис.1.20)

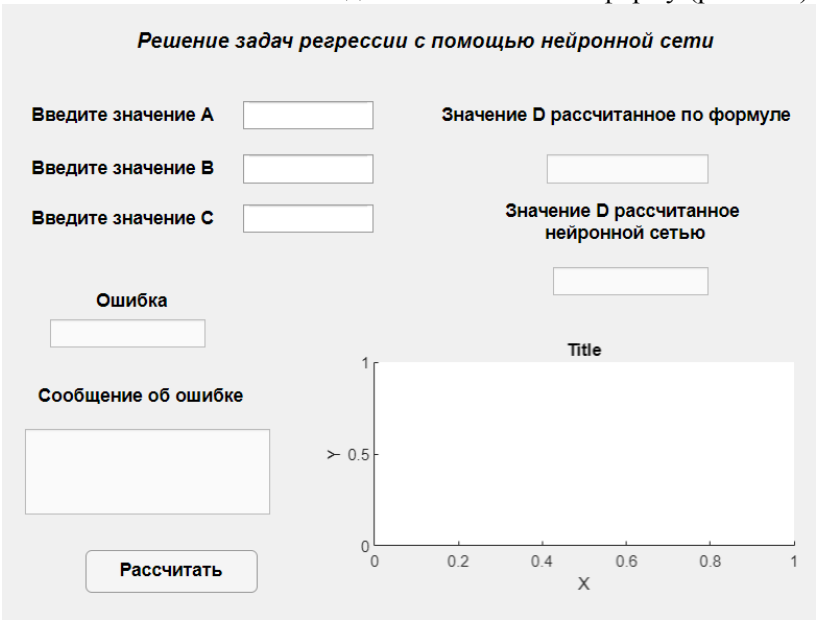


Рис.1.20. Внешний вид формы интерфейса GUI

Приведем описание элементов, которые необходимо разместить на форму.

Label (

) - Элемент, для размещение обычного текста без свойств редактирований. В качестве настроечных параметров может быть использовано положение на форме, размер, тип, цвет используемого шрифта.

EditField (Text) 

- элемент, позволяющий вводить и отображать текст на форме. В качестве настроечных параметров кроме параметров, схожих с элементом Label используются возможность редактирования (галочка Editable) во вкладке interactivity, callback function и прочие параметры.

Button 

- элемент имитирующий нажатие кнопки. Одним из главных настроечных параметров является параметр ButtonPushedCallback, позволяющий добавить действия, которое будет выполнено по нажатию кнопки.

Axes 

- элемент, позволяющий отобразить график на форме.

Для компоновки формы, представленной на рисунке 1.20 используются следующие элементы: Label для заголовка формы. Button для кнопки «Рассчитать», Axes для вывода графика, в остальных случаях EditField (Text) без возможности редактирования за исключением ввода текста в поля A, B и C. Также необходимо обратить внимание на тот факт, что для полей ввода значений и вывода результата в числовой форме предпочтительней использовать поле EditField (Numeric) поэтому при выполнении индивидуального задания необходимо пересмотреть код с учетом использования другого типа поля для ввода и вывода числовых значений.

4. Для корректной работы формы во вторую часть работы в конце скрипта необходимо добавить следующий код:

```
save ('nnobj.mat','net');
```

5. На рисунке 1.21 представлен внешний вид браузера компонентов. Назовите расположенные на форме компоненты в соответствии рис. 1.21.

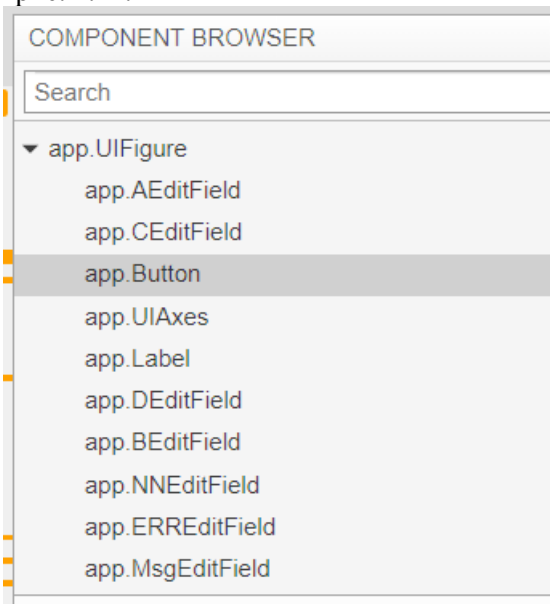


Рис.1.21. Внешний вид браузера компонентов

6. Нажмите правой кнопкой на компонент app.AEditField, затем Calbacks – Add ValueChangedFcn Callback. Послеле ввода данной команды автоматически добавится в поле кода функция, позволяющая фиксировать значение, введённое пользователем в поле A (рис.1.22).

```
% Callback function
function AEditFieldValueChanged(app, event)
    value = app.AEditField.Value;

end
```

Рис. 1.22. Фрагмент кода

Повторите те же самые действия для компонентов app.BEditField и app.CEditField.

7. Нажмите правой кнопкой на компонент app.Button, затем Calbacks – Add ButtonPushedFcn Callback. В коде добавьте функцию, которая будет выполняться при нажатии кнопки:

calcD (app);

8. Добавляем в область кода сразу за объявлением переменных следующий код:

```
  properties (Access = public)



  end
  methods

    function calcD (app)
    %Считываем значение из ячеек A, B и C и переводим их в
%формат numeric
        A=str2num(convertCharsToStrings(app.AEditField.Value));
        B=str2num(convertCharsToStrings(app.BEditField.Value));
        C=str2num(convertCharsToStrings(app.CEditField.Value));
        %вычисляем значение функции по формуле
        D= A*A+2*B+3*C;
        %переводим результат в формат string
        D1=num2str(D);
        %отображаем результат в поле DEditField
        app.DEditField.Value=D1;
        %Загружаем нейронную сеть, сохраненную как объект в
%файле
        %nnobj.mat. Файл обязательно помещаем в тот же каталог
        nn=load ('nnobj.mat');
        %формируем входной массив данных
        xvh(1,1)=A;
        xvh(2,1)=B;
        xvh(3,1)=C;
        %получаем результат на выходе нейронной сети
        test_prim_1=sim(nn.net,xvh);
        %переводим результат в формат string
        D2=num2str(test_prim_1);
```

```
        %отображаем результат в поле NNEditField
        app.NNEditField.Value=D2;
        %Вычисляем ошибку работы сети
        Err=(D-test_prim_1)/D;
        Err=abs(Err);
        %переводим результат в формат string
        Err2=num2str(Err);
        %формироруем сообщение о качестве работы сети и выводим
%его в поле MsgEditField
        Errmsg1="Требуется дообучение сети";
        Errmsg2="Дообучение сети не требуется";
        app.ERREditField.Value=Err2;
        if Err>=0.05
           app.MsgEditField.Value=Errmsg1;
        else
           app.MsgEditField.Value=Errmsg2;
        end
        %формируем график и отображаем его в поле UIAxes
        s1= scatter(app.UIAxes, A, test_prim_1,'o');
        hold (app.UIAxes,'on')
        scatter(app.UIAxes, B, test_prim_1,'o');
        hold (app.UIAxes,'on')
        scatter(app.UIAxes, C, test_prim_1,'o');
        hold (app.UIAxes,'on')
        scatter(app.UIAxes, A, D,'x');
        hold (app.UIAxes,'on')
        scatter(app.UIAxes, B, D,'x');
        hold (app.UIAxes,'on')
        scatter(app.UIAxes, C, D,'x');
        hold (app.UIAxes,'off')
        %параметры масштабирования графика
        xAlimin=A-A*0.6;
        xAlimmax=A+A*0.6;
        yDlimin=D-D*0.6;
        yDlimmax=D+D*0.6;
        ytprimlimin=test_prim_1- test_prim_1*0.6;
```

```
        ytprimlimmax= test_prim_1+D+test_prim_1*0.6;
        ylimmin=min(yDlimin,ytprimlimin);
        ylimmax=min(yDlimmax,ytprimlimmax);
        %подписываем график
        title(app.UIAxes,'Сравнение обучающей и тестовой выборок')
        xlabel(app.UIAxes, 'a')
        ylabel(app.UIAxes, 'd')
        %изменяем масштаб графика
        xlim(app.UIAxes, [xAlimin xAlimmax])
        ylim(app.UIAxes, [ylimmin ylimmax])
        end
    end
```

9. Запускаем форму на исполнение (нажимаем кнопку Run)

10. Тестируем работоспособность формы и представленных алгоритмов. На рисунке 1.23 представлена форма в рабочем состоянии.

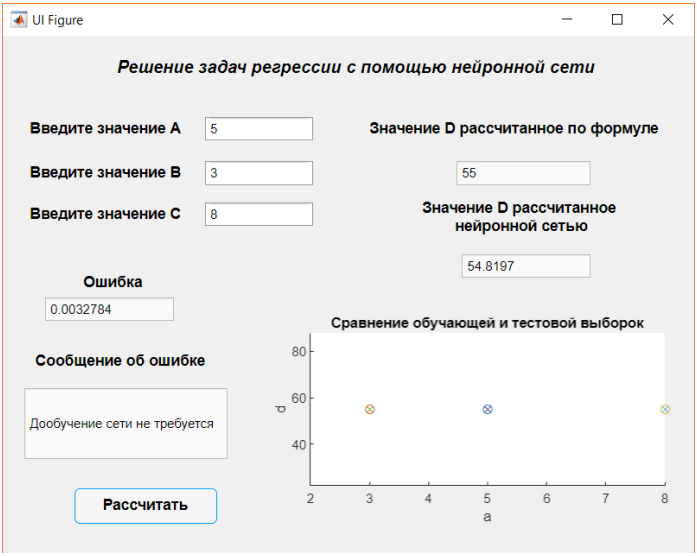


Рис. 1.23. Пользовательский интерфейс в рабочем состоянии

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Ознакомьтесь с основными теоретическими сведениями

2. Создайте нейронную сеть (согласно заданию) с помощью специального инструмента Neural Network toolbox в программной среде Matlab. Используйте представленный в методических указаниях алгоритм действий.

3. Создайте нейронную сеть в ПО Matlab без использования специального пакета инструментов для создания нейронной сети. Используйте представленный в методических указаниях алгоритм действий.

4. Выпишите дополнительно и изучите самостоятельно вопросы создания нейронной сети, которые не освящены в теоретических сведениях.

5. Создайте пользовательский интерфейс для работы с программой.

6. С помощью пользовательского еще раз протестируйте работу нейронной сети. Проанализируйте значение ошибки.

7. Самостоятельно разработайте программный код и элементы пользовательского интерфейса, позволяющего проводить дообучение нейронной сети.

8. Сделайте выводы о проделанной работе.

Контрольные вопросы и задания

1. Что такое искусственная нейронная сеть? Из каких элементов она состоит

2. Что такое нейрон? Укажите основные его особенности

3. Что такое функция активации? Для чего используется функция активации

4. Назовите правила для формирования тестовой и обучающей выборки

5. Укажите основные команды и принципы создания искусственной нейронной сети в Matlab

ЗАДАНИЕ

Создайте нейронную сеть, решающую задачу регрессии, при условии, что зависимость между входными и выходными значениями задана уравнением: $f(x)=a_1a^3+2*b_1*b^2+3*c_1c+d_1d$. При

этом значение a_1,b_1,c_1 и d_1 выберете согласно вариантам (Таблица 1.2).

Таблица 1.2

Исходные данные для выполнения работы

Номер варианта	Значения коэффициентов			
	a_1	b_1	c_1	d_1
1	2,2	-0,8	3,6	-0,1
2	7,3	-0,5	-4,6	-0,7
3	-9,0	6,0	-4,3	2,5
4	3,4	5,6	-3,2	4,4
5	1,3	5,7	-11,8	4,4
6	-5,2	2,7	5,8	-3,5
7	-1,7	-4,8	1,3	0,3
8	1,4	2,9	-3,0	-4,9
9	14,3	6,5	5,5	-4,5
10	11,1	2,0	-6,8	0,0
11	-5,4	4,1	-0,4	6,1
12	12,1	2,9	-1,0	-3,1
13	2,9	-1,2	1,3	1,5
14	-0,3	1,2	1,3	-0,9
15	2,9	-3,1	-3,5	4,5

РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА №2

Использование нейронной сети для решения задачи кластеризации

ЦЕЛЬ РАБОТЫ

Научиться применять теорию нейронной сети для решения задач кластеризации на примере решения задачи деления набора данных на два кластера.

ОБЩИЕ СВЕДЕНИЯ

Решение задачи кластеризации в интеллектуальных системах управления технологическими процессами может быть применен для определения режима работы оборудования выявления дефектов в работе оборудования, а также при решении схожих задач.

Кластеризацией называется объединение объектов в группы на основе близости их свойств. Объекты, существенно отличающиеся друг от друга соотносят к разным кластерам, объекты со схожими свойствами в одинаковые кластеры. Термин кластерный анализ был введен Трионом (Tryon) в 1939 году. Для проведения кластерного анализа используются множество алгоритмов. Кластерный анализ позволяет анализировать показатели данных различных типов, не требует строго формализованных требований к набору данных и их представлению в отличии от задач классификации. Однако важно чтобы анализируемые переменные измерялись в сравнимых шкалах.

Кластерный анализ позволяет сокращать размерность данных, делать ее наглядной. Кластерный анализ может применяться к совокупностям временных рядов, здесь могут выделяться периоды схожести некоторых показателей и определяться группы временных рядов со схожей динамикой.

Задачи кластерного анализа можно объединить в следующие группы:

1. Разработка типологии или классификации.
2. Исследование полезных концептуальных схем группирования объектов.
3. Представление гипотез на основе исследования данных.
4. Проверка гипотез или исследований для определения, действительно ли типы (группы), выделенные тем или иным способом, присутствуют в имеющихся данных.

Порядок выполнения работы при использовании ПО Matlab для решения задачи кластеризации

1. Сформируем два массива для обучения нейронной сети (Рисунок 2.1)

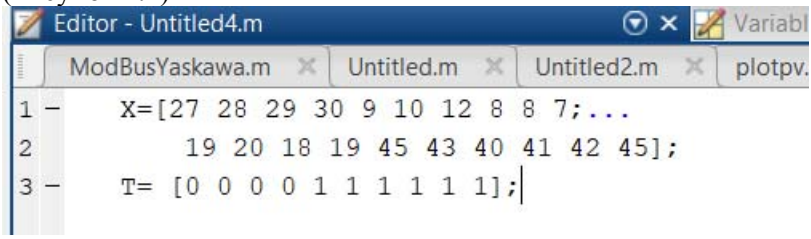


Рис. 2.1. Задание массива значений

2. Запустим скрипт на исполнение и посмотрим массивы, сформированные в рабочей зоне Workspace (Рисунок 2.2)

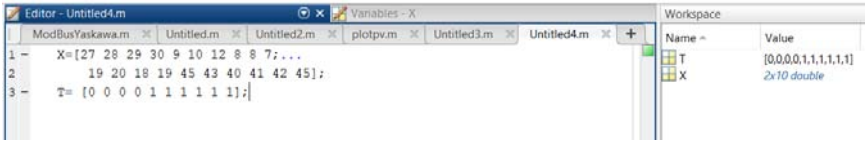


Рис. 2.2. Рабочая зона Workspace

Тут входом X заданы 10 наборов координат точек, подлежащих кластеризации, T – выход – заданные кластеры для данных точек

3. Отобразим данные точки на графике (Рисунок 2.3)

plotpv(X,T);

Функция plotpv специальная функция рисующая график для отображения входов и целевых выходов персептрона

4. Запустим скрипт на исполнение

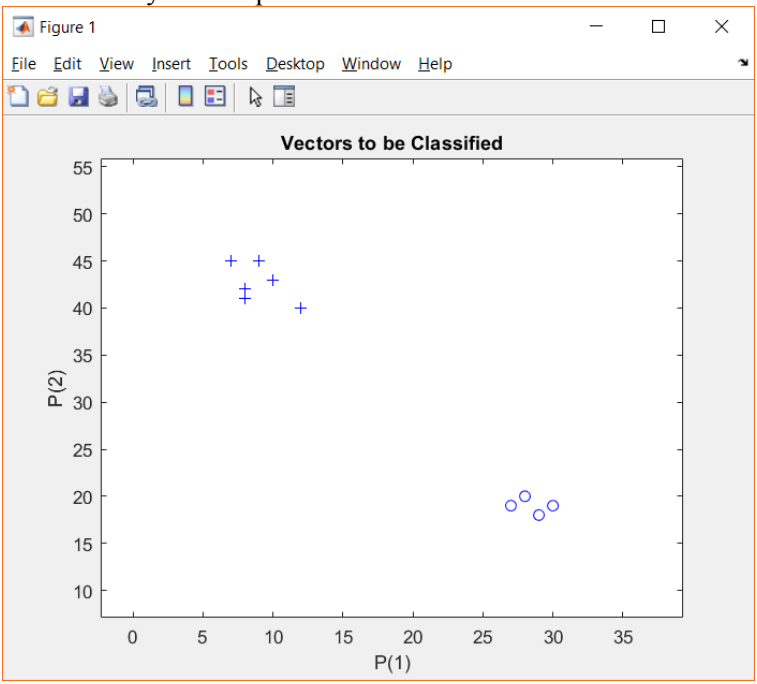


Рис. 2.3. Исходные данные

5. Создадим персептрон с пустыми конфигурационными настройками

```
net = perceptron;
net = configure(net,X,T);
```

6. Запустим скрипт. В рабочей зоне Workspace появится новый объект типа net. Это наша нейронная сеть.

7. Вновь отобразим график

plotpv(X,T);

8. Нарисуем на нем линию, разделяющую 2 кластера, и проходящую через вес и биас персептрона

9. Для проведения анализа дополнительно выведем значения веса и биаса персептрона в рабочую зону Workspace

```
plotpc(net.IW{1},net.b{1});
sw0=net.IW{1};
sb0=net.b{1};
```

10. Запустим скрипт на исполнение

Plotpc – специальная функция, обряжающую линию разделяющую 2 кластера

После запуска скрипта увидим, что линия не отобразилась. В рабочей зоне посмотрим параметры весов и биасов. Увидим значения 0. Значит персептрон не настроен.

11. Зададим команду для настройки персептрона и адаптацию его параметров под решение конкретной задачи

Для этого последовательно в массив поместим значения векторов (repmat – повтор каждого значения в массиве) и запустим функцию адаптации ADAPT

```
XX = repmat(con2seq(X),1,3);
TT = repmat(con2seq(T),1,3);
net = adapt(net,XX,TT);
```

ADAPT обновляет сеть для каждого временного шага в серии и возвращает новый объект, который работает как лучший классификатор.

12. Выведем значение весов и биасов и убедимся что они не нули

sw1=net.IW{1};

sb1=net.b{1};

Теперь можем рисовать линию, разделяющую наши значения на 2 класса (Рисунок 2.4).

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

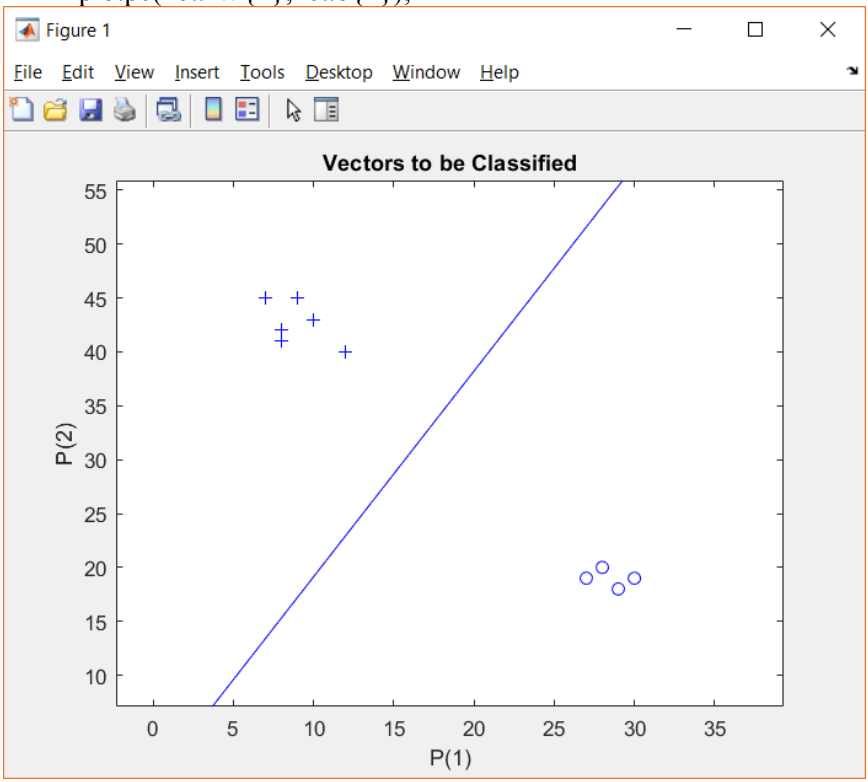


Рисунок 2.4. Линия, разделяющая значения на 2 класса

13. Протестируем работу персептрона для других значений

Зададим новый вектор значений для x

x = [20; 15];

Получаем для данных значений ответ сети

y = net(x);

14. Выводим значения на график, при этом тестовую точку рисуем красным цветом для наглядности. Проверяем результат работы сети (Рисунок 2.5).

```
plotpv(x,y);
point = findobj(gca,'type','line');
point.Color = 'red';
hold on;

plotpv(X,T);
plotpc(net.IW{1},net.b{1});
```

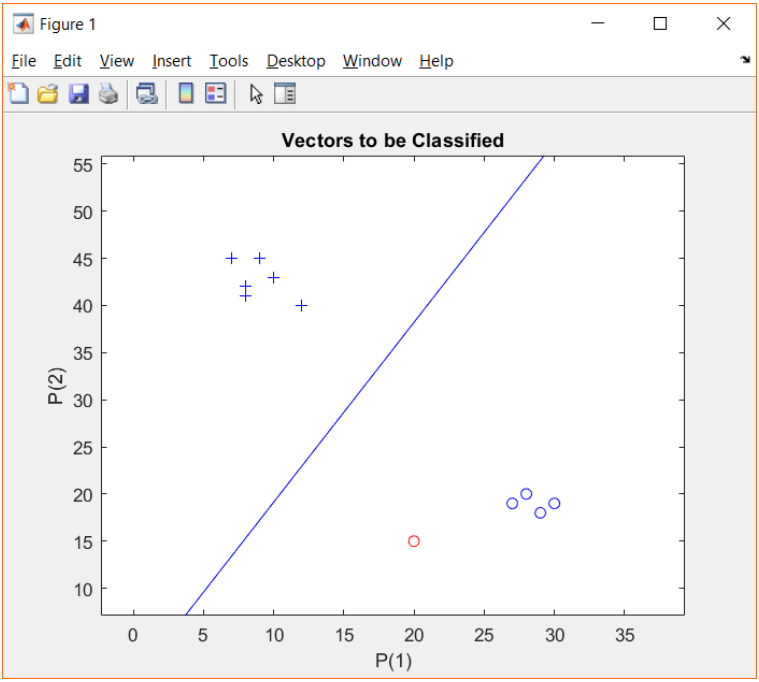


Рис. 2.5. Тестирование работоспособности сети

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Ознакомьтесь с основными теоретическими сведениями
2. Выполните программу, написанную в примере для решения задач кластеризации.

3. Сгенерируйте файл исходных данных для выполнения индивидуального задания.

4. Решите задачу кластеризации по индивидуальному заданию, сначала разделив исходные данные на два кластера, затем на три.

5. Сделайте выводы о проделанной работе.

ЗАДАНИЕ

Создайте нейронную сеть, решающую задачу кластеризации для данных, общие свойства которых указаны в таблице 2.1. Сначала разделите имеющиеся данные по двум признакам, затем по трем. Для формирования файла с исходными данными используйте функцию random. После формирования массива исходных данных сохраните его в Excel и работайте с ним.

Таблица 2.1

Исходные данные для выполнения работы

Номер варианта	Значения коэффициентов		
	1ый кластер	2ой кластер	3ий кластер
1	-15:-5	0:15	45:49
2	-25:-35	-15:-5	10:24
3	-35:-45	-18:-2	8:24
4	98:115	0:24	150:195
5	-16:-7	2:19	29:54
6	-15:-5	-38:-22	45:49
7	-25:-35	0:14	20:34
8	-35:-45	2:19	28:44
9	98:115	0:15	45:49
10	-16:-7	-25:-20	10:24
11	-56:-30	-15:0	10:20
12	-95:-75	-48:-34	-15:0
13	29:39	95:115	135:156
14	-56:-24	-19:0	9:24
15	8:12	36:54	60:86

.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Укажите особенности кластерного анализа
2. Что такое персептрон
3. Укажите основные этапы классификаций массива данных на два признака
4. Укажите основные функции Matlab, используемые для решения задач кластеризации по двум признакам.

РЕКОМЕНДУЕМЫЙ БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Рыбина, Г.В. Основы построения интеллектуальных систем. Учебное пособие [Электронный ресурс] : учеб. пособие — Электрон. дан. — Москва : Финансы и статистика, 2010. — 432 с.
https://e.lanbook.com/book/28363

2. Интеллектуальные автоматизированные системы управления технологическими объектами: Учебно-практическое пособие / Трофимов В.Б., Кулаков С.М. - Вологда:Инфра-Инженерия, 2016. - 232 с.
http://znanium.com/bookread2.php?book=760121

3. Интеллектуальный анализ данных и систем управления бизнес-правилами в телекоммуникациях: Монография / Р.Р. Вейнберг. - М.: НИЦ ИНФРА-М, 2016. - 173 с.
http://znanium.com/bookread2.php?book=520998

4. Автоматизированные нечетко-логические системы управления : монография / С.Г. Емельянов, В.С. Титов, М.В. Бобырь. — М. : ИНФРА-М, 2018. — 175 с.
http://znanium.com/bookread2.php?book=954480

5. Информационная система предприятия: Учебное пособие/Вдовенко Л. А., 2-е изд., перераб. и доп. - М.: Вузовский учебник, НИЦ ИНФРА-М, 2015. - 304 с.
http://znanium.com/bookread2.php?book=501089

6. Нечеткие гибридные системы: Теория и практика / И.З. Батыршин, А.О. Недосекин, А.А. Стецко. - М.: ФИЗМАТЛИТ, 2007. - 208 с.
http://znanium.com/bookread2.php?book=544667

СОДЕРЖАНИЕ