**Определение формы фигур на изображении по их признакам**

В данной лабораторной работе надо будет обучить нейронную сеть находить на изображении квадраты, круги, треугольники, подавая на нейронную сеть вектор признаков изображения.

Выделение признаков, это сама по себе интересная тема. Мы будем выделять 2 признака

1. ‘extent’ – коэффициент заполнения: равен отношению площади объекта к площади ограничивающего прямоугольника. Представляется числом в диапазоне (0,1]. Это будет главный признак, который мы подадим на нейронную сеть.
2. ‘centroid’ – координаты центра масс, для того, чтобы корректно вывести результаты на экран

**КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ О ВЫЧИСЛЕНИИ ПРИЗНАКОВ ИЗОБРАЖЕНИЯ**

При анализе формы областей изображения применяют различные

метрические, топологические и аналитические характеристики: расстояние,

площадь, периметр, число Эйлера, функции кривизны и их производные. Ниже

приведены функции MatLab, позволяющие выполнять выделение нужных

объектов на изображении, поиск связных областей и вычисление

морфометрических признаков: площадей объектов, центров масс объектов,

коэффициентов формы и других метрических и топологических характеристик.

**Функция выделения объектов bwselect**

Синтаксис

BWD = bwselect(BWS ,n)

BWD = bwselect(BWS ,с,r,n)

Функция BW D = bwselect(BW S ,n) выводит изображение BW S на экран и

предоставляет пользователю возможность интерактивно отметить затравочные

пикселы (пиксел, с которого начинается выделение объекта, называется

затравочным). Координаты затравочных пикселов задаются щелчком левой

кнопки мыши. Предыдущий можно удалить клавишей Backspace или Delete.

Последний затравочный пиксел задается двойным щелчком левой кнопки или

однократным щелчком правой кнопки мыши. Нажатие клавиши Enter завершает

процесс выделения затравочных пикселов. Все отмеченные объекты

переносятся сразу после нажатия клавиши Enter, после этого сразу создается

новое бинарное изображение BWD .

Функция BWD = bwselect(BWS ,с,r,n) переносит с изображения BWS на

изображение BWD все объекты c координатами затравочных пикселов из

векторов c и r.

Параметр n для всех рассматриваемых функций bwselect задает критерий

связности. Он может принимать значение 4 или 8 (по умолчанию n = 8).

**Функция поиска объектов в бинарных изображениях bwlabel**

Синтаксис

[L,num] = bwlabel(BW,n)

Функция bwlabel ищет на бинарном изображении BW связные области

пикселов объектов и создает матрицу номеров объектов L. Элементы, имеющие

значение 1, относятся к первому объекту, 2 – ко второму и т. д. Элементы,

имеющие значение 0, относятся к фону. В параметре num возвращается

количество объектов, найденных на изображении BW. Параметр n задает

критерий связности, используемый для нахождения связных областей объектов.

Он может принимать значение 4 или 8 (по умолчанию n = 8).

**Измерение свойств объектов на изображении. Функция вычисления признаков объектов**

Функция **regionprops** применяется для измерения свойств объектов в некоторой окрестности изображения и представляет результат в виде массива. Если применить это к изображению с отмеченными компонентами, то создается структура для каждой компоненты.

В этом примере используется функция **regionprops** для создания структурированного массива, содержащего некоторые основные свойства. Если установить свойствам параметр **'basic'**, тогда функция **regionprops** возвращает три измеренных параметра: площадь, центроид (или центр масс) и ограничивающий прямоугольник (локальная окрестность).

Например:

feats = regionprops(L ,'basic')

Система MATLAB выдает такой результат

feats =

101x1 struct array with fields:

Area

Centroid

BoundingBox

Для поиска области с 51-м отмеченным компонентом, необходимо найти поле Area field и его 51 элемент в структуре массива graindata.

feats(51).Area

В этом случае результат будет таким

ans =

140

Для поиска наименее возможного прямоугольника и центроида (центра масс) некоторых компонентов используется следующий код:

feats(51).BoundingBox, feats(51).Centroid

ans =

107.5000 4.5000 13.0000 20.0000

ans =

114.5000 15.4500

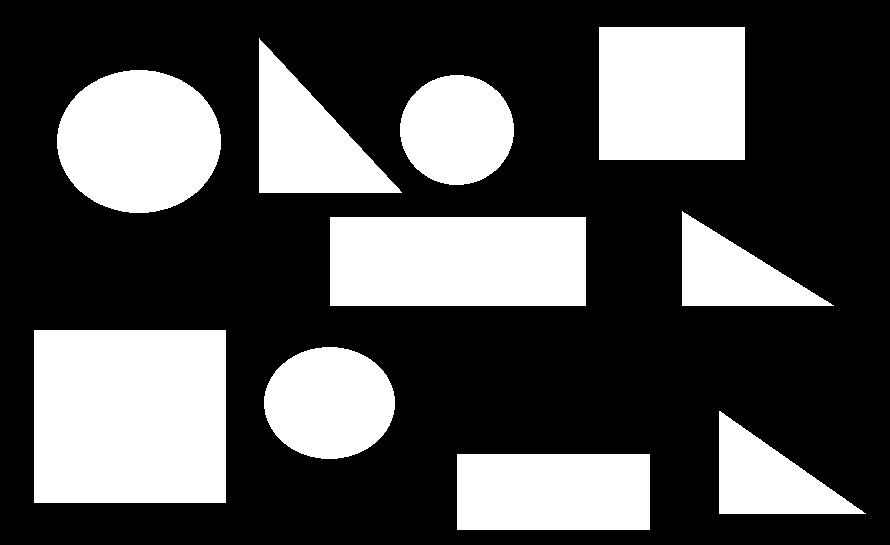
Можно задавать другие признаки

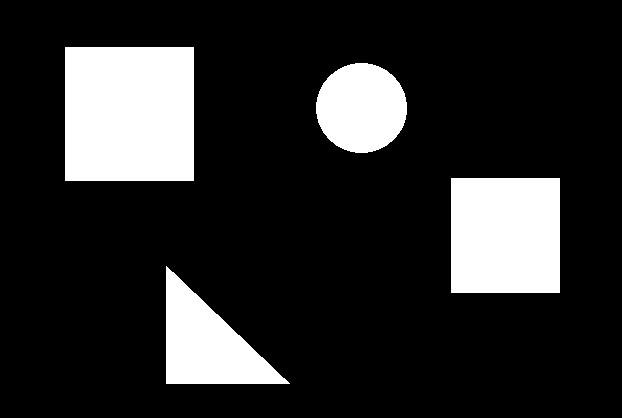
**feats =regionprops(L, 'Extent', 'Centroid','Image');**

Вот всем этим мы и будем пользоваться.

**ПОДГОТОВКА ИЗОБРАЖЕНИЯ**

Для начала заготовьте 2 изображения – для обучения сети и для проверки ее работы. Хорошо, если первое будет содержать больше объектов, чем второе. Я подготовила такие.

****

****

Итак, у нас на обучающем изображении есть 4 вида объектов 3 круга, 2 квадрата, 3 треугольника и 2 прямоугольника и у всех у них будет разное значении признака “Extent”.

Количество объектов на вашем изображении написано в варианте. Перейдем к вычислению признаков.

**ВЫЧИСЛЕНИЕ ПРИЗНАКОВ ОБЪЕКТОВ**

Загружаем первое изображение в Matlab. Помним, что оно должно храниться в папке Документы-Матлаб текущего пользователя, чтобы не прописывать путь до файла.

f=imread('PR3.bmp');

>> figure, imshow(f);

Признаки вычисляются только у полутонового или бинарного изображения, поэтому либо сохраняйте его в графическом редакторе, как черно-белое, либо сделайте полутоновым так: f=rgb2gray(f);

Теперь надо выделить объекты. Если мы не видим изображения, то пользуемся функцией

**[L num]=bwlabel(f,8);**

Но я предпочитаю пользоваться функцией **BW = bwselect(f ,8)**

В этои случае левой кнопкой мыши нужно отметить объекты, а затем нажать правую кнопку. После этого дать команду

**[L,num] = bwlabel(BW,8);**

Далее приступаем к вычислению признаков объектов, которые отмечены в матрице номеров объектов L. Значения признаков возвращаются в массиве структур feats. Как было отмечено ранее, при распознавании объектов могут использоваться любые наборы признаков, но мы используем только 2.

**feats= regionprops (L,'Extent','Centroid');**

сформируем матрицы этих признаков

**Extent=zeros(num);**

**CentX=zeros(num);**

**CentY=zeros(num);**

**for i=1:1:num;**

**Extent(i)=feats(i).Extent;**

**CentX(i)=feats(i).Centroid(1);**

**CentY(i)=feats(i).Centroid(2);**

**end;**

Посмотрим наш признак формы:

>>Extent

Extent =

1.0000 0 0 0 0 0 0 0 0 0

0.7848 0 0 0 0 0 0 0 0 0

0.5030 0 0 0 0 0 0 0 0 0

0.7846 0 0 0 0 0 0 0 0 0

1.0000 0 0 0 0 0 0 0 0 0

0.7846 0 0 0 0 0 0 0 0 0

1.0000 0 0 0 0 0 0 0 0 0

1.0000 0 0 0 0 0 0 0 0 0

0.5024 0 0 0 0 0 0 0 0 0

0.5052 0 0 0 0 0 0 0 0 0

Видим, что для разных фигур он дает разное значение

1.0000 - для квадрата

0.7848 – для круга

0.5030 – для треугольника

С прямоугольником сложно. В данном случае у меня такие прямоугольники, что у них коэффициент формы такой же, как и у квадрата, ну и пусть будут квадратами

Таким образом у нас появилось 3 класса фигур

1 класс - 1.0000 квадрат

2 класс - 0.7848 круг

3 класс - 0.5030 треугольник

Вот вам и задачка для нейронной сети

**ГОТОВИМ ДАННЫЕ ДЛЯ НЕЙРОННОЙ СЕТИ**

Создадим входной и выходной вектора

Входным вектором будет наше значение Extent, но только не в столбик, а в строчку

x=Extent(:,1)'

x =

1.0000 0.7848 0.5030 0.7846 1.0000 0.7846 1.0000 1.0000 0.5024 0.5052

Выходной делаем ручками, ориентируясь на классы, выделенные выше. Надо сказать, что можно взять самоорганизующуюся сеть и она вам сама классы выделит. Я вот решила попроще – возьму LVQ сеть и буду ее учить.

Итак, выходной вектор:

y=[1 2 3 2 1 2 1 1 3 3]

**РАЗМЕТКА ОБУЧАЮЩЕГО ИЗОБРАЖЕНИЯ**

Чтобы убедиться, что мы все сделали правильно, разметим обучающее изображение.

for i=1:num;

f(round(CentY(i)):round(CentY(i))+1,round(CentX(i)):round(CentX(i))+1)=0;

if y (i)==1;

text(round(CentX(i)),(round(CentY(i))), ' Квадрат ')' ;

end;

if y(i)==2;

text(round(CentX(i)),(round(CentY(i))), ' Круг ')'

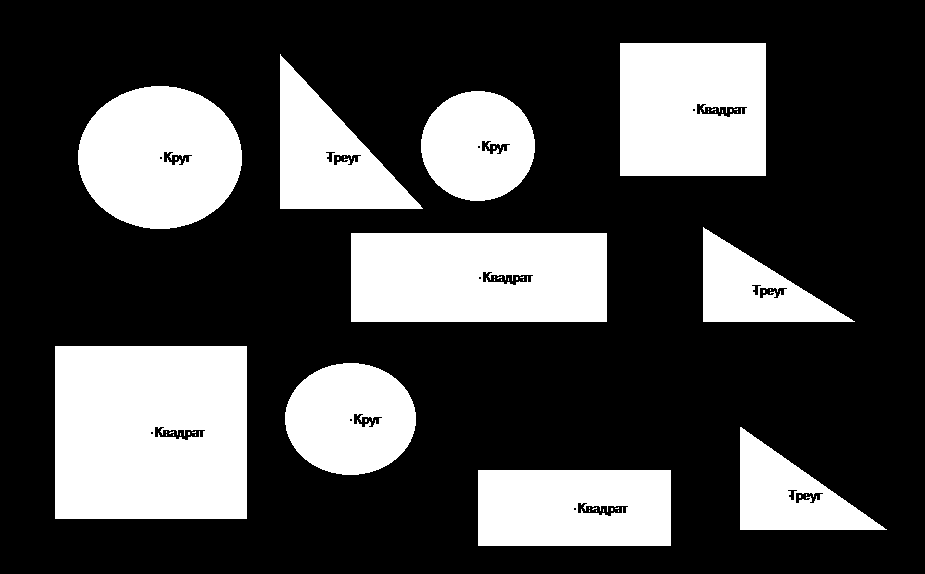
end;

if y(i)==3;

text(round(CentX(i)),(round(CentY(i))), 'Треуг')'

end;

end;

 Вот что получили – значит все правильно.

**СОЗДАНИЕ И ОБУЧЕНИЕ НЕЙРОННОЙ СЕТИ**

Создадим матрицу связности для наших выходов

T = ind2vec(y);

T = full(T);

>> T

T =

1 0 0 0 1 0 1 1 0 0

0 1 0 1 0 1 0 0 0 0

0 0 1 0 0 0 0 0 1 1

Процентные доли входных векторов в каждом классе равны 0.4, 0.3 и 0.3 соответственно. Кластеризовать нам, вроде ничего не надо, но давайте выделим 4 кластера, (помним, что кластеров должно быть больше, чем классов), все равно нам надо указать количество нейронов в скрытом слое. Создаем сеть:

**net = newlvq(minmax(x),4,[.4 .3 .3],0.1);**

Проводим обучение:

**net = train(net,x,T);**

Проверяем на обучающем векторе:

>> ac = vec2ind(sim(net,x))

ac =

1 2 3 2 1 2 1 1 3 3

Вроде все нормально

**ОПРЕДЕЛЕНИЕ ФОРМЫ ФИГУР НА ТЕСТОВОМ ИЗОБРАЖЕНИИ**

Теперь берем другое изображение и опять готовим последовательности для входного и выходного векторов

f1=imread('PR2.bmp');

>> figure, imshow(f1);

**BW1 = bwselect(f1 ,8)**

**[L1,num1] = bwlabel(BW1,8);**

**feats= regionprops (L1,'Extent','Centroid');**

**Extent1=zeros(num1);**

**CentX1=zeros(num1);**

**CentY1=zeros(num1);**

**for i=1:1:num1;**

**Extent1(i)=feats(i).Extent;**

**CentX1(i)=feats(i).Centroid(1);**

**CentY1(i)=feats(i).Centroid(2);**

**end;**

x1=Extent1(:,1)'

И подадим уже новые входные данные на обученную сеть

ac1 = vec2ind(sim(net,x1))

ac1 =

1 2 3 2

Визуализируем

for i=1:num1;

f1(round(CentY1(i)):round(CentY1(i))+1,round(CentX1(i)):round(CentX1(i))+1)=0;

if ac1(i)==1;

text(round(CentX1(i)),(round(CentY1(i))), ' Квадрат ')' ;

end;

if ac1(i)==2;

text(round(CentX1(i)),(round(CentY1(i))), ' Круг ')'

end;

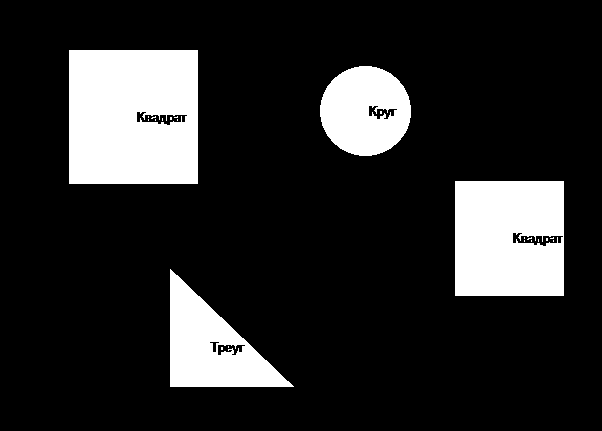
if ac1(i)==3;

text(round(CentX1(i)),(round(CentY1(i))), 'Треуг')'

end;

end;

Получилось:



Сеть распознала фигуры.

Если не получается с визуализацией помните, что у вас есть признаки с предыдущего изображения, поэтому прежде чем, загружать новое удалите эти переменные из памяти, чтобы не мешались.