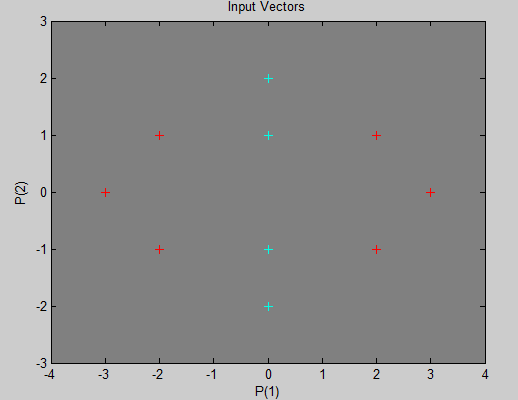
**Подготовка данных и создание LVQ нейронной сети**

Зададим обучающую последовательность в следующем виде:

P = [-3 -2 -2 0 0 0 0 2 2 3; 0 1 -1 2 1 -1 -2 1 -1 0];

C = [1 1 1 2 2 2 2 1 1 1];

Построим картину распределения входных векторов по классам

T = ind2vec(C);

colormap(hsv);

plotvec(P,C)

title('Input Vectors');

xlabel('P(1)');

ylabel('P(2)');

hold on

Векторы, относящиеся к разным классам, отмечены разными символами. Как следует из расположения векторов, классы не являются линейно отделимыми, и задача не может быть решена с использованием, например, персептрона.

Массив целевых векторов T будет выглядеть так:

>> T = full(T)

T =

1 1 1 0 0 0 0 1 1 1

0 0 0 1 1 1 1 0 0 0

Процентные доли входных векторов в каждом классе равны 0.6 и 0.4 соответственно. Из картинки видно, что можно брать 4 нейрона конкурирующего слоя ( 4 кластера). Параметр скорости настройки - 0.1. Теперь подготовлены все данные, необходимые для вызова функции newlvq. Вызов может быть реализован c использованием следующего оператора:

**net = newlvq(minmax(P),4,[.6 .4],****0.1)**

Внимательно рассмотрим архитектуру сети

net =

Neural Network object:

architecture:

numInputs: 1

numLayers: 2

biasConnect: [0; 0]

inputConnect: [1; 0]

layerConnect: [0 0; 1 0]

outputConnect: [0 1]

numOutputs: 1 (read-only)

numInputDelays: 0 (read-only)

numLayerDelays: 0 (read-only)

и архитектуру слоев

**net.layers{1}**

ans =

dimensions: 4

distanceFcn: ''

distances: []

initFcn: 'initwb'

name: 'Layer'

netInputFcn: 'netsum'

netInputParam: [1x1 struct]

positions: [0 1 2 3]

size: 4

topologyFcn: 'hextop'

transferFcn: 'compet'

transferParam: [1x1 struct]

userdata: [1x1 struct]

Посмотрим начальные веса

**net.layers{2}**

ans =

dimensions: 2

distanceFcn: ''

distances: []

initFcn: 'initwb'

name: 'Layer'

netInputFcn: 'netsum'

netInputParam: [1x1 struct]

positions: [0 1]

size: 2

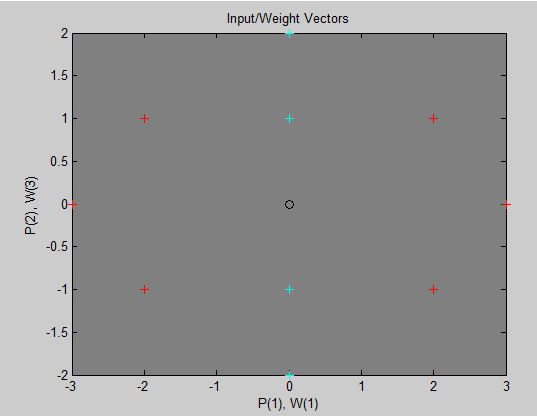
topologyFcn: 'hextop'

transferFcn: 'purelin'

transferParam: [1x1 struct]

userdata: [1x1 struct]

**net.inputWeights{1}**

ans =

delays: 0

initFcn: 'midpoint'

learn: 1

learnFcn: 'learnlv1'

learnParam: [1x1 struct]

size: [4 2]

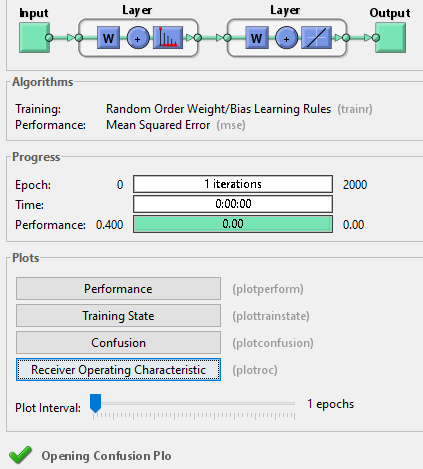
userdata: [1x1 struct]

weightFcn: 'negdist'

weightParam: [1x1 struct]

Построим эти веса на графике

W1 = net.IW{1};

plot(W1(1,1),W1(1,2),'ok')

title('Input/Weight Vectors');

xlabel('P(1), W(1)');

ylabel('P(2), W(3)');

Зададим параметры обучения

net.trainParam.epochs=150;

net.trainParam.show=Inf;

И проведем обучение

net = train(net,P,T);

В результате обучения получим следующие весовые коэффициенты нейронов конкурирующего слоя, которые определяют положения центров кластеризации:

V = net.IW{1,1}

V =

-1.3116 0.0550

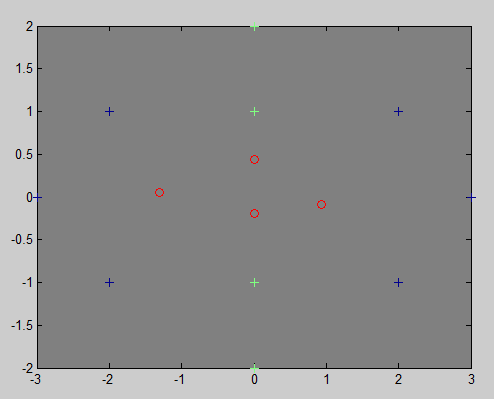
0.9367 -0.0905

0 0.4420

0 -0.1900

Теперь проведем кластеризацию на графике

cla;

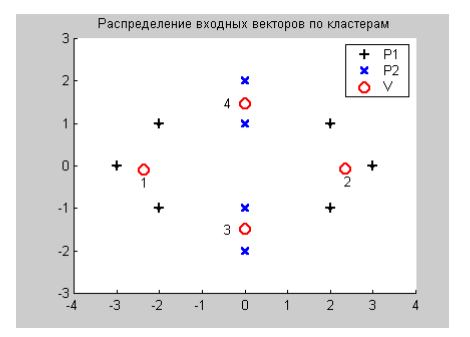
plotvec(P,C);

hold on;

plot(V(:,1),V(:,2),'or')

Видим на графике 4 центра кластеризации

Здесь центры кластеризации разъехались недостаточно. Нужно подобрать другое количество эпох обучения и параметр обучения. Поэкспериментируйте.



Должно получиться так:

Выход второго слоя

net.LW{2}

ans =

1 1 0 0

0 0 1 1

Определим к какому классу отнести этот вектор

>> p = [0.2; 1];

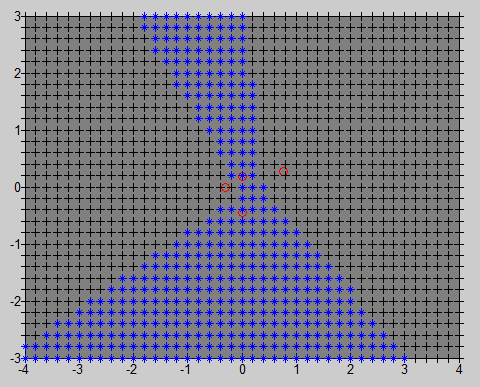
a = vec2ind(sim(net,p))

a =

2 Он относится ко второму классу и к верхнему кластеру ( видим на графике)

Теперь построим области, разделяющие классы и здесь же будет видно кластерное разделение. Для этого покроем сеткой прямоугольную область и определим принадлежность каждой точки к тому или иному классу

x = -4:0.2:4; y = -3:0.2:3;

[X,Y] = meshgrid(x,y); (прямоугольник)

P = cat(3,X,Y); (объединение массивов по

Размерности 3)

[n1,n2,n3] = size(P);

P = permute(P,[3 2 1]); (перестановка размерностей массива)

P = reshape(P, [n3 n1\*n2]); (преобразование для сети)

Y = sim(net,P);

(Симуляция на всем прямоугольнике обученной сети)

Yc = vec2ind(Y); (обратное преобразование в двумерную матрицу)

I1 = find(Yc==1); (присвоение значений классов)

I2 = find(Yc==2);

>> plot(P(1,I1),P(2,I1),'+k')

hold on

plot(P(1,I2),P(2,I2),'\*b')

Анализ рисунка подтверждает, что граница между областями не является прямой линией

**Решение задачи классификации и кластеризации**

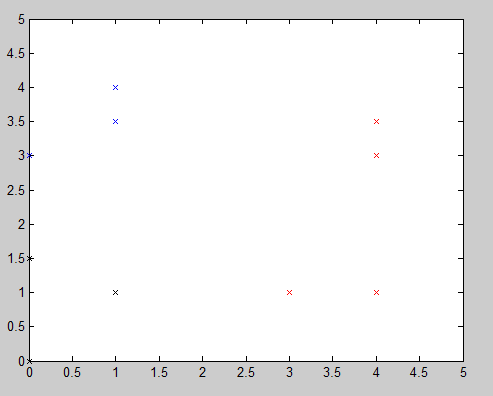
Разбить на 3 класса точки пространства так, чтобы точка [1, 3] принадлежала 2 классу, [0,1] принадлежала 1 классу и [5,2] принадлежала 3 классу.

Возьмем по нескольку точек, которые будут находиться в окрестности данных точек и, следовательно принадлежать к выделенным классам, например так:

[0 0 ; 1 1; 0 1.5; 1 3.5 ; 0 3;1 4;3 1;4 1;4 3; 4 3.5]

На 3 класса: [1 1 1 2 2 2 3 3 3 3]

P= [0 0;1 1; 0 1.5; 1 3.5 ; 0 3;1 4;3 1;4 1;4 3; 4 3.5]'; % Транспонируем матрицу входных векторов

C = [1 1 1 2 2 2 3 3 3 3]; % Создадим матрицу выходов

T = ind2vec(C); % Создадим матрицу связности

T = full(T);

Визуализируем входные данные:

I1 = find(C==1);

I2 = find(C==2);

I3 = find(C==3);

axis([0,5,0,5]), hold on

P1 = P(:,I1);

P2 = P(:,I2);

P3 = P(:,I3);

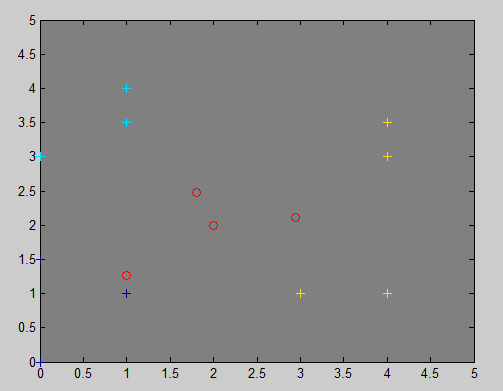
plot(P1(1,:),P1(2,:),'xk');

plot(P2(1,:),P2(2,:),'xb');

plot(P3(1,:),P3(2,:),'xr');

Процентные доли входных векторов в каждом классе равны 0.3, 0.3 и 0.4 соответственно. Из картинки видно, что можно брать 4 нейрона конкурирующего слоя ( 4 кластера). Создаем сеть:

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

Проводим обучение:

net.trainParam.epochs=250;

net.trainParam.show=Inf;

net = train(net,P,T);

положения центров кластеризации:

V = net.IW{1,1}

cla;

plotvec(P,C);

hold on;

plot(V(:,1),V(:,2),'or')

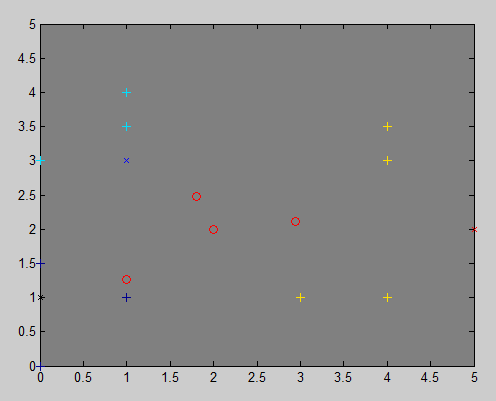
Классифицируем наши заданные векторы

a= [0 1; 1 3; 5 2]';

ac = vec2ind(sim(net,a))

ac =

1 2 3

Нанесем их на график

hold on

I1 = find(ac==1);

I2 = find(ac==2);

I3 = find(ac==3);

a1 = a(:,I1);

a2 = a(:,I2);

a3 = a(:,I3);

plot(a1(1,:),a1(2,:),'xk');

plot(a2(1,:),a2(2,:),'xb');

plot(a3(1,:),a3(2,:),'xr');

построим области, разделяющие классы

x = 0:0.2:5; y = 0:0.2:5;

[X,Y] = meshgrid(x,y);

P = cat(3,X,Y);

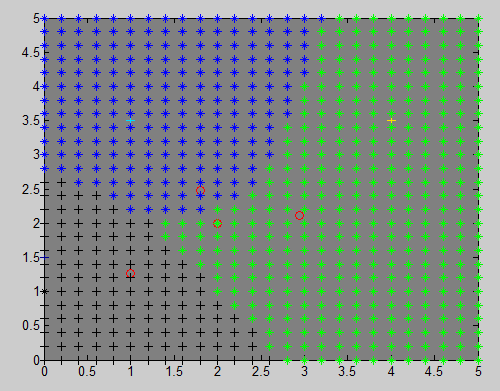
[n1,n2,n3] = size(P);

P = permute(P,[3 2 1]);

P = reshape(P, [n3 n1\*n2]);

Y = sim(net,P);

Yc = vec2ind(Y);

I1 = find(Yc==1);

I2 = find(Yc==2);

I3 = find(Yc==3);

hold on

plot(P(1,I1),P(2,I1),'+k');

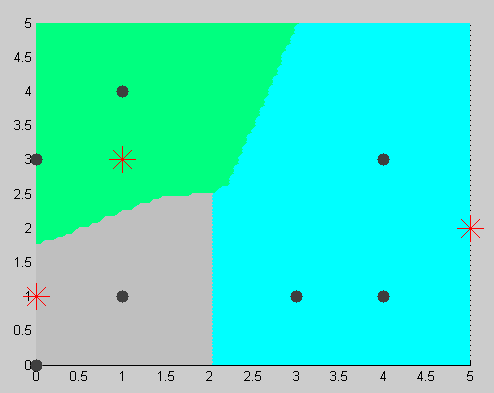
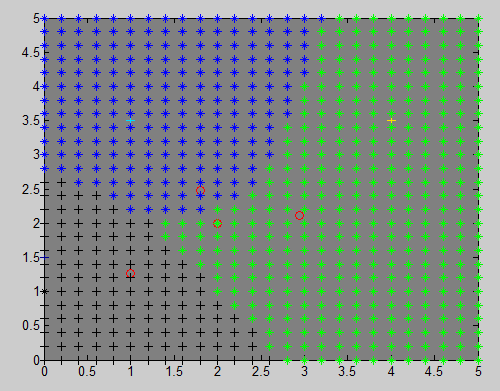
hold on

plot(P(1,I2),P(2,I2),'\*b');

hold on

plot(P(1,I3),P(2,I3),'\*g');

Сравним результаты с классификацией из лабораторной № 5 радиально-базисной сетью PNN



Видим, что результаты похожи, но не идентичны. Впрочем, если проделать классификацию обеими сетями еще раз, то результаты будут немного отличаться тоже. На то она и нечеткая логика. Преимущество LVQ сети в том, что мы можем видеть и центры кластеризации.