3. Исследование нейронных сетей с радиально-базисными функциями (РБФ)

Цель работы:

- приобретение навыков построения, инициализации и обучения РФБ-НС
- исследование РФБ-НС при решении задач аппроксимации статических зависимостей и классификации

Теоретические сведения

РБФ-НС – общие сведения

НС с радиально-базисными функциями (РБФ) содержат в своем составе специальные РБФ-нейроны. Рассмотрим структуру РБФ-нейрона. Она отличается от структуры нейронов в обычных НС ПР.

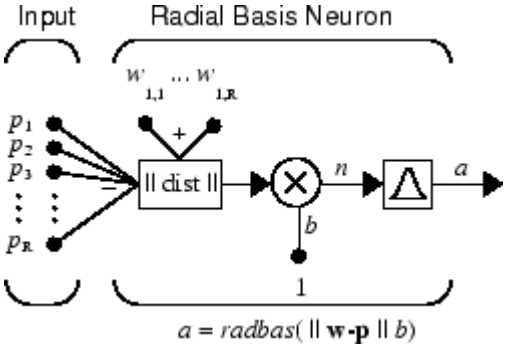

Рис. Структура РБФ-нейрона

Во-первых, активация нейрона – это не сумма входных сигналов, взвешенных коэффициентами, а расстояние между вектором входных сигналов и вектором весовых коэффициентов

$$n = \|P - W\| * b$$

Во-вторых, передаточная функция нейрона является гауссианом:

$$a = radbas(n) = \exp(-n^2) = \exp\left(\frac{-\|P-W\|^2}{(1/b)^2}\right)$$

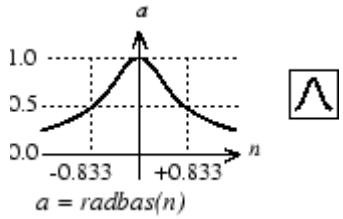

Рис. График радиально-базисной функции

Эта функция имеет максимум, равный 1, при n = 0 и плавно убывает при увеличении n, достигая значения 0.5 при n = ±0.833. То есть, если входной вектор P полностью совпадает с вектором весовых коэффициентов, то на выходе РБФ-нейрона будет 1. Если же расстояние между P и W, равняется 0.833/b, то на выходе нейрона будет 0.5. С дальнейшим увеличением расстояния между P и W выходное значение a стремится к 0. В связи с этим вектор весовых коэффициентов часто называется **центром** РБФ-нейрона.

Из выражения для a видно, что b влияет на скорость убывания функции выхода a от расстояния $\|P-W\|$. С ростом b функция убывает быстрее, с уменьшением b она убывает медленнее.

Базовая РБФ-НС

Базовая РБФ-НС состоит из двух слоев. Первый слой представлен РБФ-нейронами, второй слой – линейный.

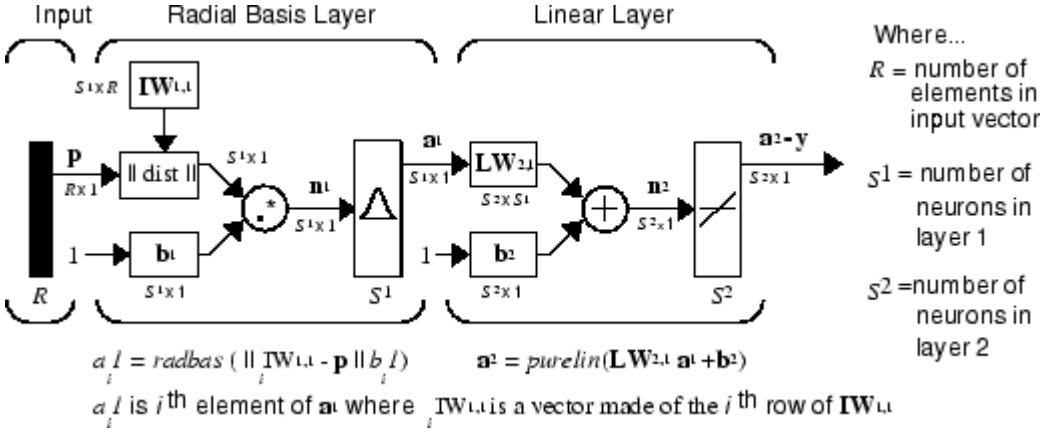

Рис. Структура базовой РБФ-НС

РБФ-нейроны первого слоя играют роль детекторов значений входного сигнала. Нейроны второго, линейного слоя проецируют на выход сигналы от РБФ-нейронов.

Синтез РБФ-НС для решения задачи аппроксимации

За счет своей организации РБФ-НС могут быть легко использованы для решения задачи аппроксимации непрерывных функций. Для этого необходимо в РБФ-слой записать наиболее характерные значения входного аргумента, а линейный слой настроить так, чтобы суммарная ошибка была минимальной.

В случае если имеется обучающая выборка, возможны два варианта:

1. Точный синтез РБФ-НС. Для каждого примера (Pi,Ti) создается РБФ-нейрон с центром, соответствующим входному вектору примера Pi. Затем для каждого примера Pi вычисляется отклик всех РБФ-нейронов $A^{(i)} = [A_1^{(i)} A_2^{(i)} ... A_{S1}^{(i)}]$. Для вычисления коэффициентов линейного слоя решается система уравнений

$[A^{(1)}; A^{(2)}; ...; A^{(n)}]W = [T^{(1)}; T^{(2)}; ...; T^{(S2)}]$.

2. Приближенный синтез РБФ-НС. Задается максимально допустимая ошибка аппроксимации. РБФ-нейрон создается не для каждого примера, а лишь для части. В результате система уравнений для нахождения коэффициентов линейного слоя получается неполной и появляется ненулевая ошибка. Если эта ошибка меньше допустимой, то РБФ-НС считается сформированной. В противном случае увеличивается число РБФ-нейронов.

В случае приближенного синтеза достаточно "остро" встает проблема выбора набора примеров, составляющих центры РБФ-нейронов. В простейшем случае можно выбрать примеры как-то случайно. Более сложные алгоритмы предполагают предварительную ***кластеризацию*** множества входных данных с заданием в качестве числа кластеров числа РБФ-нейронов. В результате кластеризации формируются центры РБФ-нейронов.

Выбор параметра b – ширины гауссиана

В обоих случаях помимо обучающей выборки необходимо задать параметр b, определяющий скорость убывания функции РБФ-нейронов от расстояния. Очевидно, что он зависит от типа аппроксимируемой функции. В точках, где функция изменяется быстро, b нужно задавать больше, где медленно - меньше. Если параметр b задан неверно, то аппроксимация будет неудовлетворительной

- если функция медленно изменяющаяся, а РБФ-нейроны быстрые, то в точках, соответствующих обучающим примерам, аппроксимация будет хорошей, а в точках между обучающими примерами аппроксимация будет очень плохой;

- если функция быстро изменяющаяся, а РБФ-нейроны медленные, то, скорее всего, система уравнений для нахождения W будет иметь неполный ранг и аппроксимация будет плохой во всех точках.

Таким образом, для настройки РБФ-НС необходимо правильно подобрать следующие параметры:

- количество РБФ-нейронов
- примеры, входные значения которых будут являться центрами РБФ-нейронов
- скорость РБФ-нейронов (скорость убывания функции выхода от расстояния).

При этом критерием выбора этих параметров является ошибка аппроксимации на тестовой выборке.

В отличие от НСПР с сигмоидальными нейронами РБФ-НС требуют для аппроксимации большего числа нейронов, но при этом их обучение занимает гораздо меньшее время. Это объясняется тем, что сигмоидальные функции изменяются в большем диапазоне входных значений, чем радиально-базисные функции.

РБФ-НС с обобщенной регрессией (GRNN)

Помимо стандартных РБФ-НС были предложены т.н. РБФ-НС с обобщенной регрессией (Generalized Regression NN, GRNN), отличающиеся выходным слоем. Этот слой для вычисления активации использует не просто скалярное произведение - V=W*P, а скалярное произведение с нормализацией V=W*P/sum(P). GRNN предназначены для аппроксимации функций.

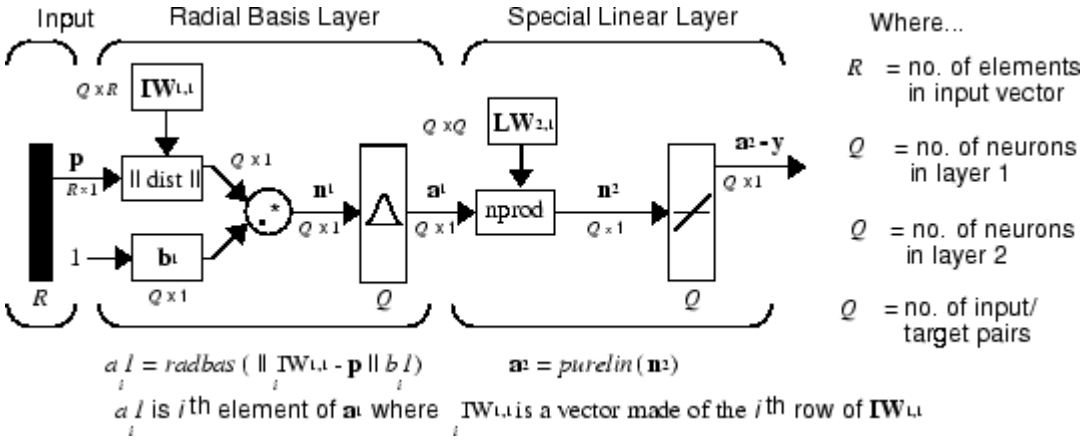

Рис. Структура GRNN

Весовые коэффициенты от i РБФ-нейрона к выходному слою задаются равными Ti - выходным сигналам i примера. В результате, при подаче на вход НС сигнала P, близкого к Pi – входному сигнала i примера, соответствующий i РБФ-нейрон выдаст значение, близкое к 1, а остальные РБФ-нейроны в идеале выдадут значения, близкие к нулю. В соответствии с весовыми коэффициентами на выходе РБФ-НС будет значение, близкое к Ti. Если же входной сигнал P находится примерное посередине между Pi и Pj - P=(Pi+Pj)/2, то на выходе НС будет значение, близкое к (Ti+Tj)/2 (при условии правильного задания ширины гауссианов).

Вероятностные НС (PNN)

Классические РБФ-НС используются для аппроксимации непрерывных функций. Это происходит из-за того, что выходной слой в них линейный. Если же сделать этот слой соревновательным, то РБФ-НС смогут решать задачу классификации. Соответствующие НС называются еще вероятностными НС (Probability Neural Network). Название происходит от того, что соревновательная передаточная функция применяется к набору значений, физический смысл которых – вероятность того, что входной сигнал принадлежит соответствующему классу. В результате выбирается тот класс, к которому входной образ подходит с максимальной вероятностью.

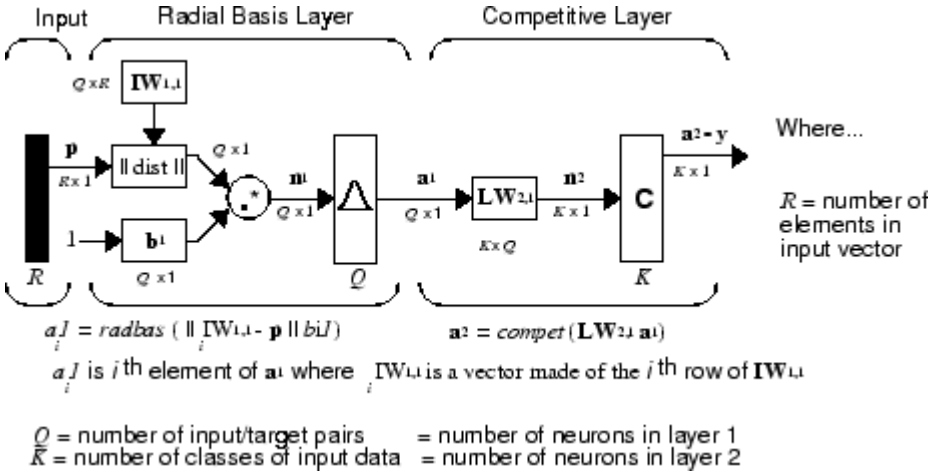

Рис. Структура PNN

РБФ-нейроны в PNN настраиваются также, как и во всех РБФ-НС – центры РБФ-нейронов соответствуют векторам входных значений примеров.

Весовые коэффициенты от i РБФ-нейрона к нейронам соревновательного слоя по аналогии с GRNN задаются равными Ti - желаемым выходным сигналам i-го примера. В данном случае Ti содержит информацию о том, к какому классу принадлежит i-й пример (если i-й пример относится к j классу, то Ti(j)=1, Ti(k)=0, k≠j).

Недостаток PNN – если на входе будет образ, не похожий ни на один из классов, то на выходе все равно будет один из классов.

Построение РБФ-НС в среде Matlab

Формирование на уровне нейронов

Рассмотрим вначале, формирование РБФ-НС на уровне нейронов, а затем перейдем к формированию на уровне НС.

РБФ-нейрон отличается от обычных нейронов алгоритмом вычисления активации и передаточной функцией. Поэтому в Matlab для нейронов РБФ-слоя необходимо задать следующие параметры (предполагается, что РБФ-нейроны располагаются в первом слое):

- вычисление расстояния между входным вектором и вектором весовых коэффициентов: net.inputWeights{1,1}.weightFcn = 'dist';
- умножение расстояния на смещение: net.layers{1}.netInputFcn = 'netprod';
- применение РБФ: net.layers{1}.transferFcn = 'radbas'.

Обобщая, можно записать, что выход a{i} i-го РБФ-нейрона в Matlab вычисляется по формуле

a{i} = radbas(netprod(dist(net.IW{1,1}(i,:), p), net.b{1}))

Выходной слой стандартной РБФ-НС является линейным, поэтому нейроны этого слоя имеют следующие параметры, определяющие алгоритм их работы:

- net.layerWeights{2,1}.weightFcn = 'dotprod';
- net.layers{2}.netInputFcn = 'netsum';
- net.layers{2}.transferFcn = 'purelin'.

В РБФ-НС с обобщенной регрессией (GRNN) выходной слоя является модифицированным линейным (активация вычисляется, как скалярное произведение с нормализацией) поэтому нейроны от нейронов линейного слоя отличаются в следующем:

- net.layerWeights{2,1}.weightFcn = 'normprod'

Наконец, в PNN выходной слоя является соревновательным, поэтому нейроны от нейронов линейного слоя отличаются в следующем:

- net.layers{2}.transferFcn = 'compet'

Формирование на уровне НС (задачи)

В Matlab имеются несколько встроенных функций, которые значительно упрощают процедуру создания и формирования РБФ-НС: newrb, newrbe, newgrnn, newpnn. Рассмотрим каждую из этих функций отдельно.

Функция newrbe - создание точной РБФ-НС (по всему набору примеров)

```
net = newrbe(P,T,spread)
```

P – набор желаемых входных значений примеров (матрица RxQ)

T – набор желаемых входных значений примеров (матрица SxQ)

spread – параметр, управляющий шириной гауссианов РБФ-нейронов

Смещение всех элементов в слое выбирается равным b = 0.8326/spread. С увеличением значения spread гауссоида расширяется, с уменьшением – сужается. Если расстояние между входным вектором и центром РБФ-нейрона равно 0, то на выходе нейрона будет 1, если расстояние равно spread, то на выходе РБФ-нейрона будет 0.5. То есть spread определяет ширину гауссиан слоя РБФ-нейронов.

Функция newrb - создание приближенной РБФ-НС (с заданной ошибкой)

```
net = newrb(P,T,goal,spread,MN,DF)
```

Параметры P, T, spread имеют тот же смысл, что и newrbe

goal – максимально допустимая ошибка

MN – максимальное количество нейронов (по умолчанию Q – число примеров)

DF – через какое количество добавляемых нейронов производить визуализацию

Функция newgrnn - создание GRNN (по алгоритму) для аппроксимации

```
net = newgrnn(P,T,spread)
```

Параметры P, T, spread имеют тот же смысл, что и в функции newrbe.

Функция newpnn - создание PNN (по алгоритму) для классификации

```
net = newpnn(P,T,spread)
```

Параметры P, T, spread имеют тот же смысл, что и в функции newrbe.

Примеры

Пример 1 – РБФ –нейронные сети для аппроксимации

1. Постройте график радиально-базисной функции

```
x = -3:.1:3;
a = radbas(x);
plot(x,a)
title('Radial Basis Transfer Function');
xlabel('Input p');
ylabel('Output a');
```

2. Постройте композицию трех РБФ в разных точках и с различными весами

```
a2 = radbas(x-1.5);
a3 = radbas(x+2);
a4 = a + a2*1 + a3*0.5;
plot(x,a,'b-',x,a2,'b--',x,a3,'b--',x,a4,'m-')
title('Weighted Sum of Radial Basis Transfer Functions');
xlabel('Input p');
ylabel('Output a');
```

3. Сформируйте аппроксимируемую зависимость в виде набора точек и постройте ее.

```
X = -1:.1:1;
T = [-.9602 -.5770 -.0729 .3771 .6405 .6600 .4609 ...
 .1336 -.2013 -.4344 -.5000 -.3930 -.1647 .0988 ...
 .3072 .3960 .3449 .1816 -.0312 -.2189 -.3201];
plot(X,T,'+');
title('Training Vectors');
xlabel('Input Vector P');
ylabel('Target Vector T');
```

4. Для сформированной зависимости создайте и обучите РБФ-НС с допустимой ошибкой 0.02 и шириной РБФ, равной 1.

```
eg = 0.02; % sum-squared error goal
sc = 1; % spread constant
net = newrb(X,T,eg,sc);
```

5. Постройте график функции, реализуемой РБФ-НС

```
plot(X,T,'+');
xlabel('Input');

X = -1:.01:1;
Y = net(X);

hold on;
plot(X,Y);
hold off;
legend({'Target','Output'})
```

6. Задайте ширину РБФ spread равной 0.01, создайте РБФ-НС, обучите ее и постройте график функции, реализуемой РБФ-НС. Сравните результаты.

7. Задайте ширину РБФ spread равной 100, создайте РБФ-НС, обучите ее и постройте график функции, реализуемой РБФ-НС. Сравните результаты.

Пример 2 – Нейронная сеть обобщенной регрессии (GRNN)

1. Сформируйте аппроксимируемую зависимость и постройте ее.

```
X = [1 2 3 4 5 6 7 8];
T = [0 1 2 3 2 1 2 1];

plot(X,T,'.','markersize',30)
axis([0 9 -1 4])
title('Function to approximate.')
xlabel('X')
```

```
ylabel('T')
```

2. Сформируйте GRNN с шириной РБФ, равной 0.7, обучите ее на исходную зависимость и постройте результаты аппроксимации РБФ-НС в точках обучающей выборки.

```
spread = 0.7;
net = newgrnn(X,T,spread);
A = net(X);
hold on
outputline = plot(X,A,'.','markersize',30,'color',[1 0 0]);
title('Create and test y network.')
xlabel('X')
ylabel('T and A')
```

3. Постройте результат аппроксимации в точке x=3.5

```
x = 3.5;
y = net(x);
plot(x,y,'.','markersize',30,'color',[1 0 0]);
title('New input value.')
xlabel('X and x')
ylabel('T and y')
```

4. Постройте плавный график кривой, реализуемой GRNN

```
X2 = 0:.1:9;
Y2 = net(X2);
plot(X2,Y2,'linewidth',4,'color',[1 0 0])
title('Function to approximate.')
xlabel('X and X2')
ylabel('T and Y2')
```

Пример 3 – Вероятностная нейронная сеть (PNN)

1. Сформируйте три 2-мерные точки, каждая из которых принадлежит отдельному классу и постройте их на графике.

```
X = [1 2; 2 2; 1 1]';
Tc = [1 2 3];
plot(X(1,:),X(2,:),'.','markersize',30)
for i=1:3, text(X(1,i)+0.1,X(2,i),sprintf('class %g',Tc(i))), end
axis([0 3 0 3])
title('Three vectors and their classes.')
xlabel('X(1,:)')
ylabel('X(2,:)')
```

2. Создайте PNN для классификации трех точек

```
T = ind2vec(Tc);
spread = 1;
net = newpnn(X,T,spread);
```

3. Постройте на графике результаты классификации (точка и класс)

```
Y = net(X);
Yc = vec2ind(Y);
plot(X(1,:),X(2,:),'.','markersize',30)
axis([0 3 0 3])
for i=1:3,text(X(1,i)+0.1,X(2,i),sprintf('class %g',Yc(i))),end
title('Testing the network.')
xlabel('X(1,:)')
ylabel('X(2,:)')
```

4. Произведите классификацию новой точки x = [2; 1.5]

```
x = [2; 1.5];
y = net(x);
ac = vec2ind(y);
hold on
plot(x(1),x(2),'.','markersize',30,'color',[1 0 0])
text(x(1)+0.1,x(2),sprintf('class %g',ac))
hold off
```

```
title('Classifying y new vector.')
xlabel('X(1,:) and x(1)')
ylabel('X(2,:) and x(2)')
```

5. Постройте диаграмму Вороного, демонстрирующую разбиение плоскости на классы, реализуемое PNN

```
x1 = 0:.05:3;
x2 = x1;
[X1,X2] = meshgrid(x1,x2);
xx = [X1(:) X2(:)]';
yy = net(xx);
yy = full(yy);
m = mesh(X1,X2,reshape(yy(1,:),length(x1),length(x2)));
set(m,'facecolor',[0 0.5 1],'linestyle','none');
hold on
m = mesh(X1,X2,reshape(yy(2,:),length(x1),length(x2)));
set(m,'facecolor',[0 1.0 0.5],'linestyle','none');
m = mesh(X1,X2,reshape(yy(3,:),length(x1),length(x2)));
set(m,'facecolor',[0.5 0 1],'linestyle','none');
plot3(X(1,:),X(2,:),[1 1 1]+0.1,'.','markersize',30)
plot3(x(1),x(2),1.1,'.','markersize',30,'color',[1 0 0])
hold off
view(2)
title('The three classes.')
xlabel('X(1,:) and x(1)')
ylabel('X(2,:) and x(2)')
```

Практические задания

Задание 1 – Аппроксимация с помощью базовой РБФ-НС и GRNN

1. С помощью точной РБФ-НС (newrbe) аппроксимируйте функцию (для вашего варианта из заданий с номером 5).

а. *Задайте* достаточное для точной аппроксимации количество обучающих примеров (в данном случае совпадающее с числом РБФ-нейронов).

б. Изменяя ширину РБФ (spread), *определите наилучшее значение* этого параметра с точки зрения качества аппроксимации. *Постройте три графика аппроксимации* для разных значений spread: оптимального, больше и меньше оптимального, когда явно видно, что аппроксимация плохая. При построении графиков аппроксимации не забывайте приводить графики исходной желаемой зависимости.

в. *Постройте график* зависимости ошибки аппроксимации от параметра spread в логарифмическом масштабе.

2. Произведите аппроксимацию с помощью, приближенной РБФ-НС (newrb).

а. *Используйте* найденное в п.1 оптимальное значение spread и достаточный для точной аппроксимации объем обучающей выборки.

б. Изменяя значение допустимой ошибки (goal), *постройте зависимость* числа используемых РБФ-нейронов от допустимой ошибки аппроксимации. Для промежуточных результатов постройте *графики аппроксимации*.

Зам. Для более точного нахождения количества нейронов, при которых достигается заданная цель (значение ошибки), количество нейронов, прибавляемое на каждом шаге (5-й параметр newrb, DF), следует задавать не очень большим.

Зам. Максимальное количество нейронов (4-й параметр функции newrb, MN) по построению РБФ-НС не может превышать объема обучающей выборки.

3. Выполните аппроксимацию с помощью GRNN.

а. *Задайте* вначале достаточный для точной аппроксимации объем обучающей выборки.

б. По аналогии с newrb изменяя ширину РБФ (spread), *определите наилучшее значение* этого параметра с точки зрения качества аппроксимации. *Постройте график* аппроксимации на исходной зависимости.

в. Уменьшите объем обучающей выборки в несколько раз (рассмотрите 3 случая) и *подберите оптимальные значения* параметра spread для каждого случая. *Постройте полученные графики* аппроксимации. Приведите значения ошибок.

3. Сравните качество аппроксимации сетями прямого распространения и различными вариантами РБФ-НС (точной, приближенной РБФ-НС и GRNN) по различным показателям:

- число нейронов, требуемое для достижения заданного качества аппроксимации;
- время обучения;
- сложность настройки (выбора параметров) НС.

Постройте на одном графике зависимости ошибки от числа нейронов для различных типов НС.

Зам. При расчете ошибок для различных типов НС следует использовать одни и те же формулы.

Задание 2 – Классификация с помощью PNN (2 класса)

1. *Задайте* достаточное для точной классификации количество обучающих примеров.

2. *Подберите оптимальное* значение spread в смысле минимальной ошибки на тестовой выборке. *Визуализируйте* результаты классификации (диаграмма соотнесения тестовых примеров с классами, раскраска плоскости). *Приведите значение* средней ошибки.

3. *Постройте дополнительно графики классификации* для значений spread, больших и меньших оптимального, когда явно видно, что классификация неудовлетворительная.

4. Уменьшите объем обучающей выборки в несколько раз (рассмотрите 3 случая) и *подберите оптимальные значения* параметра spread для каждого случая. *Визуализируйте* результаты классификации и *приведите* значение средней ошибки.

5. *Постройте поверхность* ошибки в плоскости двух параметров: ширина РБФ-функции spread и объем обучающей выборки.

6. *Сравните* полученные результаты с НС прямого распространения по аналогии с п. 3 задания 1.

Задание 3 – Классификация с помощью PNN (>2 классов)

С помощью PNN *произведите классификацию* линейно неразделимых образов (для вашего варианта из заданий с номером 4) *по аналогии* с заданием 2.

Для п. 2 и 3 дополнительно *приведите* матрицы неточностей и *рассчитайте* ошибки первого и второго родов.

Задание 4 – Классификация многомерных образов с помощью PNN

Попробуйте решить задачу классификации многомерных образов с помощью PNN.

1. *Сформируйте* обучающую выборку *достаточного* объема.

2. *Подберите оптимальное* значение spread.

3. *Исследуйте* качество классификации на тестовой выборке, содержащей зашумленные примеры. Приведите матрицу неточностей, рассчитайте среднюю ошибку и ошибки 1,2 родов.

4. *Попробуйте увеличить и уменьшить* объем обучающей выборки в несколько раз. Для каждого случая подберите оптимальное значение spread и *рассчитайте* показатели качества классификации.

5. *Проанализируйте* полученные результаты, *сравнив* их с результатами классификации тех же самых образов НС прямого распространения *по аналогии* с п. 3 задания 1.