**模式识别实验报告**

第三次实验

电信钱51

杨晨烨

2150300135

**实验一**

Explore some of the properties of density estimation in the following way.

1. **代码：**

x=unifrnd(-1/2,1/2,1,10000);

y=unifrnd(-1/2,1/2,1,10000);

z=unifrnd(-1/2,1/2,1,10000);

plot3(x,y,z,'r.')

**结果：**



1. **代码：**

x=unifrnd(-1/2,1/2,1,10000);

y=unifrnd(-1/2,1/2,1,10000);

z=unifrnd(-1/2,1/2,1,10000);

n=10000;

p=[];

t=[];

h=0:0.001:1;

for h=0:0.001:1

m=0;

for i=1:10000

if (abs(x(i))<(1/2\*h))&(abs(y(i))<(1/2\*h))&(abs(z(i))<(1/2\*h))

m=m+1;

end

end

t=[t;h];

k=m/(n\*(h^3));

p=[p;k];

end

plot(t,p)

xlabel('h')

ylabel('概率密度p')

**结果：**



1. **代码：**

k=1:10000;

n=10000;

x=[];

y=[];

for k=1:10000

x=[x;k];

p=(1/n)/(1/sqrt(k));

y=[y;p];

end

plot(x,y)

xlabel('n')

ylabel('概率密度p')

**结果：**



1. **代码：**

N=10000

ang1=rand(1,N)\*2\*pi;% 随机10000个0~2pi高斯分布的角度1

ang2=acos(rand(1,N)\*2-1);% 随机10000个-1~1高斯分布的反余弦获得角度2

r=rand(1,N).^(1/3);% 随机10000个0~1高斯分布数的开立方为到原点距离

x=r.\*cos(ang1).\*sin(ang2);% x

y=r.\*sin(ang1).\*sin(ang2);% y

z=r.\*cos(ang2);% z

figure(1)

plot3(x,y,z,'r.');

grid on;

axis square;

n=10000;

p=[];

t=[];

h=0:0.001:1;

for h=0:0.001:1

m=0;

for i=1:10000

if (abs(x(i))<(1/2\*h))&(abs(y(i))<(1/2\*h))&(abs(z(i))<(1/2\*h))

m=m+1;

end

end

t=[t;h];

k=m/(n\*(h^3));

p=[p;k];

end

figure(2)

plot(t,p)

xlabel('h')

ylabel('概率密度p')

k=1:10000;

n=10000;

x=[];

y=[];

for k=1:10000

x=[x;k];

p=1/n/(1/sqrt(k));

y=[y;p];

end

figure(3)

plot(x,y)

xlabel('n')

ylabel('概率密度p')

**结果：**







**实验二**

Consider Parzen-window estimates and classifiers for points in the table above. Let your window function be a spherical Gaussian.

1. **代码：**

function [ p ] = Parzen( w,x,h )

%UNTITLED3 Summary of this function goes here

% Detailed explanation goes here

[xt,yt,zt] = size(w);

p = zeros(1,zt);

for i = 1:zt

hn = h;

for j = 1:xt

hn = hn / sqrt(j);

p(i) = p(i) + exp(-(x - w(j,:,i))\*(x - w(j,:,i))'/ (2 \* power(hn,2))) / (hn \* sqrt(2\*3.14));

end

p(i) = p(i) / xt;

end

end

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

w1(:,:,1) = [ 0.28 1.31 -6.2;...

0.07 0.58 -0.78;...

1.54 2.01 -1.63;...

-0.44 1.18 -4.32;...

-0.81 0.21 5.73;...

1.52 3.16 2.77;...

2.20 2.42 -0.19;...

0.91 1.94 6.21;...

0.65 1.93 4.38;...

-0.26 0.82 -0.96];

w1(:,:,2) = [0.011 1.03 -0.21;...

1.27 1.28 0.08;...

0.13 3.12 0.16;...

-0.21 1.23 -0.11;...

-2.18 1.39 -0.19;...

0.34 1.96 -0.16;...

-1.38 0.94 0.45;...

-0.12 0.82 0.17;...

-1.44 2.31 0.14;...

0.26 1.94 0.08];

w1(:,:,3) = [ 1.36 2.17 0.14;...

1.41 1.45 -0.38;...

1.22 0.99 0.69;...

2.46 2.19 1.31;...

0.68 0.79 0.87;...

2.51 3.22 1.35;...

0.60 2.44 0.92;...

0.64 0.13 0.97;...

0.85 0.58 0.99;...

0.66 0.51 0.88];

x=zeros(3,3);

x(1,:) = [0.5 1 0];

x(2,:) = [0.31 1.51 -0.5];

x(3,:) = [-0.3 0.44 -0.1];

h =1; %重要参数

p = Parzen(w1,x(1,:),h);

num = find(p == max(p));

disp(['点：[',num2str(x(1,:)),']判为三类概率分别为：',num2str(p)]);

disp(['点：[',num2str(x(1,:)),']判为第',num2str(num),'类']);

p = Parzen(w1,x(2,:),h);

num = find(p == max(p));

disp(['点：[',num2str(x(2,:)),']判为三类概率分别为：',num2str(p)]);

disp(['点：[',num2str(x(2,:)),']判为第',num2str(num),'类']);

p = Parzen(w1,x(3,:),h);

num = find(p == max(p));

disp(['点：[',num2str(x(3,:)),']判为三类概率分别为：',num2str(p)]);

disp(['点：[',num2str(x(3,:)),']判为第',num2str(num),'类']);

**结果：**

点：[0.5 1 0] 判为三类概率分别为：0.021399 0.063487 0.036143

点：[0.5 1 0]判为第2类

点：[0.31 1.51 -0.5]判为三类概率分别为：0.020753 0.04831 0.031649

点：[0.31 1.51 -0.5]判为第2类

点：[-0.3 0.44 -0.1]判为三类概率分别为：0.030391 0.034147 0.0032054

点：[-0.3 0.44 -0.1]判为第2类

1. **代码：**

h = 0.1;

**结果：**

点：[0.5 1 0]判为三类概率分别为：4.3451e-43 2.7003e-07 2.4543e-47

点：[0.5 1 0]判为第2类

点：[0.31 1.51 -0.5]判为三类概率分别为：1.9187e-44 6.7674e-10 2.0289e-43

点：[0.31 1.51 -0.5]判为第2类

点：[-0.3 0.44 -0.1]判为三类概率分别为：7.4653e-28 4.7765e-11 3.3355e-127

点：[-0.3 0.44 -0.1]判为第2类

**实验三**

Consider k-nearest-neighbor density estimations in different numbers of dimensions.

1. **代码：**

function [ px ] = kneighbor( a, kn, x )

%UNTITLED4 Summary of this function goes here

% Detailed explanation goes here

[m, n] = size(a);

b = x;

N = 100;

if n == 1

px = zeros(N,1);

vn1 = zeros(N,1);

for i = 1:N

for j = 1:m

vn1(j) = abs(b(i) - a(j));

end

vn1 = sort(vn1);

px(i) = kn / N / (vn1(kn));

end

end

if n == 2

px = zeros(N,1);

vn2 = zeros(N,1);

for i = 1:N

for j = 1:m

vn2(j) = sqrt((b(i,1)-a(j,1))^2+(b(i,2)-a(j,2))^2);

end

vn2 = sort(vn2);

px(i) = kn/N/(vn2(kn));

end

end

if n == 3

px = zeros(3,1);

vn3 = zeros(3,1);

for i = 1:3

for j = 1:m

vn3(j) = sqrt((b(i,1)-a(j,1))^2+(b(i,2)-a(j,2))^2+(b(i,3)-a(j,3))^2);

end

vn3 = sort(vn3);

px(i) = kn/3/(vn3(kn));

end

end

end

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

w1= [ 0.28 1.31 -6.2;...

0.07 0.58 -0.78;...

1.54 2.01 -1.63;...

-0.44 1.18 -4.32;...

-0.81 0.21 5.73;...

1.52 3.16 2.77;...

2.20 2.42 -0.19;...

0.91 1.94 6.21;...

0.65 1.93 4.38;...

-0.26 0.82 -0.96];

w2 = [0.011 1.03 -0.21;...

1.27 1.28 0.08;...

0.13 3.12 0.16;...

-0.21 1.23 -0.11;...

-2.18 1.39 -0.19;...

0.34 1.96 -0.16;...

-1.38 0.94 0.45;...

-0.12 0.82 0.17;...

-1.44 2.31 0.14;...

0.26 1.94 0.08];

w3 = [ 1.36 2.17 0.14;...

1.41 1.45 -0.38;...

1.22 0.99 0.69;...

2.46 2.19 1.31;...

0.68 0.79 0.87;...

2.51 3.22 1.35;...

0.60 2.44 0.92;...

0.64 0.13 0.97;...

0.85 0.58 0.99;...

0.66 0.51 0.88];

a1=w3(:,1);

N=100;

b=2\*rand(N,1);

k1=1;k2=3;k3=5;

p1=kneighbor(a1,k1,b);

p2=kneighbor(a1,k2,b);

p3=kneighbor(a1,k3,b);

figure(1);

subplot(1,4,1);

plot(b,p1,'.');

subplot(1,4,2);

plot(b,p2,'.');

subplot(1,4,3);

plot(b,p3,'.');

**结果：**



1. **代码：**

b1=[];

a2=w2(:,1);

a3=w2(:,2);

b1=[a2 a3];

N=100;

b2=3\*rand(N,2);

k1=1;k2=3;k3=5;

p11=kneighbor(b1,k1,b2);

p12=kneighbor(b1,k2,b2);

p13=kneighbor(b1,k3,b2);

data1=[b2 p11];

figure(2);

plot3(b2(:,1),b2(:,2),p11,'.');

grid on;

figure(3);

plot3(b2(:,1),b2(:,2),p12,'.');

grid on;

figure(4);

plot3(b2(:,1),b2(:,2),p13,'.');

grid on;

**结果：**







1. **代码：**

b3=[-0.41,0.82,0.88;0.14,0.72,4.1;-0.81,0.61,-0.38];

k=1;

p21=kneighbor(w1,k,b3);

p22=kneighbor(w2,k,b3);

p23=kneighbor(w3,k,b3);

disp([' w1的概率密度分别为： ',num2str(p21(1)), ' ', num2str(p21(2)), ' ' , num2str(p21(3))]);

disp([' w2的概率密度分别为： ',num2str(p22(1)), ' ', num2str(p22(2)), ' ' , num2str(p22(3))]);

disp([' w3的概率密度分别为： ',num2str(p23(1)), ' ', num2str(p23(2)), ' ', num2str(p23(3))]);

**结果：**

w1的概率密度分别为： 0.19107 0.24827 0.40334

w2的概率密度分别为： 0.43463 0.084605 0.36871

w3的概率密度分别为： 0.30568 0.10439 0.17194

**实验四**

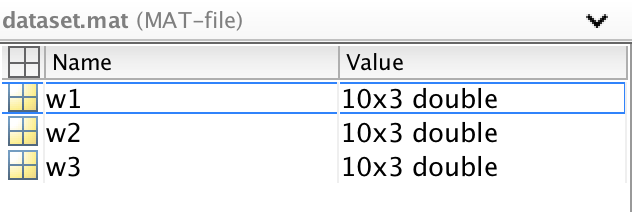
Write a program to create a Voronoi tessellation in two dimensions as follows.

function [x4,y4]=find\_symmetry\_p(x1,y1,x2,y2,x3,y3)

x4=(2\*(y1-y2)\*(x2-x3)+2\*x2\*(y2-y3)+((x2-x3)^2-(y2-y3))\*x1)/((y2-y3)+(x2-x3)^2);

y4=-((x2-x3)/(y2-y3))\*(x4-x1)+y1;

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~



~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

clear;clc;

load('dataset.mat');

x=[w1(:,1);w3(:,1)]; %将w1和w3的x1坐标设为一个变量

y=[w1(:,2);w3(:,2)]; %将w1和w3的x2坐标设为一个变量

N=10;k=4;

1. **结果：**

到点和点距离相等的点应满足方程



化简得



1. **代码：**

clear;

Cmax=3;Dir=3;NM=10;

x=[x1(:,1);x3(:,1)];

y=[x1(:,2);x3(:,2)];

[vx,vy]=voronoi(x,y);

l=0;

threshold=1e-3;

for i=1:size(vx,2)

i;

[sx,sy]=find\_symmetry\_p(x(1),y(1),vx(1,i),vy(1,i),vx(2,i),vy(2,i));

kx=find(abs(x-sx)<threshold);

ky=find(abs(y-sy)<threshold);

kk=0;

for j=1:size(kx,1)

flag=0;

for k=1:size(ky,1)

if kx(j)==ky(k) && kx(j)~=1

kk=kx(j)

l=l+1;

ux(:,l)=vx(:,i);

uy(:,l)=vy(:,i);

flag=1;

break;

end

end

if flag==1

break;

end

end

end

plot(x(1:N),y(1:N),'r+',x(N+1:2\*N),y(N+1:2\*N),'b\*',ux,uy,'b-');

hold on;

plot(x(1),y(1),'g\*');

axis equal;

**结果：**



1. **代码：**

[u,v]=voronoi(x,y); %画Voronoi的图像，[vx,vy]里面存的是边的有限顶点

f1=figure();

plot(w1(:,1),w1(:,2),'r\*',w3(:,1),w3(:,2),'b.',u,v,'b-')

xlim([min(x) max(x)])

ylim([min(y) max(y)])

**结果：**



1. **代码：**

f2=figure();

[vx,vy]=voronoi(x,y);

k=0;

threshold=1e-3;

for i=1:size(vx,2)

i;

flag=1;

for j=1:2\*N

[sx,sy]=find\_symmetry\_p(x(j),y(j),vx(1,i),vy(1,i),vx(2,i),vy(2,i));

kx=find(abs(x-sx)<threshold);

ky=find(abs(y-sy)<threshold);

if kx==ky & kx~=j & ((0<kx && kx<=N && 0<j && j<=N) || (N<kx && kx<=2\*N && N<j && j<=2\*N))

i;

kx;

flag=0;

break;

end

end

if flag==1

k=k+1;

ux(:,k)=vx(:,i);

uy(:,k)=vy(:,i);

end

end

plot(x(1:N),y(1:N),'r+',x(N+1:2\*N),y(N+1:2\*N),'b\*',ux,uy,'b-');

axis([-1 3 -1 4]);

**结果：**



1. **代码：**

f3=figure();

d=zeros(2\*N,2\*N);

for i=1:2\*N

for j=i:2\*N

d(j,i)=(x(i)-x(j)).^2+(y(i)-y(j)).^2;

d(i,j)=d(j,i);

end

end

m=0;

hold on;

for i=1:2\*N

flag1=0;

flag=ones(1,2\*N);

flag(i)=0;

for j=1:k

min\_d=1e4;

for l=1:2\*N

if flag(l)==0

continue;

end

if d(i,l)<min\_d

q=l;

min\_d=d(i,l);

end

end

flag(q)=0;

if (q>N && i<=N) || (q<=N && i>N)

flag1=1;

break;

end

end

if flag1==1

m=m+1;

zuoye(m)=x(i);

ey(m)=y(i);

if i<=N

plot(zuoye(m),ey(m),'r\*');

class(m)=0;

else

plot(zuoye(m),ey(m),'b\*');

class(m)=1;

end

end

end

plot(x(1:N),y(1:N),'r.',x(N+1:2\*N),y(N+1:2\*N),'b.');

**结果：**



1. **代码：**

f4=figure();

voronoi(zuoye,ey);

hold on;

plot(x(1:N),y(1:N),'r.',x(N+1:2\*N),y(N+1:2\*N),'b.');

for i=1:length(zuoye)

if class(i)==0

plot(zuoye(i),ey(i),'r\*');

else

plot(zuoye(i),ey(i),'b\*');

end

end

**结果：**



**实验五**

Explore the tradeoff between computational complexity and search accuracy in nearest-neighbor classifiers in the following exercise.

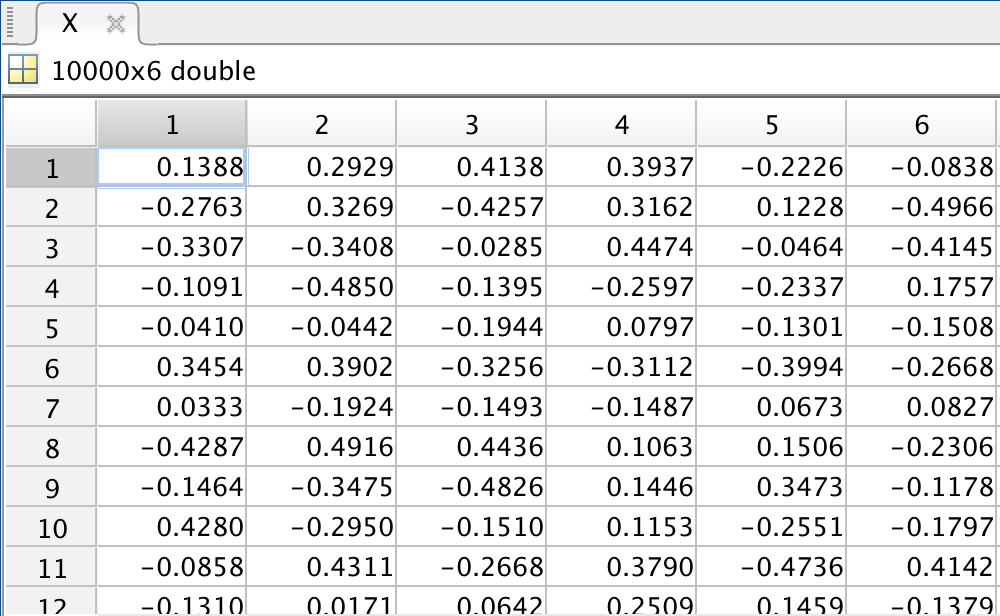
1. **代码：**

d = 6;

X = unifrnd(-1/2,1/2,1e4,d); %prototypes

Xt = unifrnd(-1/2,1/2,100,d); %test set

**结果：**



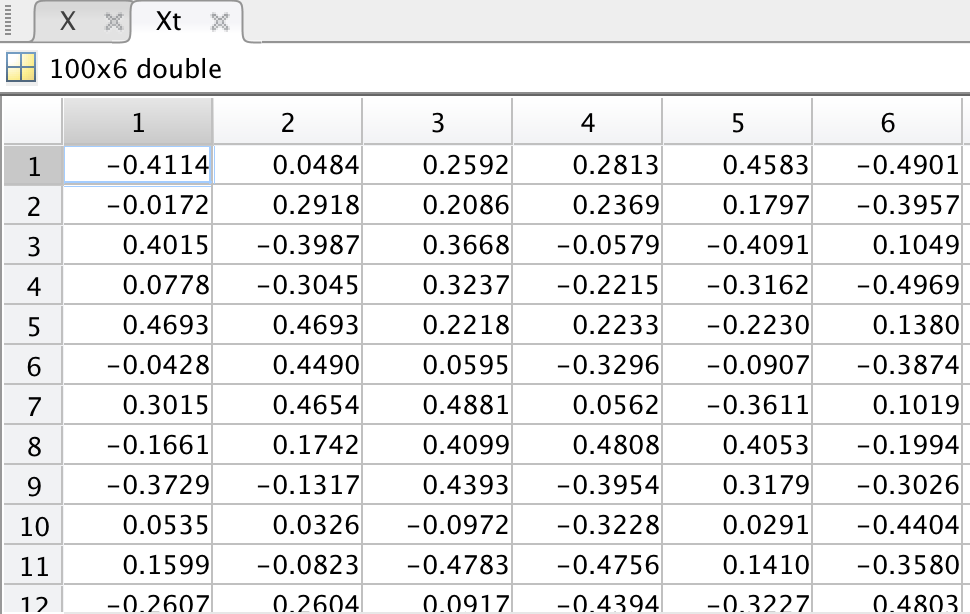
1. **代码：**

d = 6;

X = unifrnd(-1/2,1/2,1e4,d); %prototypes

Xt = unifrnd(-1/2,1/2,100,d); %test set

**结果：**



1. **代码：**

D=size(X,2);N=size(X,1);Nt=size(Xt,1);

for j=1:Nt

min\_dis=1e300;

for i=1:N

dis=norm(X(i,:)-Xt(j,:));

if dis<min\_dis

min\_dis=dis;

neighbor(j)=i;

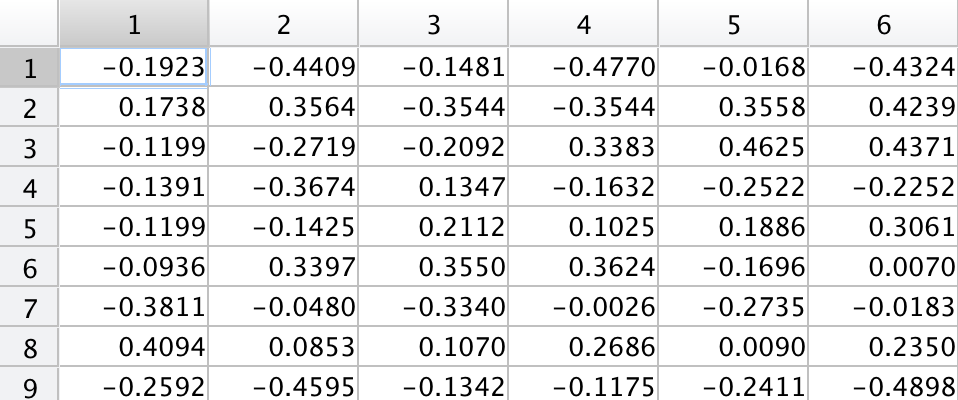
Xt(j)=X(i);

end

end

end

**结果：**



1. **代码：**

D=size(X,2);Nm=size(X,1);Ntt=size(Xt,1);

res=zeros(1,D);

for d=1:D

for j=1:Nt

min\_dis=1e300;

for i=1:Nm

dis=norm(X(i,1:d)-Xt(j,1:d));

if dis<m\_d

m\_d=dis;

neighbor1(d,j)=i;

Xt\_c1(j)=X\_c(i);

end

end

if neighbor1(d,j)==neighbor(j)

res(d)=res(d)+1;

end

end

end

plot(1:d,res(1:d)/Nt);

**结果：**



**实验六**

Consider nearest-neighbor classifiers employing different values of k in the Lk norm or Minkowski metric.

**代码：**

function [class]=ce6\_a(k,p)

C=3;D=3;N=10;

min\_dis=1e4;

for c=1:C

for i=1:N

if k==inf

dis=max(abs(x(c,i,1)-p(1)),max(abs(x(c,i,2)-p(2)),abs(x(c,i,3)-p(3))));

else

dis=(abs(x(c,i,1)-p(1)).^k+abs(x(c,i,2)-p(2)).^k+abs(x(c,i,3)-p(3)).^k).^(1/k);

end

if dis<m\_d

m\_d=dis;

neighbor=x(c,i,:);

class=c;

end

end

end

**结果：**

>> ce6\_a(1,[-0.15,1.17,6.19])

ans = 1

>> ce6\_a(2,[-0.15,1.17,6.19])

ans = 1

>> ce6\_a(4,[-0.15,1.17,6.19])

ans = 1

>> ce6\_a(inf,[-0.15,1.17,6.19])

ans = 1

>> ce6\_a(1,[0.01,1.34,2.6])

ans = 2

>> ce6\_a(2,[0.01,1.34,2.6])

ans = 2

>> ce6\_a(4,[0.01,1.34,2.6])

ans = 2

>> ce6\_a(inf,[0.01,1.34,2.6])

ans = 2

**实验七**

Create a 10x10 pixel grayscale pattern x’ of a handwritten 4.



1. **代码：**

clear;clc;

pic\_rgb = imread('4.png');

pic\_gray = rgb2gray(pic\_rgb);

pic\_re = imresize(pic\_gray, [10, 10]);

vector = pic\_re(:);

max = 20;

for x=0:1:max

pic\_re\_trans = zeros(10, 10);

if x<= 9

pic\_re\_trans(:, 1+x:10) = pic\_re(:, 1:10-x);

end

vector\_trans = pic\_re\_trans(:);

y(x+1) = sqrt(sum((vector\_trans - double(vector)).^2));

end

x = 0:1:max;

plot(x, y)

**结果：**



**实验八**

Repeat Computer exercise 7 but for a handwritten 7, and vertical translations.



1. **代码：**

clear;clc;

% pic\_rgb = imread('4.png');

% pic\_gray = rgb2gray(pic\_rgb);

pic\_gray = imread('7.png');

pic\_re = imresize(pic\_gray, [10, 10]);

vector = pic\_re(:);

max = 20;

for x=0:1:max

pic\_re\_trans = zeros(10, 10);

if x<= 9

pic\_re\_trans(:, 1+x:10) = pic\_re(:, 1:10-x);

end

vector\_trans = pic\_re\_trans(:);

y(x+1) = sqrt(sum((vector\_trans - double(vector)).^2));

end

x = 0:1:max;

plot(x, y)

**结果：**

