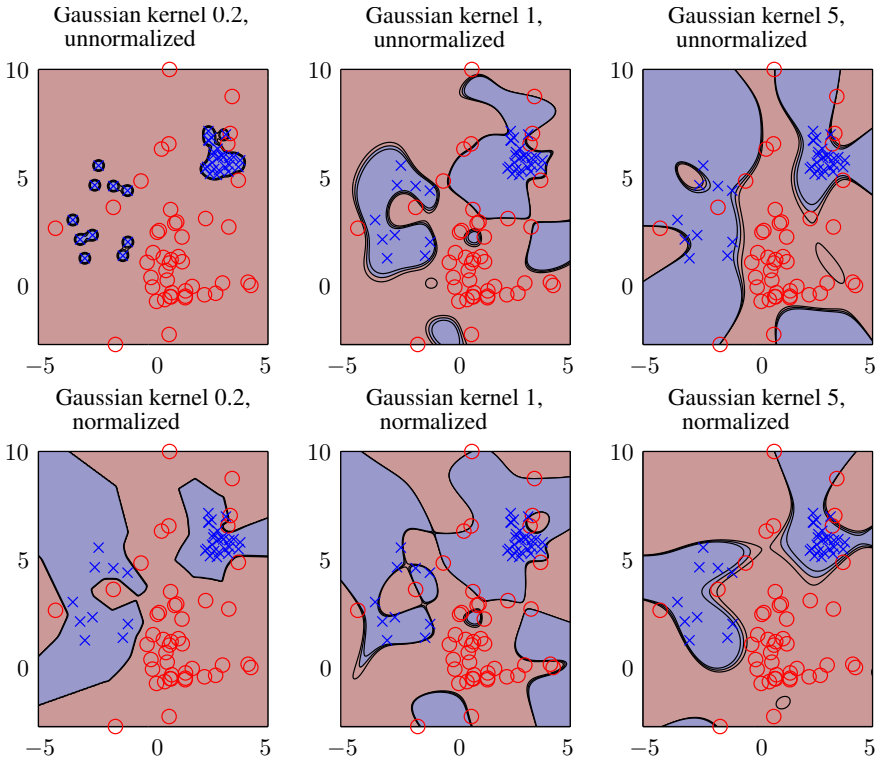
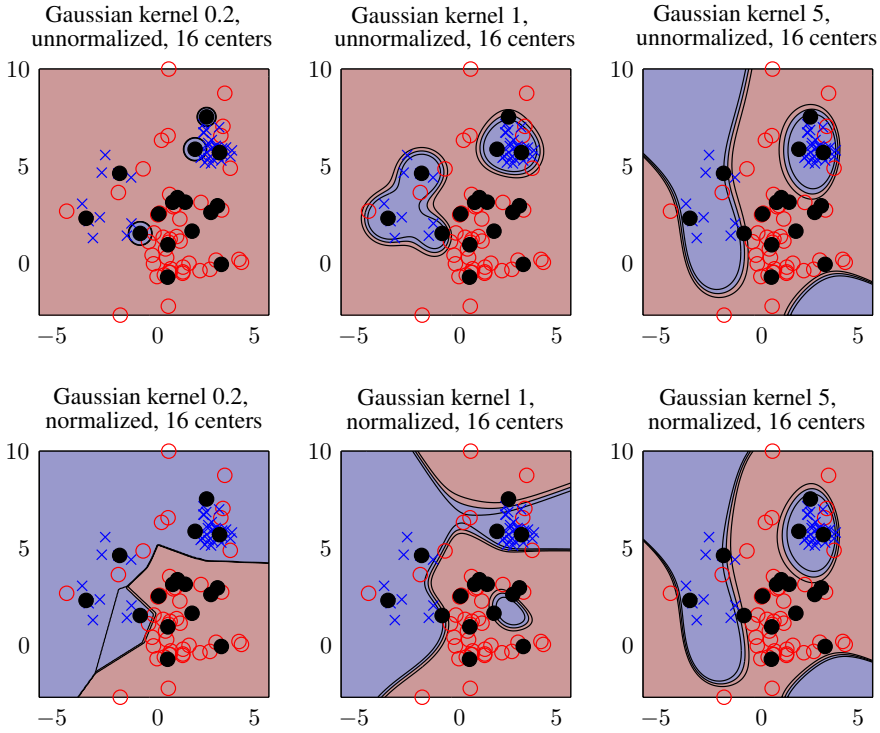
Problem Set 5
Solutions

part a.



part b.



part c. Selecting a width is tricky. Something proportional to the average distance from a center to its closest point, seems about right. When the points are denser, a smaller width is better.

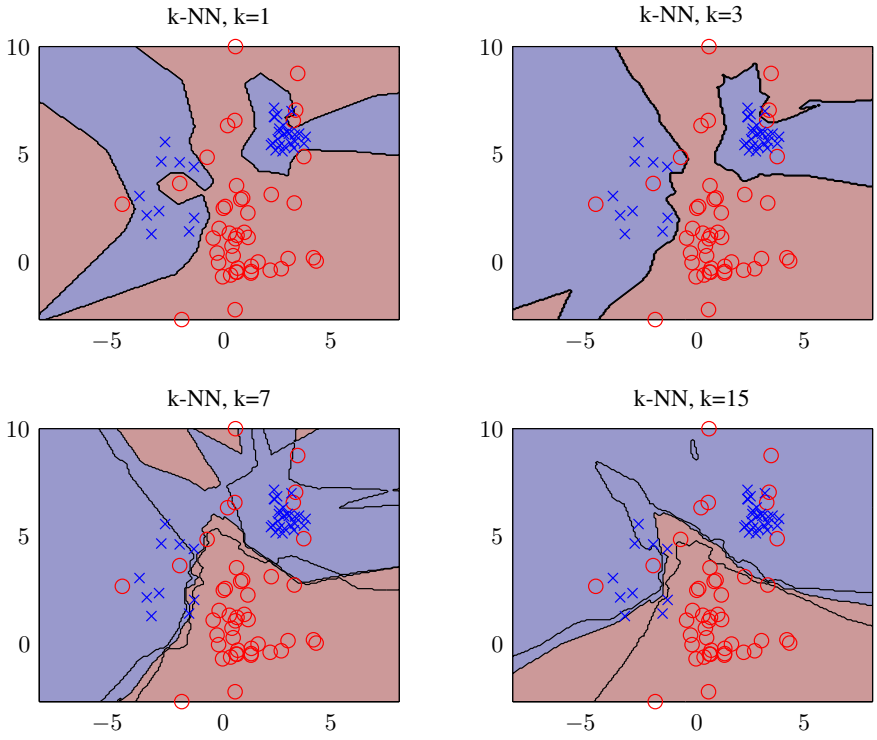
As the kernel width goes to zero, a normalized kernel becomes 1 for the closest point and 0 for other points:

$$\begin{aligned}\lim_{w\to 0}\frac{e^{-\|x-x_i\|^2/w}}{\sum_j e^{-\|x-x_j\|^2/w}} &= \lim_{w\to 0}\frac{e^{-\|x-x_i\|^2/w+D^2/w}}{\sum_j e^{-\|x-x_j\|^2/w+D^2/w}} \\ &= \lim_{w\to 0}\frac{e^{(-\|x-x_i\|^2+D^2)/w}}{|\mathcal{N}|+\sum_{j\notin\mathcal{N}} e^{(-\|x-x_j\|^2+D^2)/w}} \\ &= \frac{1}{|\mathcal{N}|}\delta(i=k)\end{aligned}$$

where we let D be the distance to the closest point(s) to x and $\mathcal{N}$ be the set of points at this distance. That means that all of the terms (except $|\mathcal{N}|$ of them) in the denominator go to zero. The numerator is 1 if $i = k$ and otherwise it also goes to zero. This leaves a function that is 0 for all points except the points that are (equally) closest to x. Unless multiple points are at exactly the same distance, $|\mathcal{N}| = 1$.

Therefore, a RBFN with normalized kernels approaches 1-NN as the kernel width goes to zero, with an averaging rule to break ties.

part d.



code

```
load -ascii class2d.ascii
trainX = class2d(:,1:2);
trainY = class2d(:,3);

% part a.
figure(1)
fnum = 1;
sigmas = [0.2, 1.0, 5.0];
ks = {@gaussk, @gausskn};
for n = 1:2
  for s = sigmas
    w = trainklls(trainX,trainX,trainY,ks{n},s);
    subplot(2,length(sigmas),fnum);
    fnum = fnum+1;
    plotall(trainX,trainY,@klls,200,trainX,w,ks{n},s);
    if (n==1)
      title(sprintf(['Gaussian kernel %g,' ...
          '\nunnormalized'],s));
    else
      title(sprintf(['Gaussian kernel %g,' ...
          '\nnormalized'],s));
    end;
  end;
end;

% part b.
figure(2);
fnum = 1;
C = kmeans(trainX,16);
for n=1:2
  for s = sigmas
    w = trainklls(C,trainX,trainY,ks{n},s);
    subplot(2,length(sigmas),fnum);
    fnum = fnum+1;
    plotall(trainX,trainY,@klls,200,C,w,ks{n},s);
    hold on;
    h = plot(C(:,1),C(:,2),'ko');
    set(h,'MarkerFaceColor',[0 0 0]);
    hold off;
    if (n==1)
      title(sprintf(['Gaussian kernel %g' ...
        ',\nunnormalized, 16 centers'],s));
    else
      title(sprintf(['Gaussian kernel %g' ...
        ',\nnormalized, 16 centers'],s));
    end;
  end;
end;

% part d.
figure(3);
fnum = 1;
ks = [1 3 7 15];
for k = ks
  w = knntrain(trainX,trainY,k);
  subplot(2,2,fnum);
  %figure(fnum);
  fnum = fnum+1;
  plotall(trainX,trainY,@knn,200,w,k);
  title(sprintf('k-NN, k=%d',k));
end;
```

(all functions listed together, but they should be in separate files)

```
function w = trainklls(C,X,Y,kernelfn,varargin)
K = kernelfn(X,C,varargin{:});
w = (K'*K)\K'*Y;

function Y = klls(X,C,w,kernelfn,varargin)
K = kernelfn(X,C,varargin{:});
Y = K*w;

function w = knntrain(X,Y,k)
w = [Y X];
w = reshape(w,[numel(w) 1]);

function Y = knn(testX,w,k)
[testn,d] = size(testX);
pts = reshape(w,[numel(w)/(d+1) d+1]);
trainY = pts(:,1);
trainX = pts(:,2:end);
[trainn,d] = size(trainX);
D=repmat(permute(trainX,[1 3 2]),[1 testn 1])-...
  repmat(permute(testX,[3 1 2]),[trainn 1 1]);
D = sum(D.*D,3);
[D,P] = sort(D);  % slow if many pts, but okay here
P = P(1:k,:);
Y=reshape(trainY(reshape(P,k*testn,1)),k,testn)';
Y = sum(Y,2)/k;

function K = gaussk(X1,X2,sigma)

n1 = size(X1,1);
n2 = size(X2,1);
D = repmat(X1,[1 1 n2]) ...
  - repmat(permute(X2,[3 2 1]),[n1 1 1]);
D = squeeze(sum(D.*D,2));
K = exp(-D/(2*sigma*sigma));

function K = gausskn(X1,X2,sigma)

K = gaussk(X1,X2,sigma);
K = K./(repmat(sum(K,2),[1 size(K,2)]));

function plotdec(testfn,npts,varargin)
axis equal;
v = axis;
[x,y] = meshgrid(v(1):(v(2)-v(1))/npts:v(2),...
    v(3):(v(4)-v(3))/npts:v(4));
[nx,ny] = size(x);
Y = testfn([reshape(x,nx*ny,1) reshape(y,nx*ny,1)],...
    varargin{:});
Y = reshape(Y,nx,ny);
hold on;
[c,h] = contourf(x,y,Y,[-Inf 0.4 0.5 0.6 Inf]);
ch = get(h,'Children');
for i=1:length(ch)
    if (get(ch(i),'CData')<0.5)
        set(ch(i),'FaceColor',[0.8 0.6 0.6]);
    else
        set(ch(i),'FaceColor',[0.6 0.6 0.8]);
    end;
end;

function plotall(X,Y,fn,ndiv,varargin)
h1 = plot(X(Y==1,1),X(Y==1,2),'bx');
hold on;
h2 = plot(X(Y==0,1),X(Y==0,2),'ro');
axis equal;
plotdec(fn,ndiv,varargin{:});
uistack([h1; h2],'top');
hold off;
```