UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
Department of Electrical and Computer Engineering

ECE 544NA Pattern Recognition

Solutions 3

Fall 2013

Assigned: Thursday, September 12, 2013 — Due: Tuesday, September 24, 2013

Reading: NNPR Chapters 2 & 3

Problem 3.1

$p(X|Y=1)$ and $p(X|Y=-1)$ are uniform distributions over the unit disks $(2,0)$ and $(-2,0)$ respectively. Let n be the number of training examples, and suppose we have at least one of them – that is, $n \geq 1$. The 1-nearest neighbor rule is always correct as long as there is at least one training example in each class. Given the geometry of the problem, any point within a particular disk is closer to a point in the disk than *any* point in the other disk (the distance between the two disks is 2). Hence, the only way 1-NN makes a mistake is if *all* the training points are in the other class.

The 3-NN classifier is wrong in the case of 1-NN, but it is also wrong when there is only one point in the correct class. Hence, the probability of error for 3-NN is greater and its risk is greater than that of 1-NN.

Problem 3.2

(a) The linear discriminant between classes C_i and C_j is given by $\{x : \|x-\mu_i\| = \|x-\mu_j\|\}$, which is equivalent to $\{x : (x-\mu_i)^T(x-\mu_i) = (x-\mu_j)^T(x-\mu_j)\}$. After some algebra, it is clear that this is equivalent to

$$\{x : (\mu_j-\mu_i)^T x - \frac{1}{2}(\mu_j-\mu_i)^T(\mu_j+\mu_i) = 0\}$$

hence $w_{ij} = \mu_j - \mu_i$ and $b_{ij} = \frac{1}{2}(\mu_j+\mu_i)$

(b) This is identical to a linear discriminant with a Mahalanobis distance determined by Σ. Hence $w_{1k} = \Sigma^{-1}(\mu_k - \mu_1)$ and $b_{1k} = \frac{1}{2}(\mu_k + \mu_1)$.

$$P(C_1|x) = \left(1 + \sum_{k=2}^{K} e^{-f_{1k}(x)}\right)^{-1}$$

and

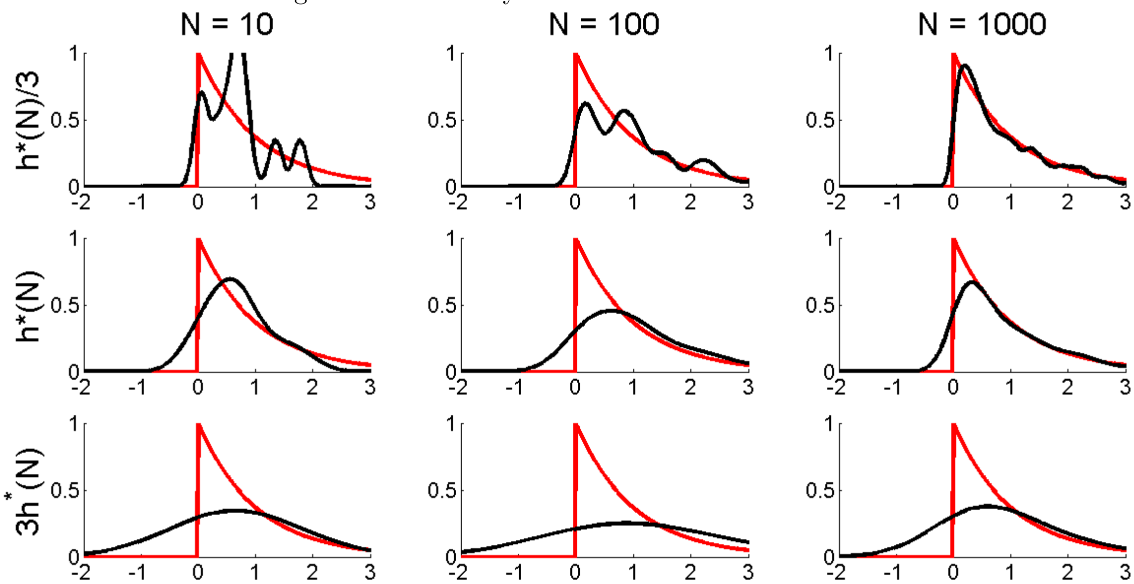
$$f_{1k}(x) = -ln\frac{\pi_k}{\pi_1} - w_{1k}^T(x - b_{1k})$$

Matlab Exercises

Problem 3.3

It is clear that as N increases, our estimator gets better – both visually and in the mean-squared sense. The bandwidth plays a crucial role in smoothing the density, and when it is too large (e.g. three times the *optimal*), there is an oversmoothing effect, and when it is too small, we can almost witness the individual

Figure 3.3-1: Density estimates for different scenarios



data spikes. Since the kernel we use is two-sided, the estimate naturally bleeds over to negative values of x when we have either oversmoothed or when we have too few examples; however, the discontinuity at $x = 0$ is captured well for large N and an appropriate choice of h. Note that our choice of h depends on the sample standard deviation, which is not universally unbiased – given some additional information about the distribution, we can indeed select a better value for h.

```
function randsamples = randgen(mypdf, numsamples)

%Given a pdf mypdf and the number of samples, numsamples, randgen(mypdf,
%numsamples) generates numsamples iid points from the distribution mypdf.
%This function uses the symbolic math toolbox and assumes that mypdf is a
%function of x
%An alternate approach is to do it numerically, by quantizing the
%continuous pdf into tiny bins, and then sampling from an equivalent
%discrete distribution using mvnrnd.

x = sym('x');
eval(['mypdf = ' mypdf ';']);
mycdf = int(mypdf, x) + int(mypdf, x, 0, Inf); %integrate
mycdfinv = finverse(mycdf); %find the inverse of the cdf

randsamples = zeros(numsamples, 1); % number of samples

for i = 1:numsamples
    uniformrand = rand; %generate a uniform random number
    randsamples(i) = subs(mycdfinv, uniformrand);

end
```

```
%% This contains scripts for solving various parts of homework 3

%% Generate the datapoints

datapts = cell(0,0);
N = [10 100 1000];
for i = 1:length(N)
    datapts{i} = randgen('exp(-x)', N(i));
end

%% Estimate the bandwidths

bandwidths = zeros(length(N), 1);
for i = 1:length(bandwidths)
    meansubtract = datapts{i} - sum(datapts{i})/N(i);
    stdevest = sqrt(meansubtract'*meansubtract/N(i));
    bandwidths(i) = 1.06*stdevest*N(i)^(-1/5);
end

%% Kernel density estimator

xvals = -3:0.001:5;
ksdensities = zeros(length(xvals), 9);
bscales = [1/3, 1, 3];
for i = 1:3
    for j = 1:3
        for k = 1:N(j)
            ksdensities(:,(i-1)*3 + j) = ksdensities(:,(i-1)*3 + j) +
            normpdf(xvals, datapts{j}(k), bandwidths(j)*bscales(i))';
        end
        ksdensities(:, (i-1)*3 + j) = ksdensities(:, (i-1)*3 + j)./N(j);
    end
end

%% Plot the estimates

figure;
expdist = exp(-xvals);
posvals = (xvals < 0);
expdist(posvals) = 0;
for i = 1:9
    subplot(3, 3, i);
    hold on;
    plot(xvals, expdist, 'r');
    plot(xvals, ksdensities(:,i), 'k');
end
```