Name: Key

EE435: Biometric Signal Processing Exam 1
Due Friday, 2/17/12 (Beginning of Class)

This is an individual exam. Do not communicate with anyone else: if you have questions, ask me. If you get stuck on a problem, you can ask me for help. In that case, I may determine a point cost for my help, based on the severity of the problem you are having. I will let you know how much the help would cost, then you will have the option to accept the help for a deduction in points, or you can go back and try again on your own.

You'll have this week's lab period to work on it, but it won't be due until Friday at the beginning of class. If turned in later, you will be penalized in points. Use any of your notes, the Digital Image Processing using MATLAB textbook (on the biometrics lab bookshelf), or online MATLAB help as needed.

There are also a few questions to answer, provide:

Hardcopy of:
Your answers to ALL questions in problems 1 and 2 (can be written on these sheets if there is room)
Two scatter plots for problem 2
Code for all parts of problem 2

Softcopy of:
The data.dat file from problem 2 where you have filled in the 2nd column

Problem	Points Possible	Points
1	40	
2	60	
Total:	100	

1. (40 pts total) Suppose you decide to use Bayesian decision theory in your senior design project, which is a pattern recognition system that uses signature biometrics. That is, identity is supposed to be based on how hard you press on a signature sensor as you sign your name. In your project, there are three midshipmen (Percy, Iggy and Cuthbert) who are enrolled in the system and who use it. The signature feature is measured only using three possible values: hard, medium and soft. You've taken some measurements of Percy, Iggy and Cuthbert over the course of the semester and determined some *a priori* knowledge about this situation. When reviewing the data you've collected about these three fellows and the lab, Percy enters the lab 35% of the time, Iggy enters 10% of the time and Cuthbert appears 55% of the time. You've also taken some measurements of their signature when they have entered the lab and have come up with the following information:

Percy signs with hard pressure 20% of the time, signs with medium pressure 60% of the time, and with light pressure 20% of the time.

Iggy signs with hard pressure 90% of the time, signs with medium pressure 8% of the time, and with light pressure 2% of the time.

Cuthbert signs with hard pressure 20% of the time, signs with medium pressure 10% of the time, and with light pressure 70% of the time.

a. (21 pts) Fill in the probability tree included on the following page for this pattern recognition problem. Include probabilitistic expressions such as p(hard|Percy) as well as numbers, so I can follow your tree's construction. Note: there is a blank page in this exam where you should include your work.

b. (4 pts) What is the probability that it is Cuthbert entering the lab? Show your work.

55% (Given)

c. (5 pts) What is the probability that the person entering the lab is pressing hard on the signature sensor? Medium? Light? Show your work.

p(hard) = 0.27
p(med) = 0.273
p(soft) = 0.457

see following page

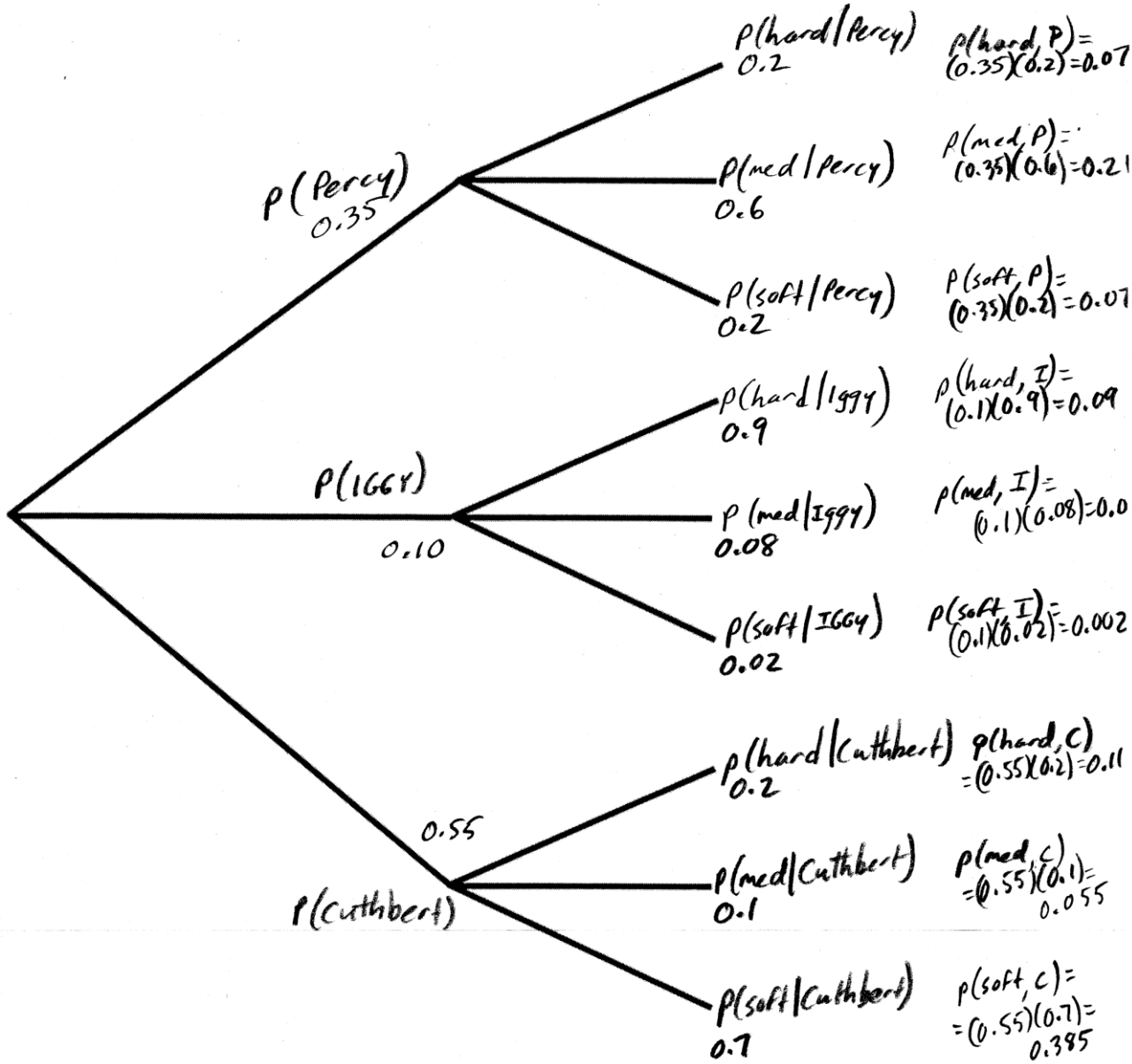
d. (10 pts) Your system decides which person it is just by measuring the signature pressure. State your overall system decision rule by filling in the name of the person below (show work):

If the person is pressing hard, it must be Cuthbert.

* work on following page

If the person is pressing medium, it must be Percy.

If the person is pressing lightly, it must be Cuthbert.



$$p(hard) = p(P, hard) + P(I, hard) + p(C, hard)$$
$$= 0.07 + 0.09 + 0.11 = \underline{0.27}$$

$$p(med) = p(P, med) + p(I, med) + p(C, med)$$
$$= 0.21 + 0.008 + 0.055 = \underline{0.273}$$

$$p(soft) = p(P, soft) + p(I, soft) + p(C, soft)$$
$$= 0.07 + 0.002 + 0.385 = \underline{0.457}$$

Hard:

$$p(Percy \mid hard) = \frac{p(percy, hard)}{p(hard)} = \frac{0.07}{0.27} = 0.2593$$

$$p(Iggy \mid hard) = \frac{P(Iggy, hard)}{p(hard)} = \frac{0.09}{0.27} = 0.3333$$

$$p(Cuthbert \mid hard) = \frac{p(Cuthbert, hard)}{p(hard)} = \frac{0.11}{0.27} = 0.4074$$

Decision rul based on 3 largest values

Med:

$$p(percy \mid med) = \frac{p(percy, med)}{p(med)} = \frac{0.21}{0.273} = 0.7692$$

$$p(Iggy \mid med) = \frac{p(Iggy, med)}{p(med)} = \frac{0.008}{0.273} = 0.0293$$

$$p(cuthbert \mid med) = \frac{p(cuthbert, med)}{p(med)} = \frac{0.055}{0.273} = 0.2015$$

Soft:

$$p(percy \mid soft) = \frac{p(percy, soft)}{p(soft)} = \frac{0.07}{0.457} = 0.1532$$

$$p(Iggy \mid soft) = \frac{p(Iggy, soft)}{p(soft)} = \frac{0.002}{0.457} = 0.0044$$

$$p(cuthbert \mid soft) = \frac{p(cuthbert, soft)}{p(soft)} = \frac{0.385}{0.457} = 0.8425$$

2. (60 pts total) Download the file called *data.dat* from the course website. This file contains samples from 3 different classes. Each sample consists of 2 features.

The matrix called *data* has 4 columns, laid out as follows:
Column #1: Ground truth (class), either 1, 2 or 3
Column #2: All zeros
Column #3: Measurement of feature 1
Column #4: Measurement of feature 2

Use the load command to bring the data into MATLAB:

```
>> load data.dat -ascii
```

a. (10 pts) Create a well-labeled scatter plot of this data, with feature 1 on the x-axis and feature 2 on the y-axis.

See attached

b. (12 pts) From all this data, assume that only Class 2 is important to us right now. This means that for now, there are effectively two classes of interest: *Class 2*, and *NOT Class 2*. Determine the equation of a line that when applied to this data, can be used as a decision boundary to determine either *Class 2* or *NOT-Class 2* BUT with the following design constraint.

Design constraint: using the given data, your decision boundary MUST allow <u>0 false accepts</u> in the given data and <u>minimize</u> the number of false rejects. Record your equation for the line below, and also describe how you determined that it met the design criteria.

Equation: feature 2 = 0.2872 feature 1 + 3.8557

- I drew a line between the highest sample in class 1 and class 3 (connect the dots). I then moved this line vertically upward so that all class 1 + 3 fell below the line. This prevented any false accepts. And because it was closely above the false accept line, this minimized the false rejects. 3 FRs, FRR = 4%

c. (8 pts) Using your equation from part b, state your decision rule for the *Class 2* or *NOT-Class 2* problem.

$testval = f_2 - (0.2872 f_1 + 3.8557)$

if testval ≥ 0, decide class 2

else decide NOT class 2

d. (6 pts) Write a MATLAB function called *classify2.m* that will incorporate your decision boundary and will decide whether the input feature vector is from class 2 or not.

Usage: `y = classify2(x)`

Input: x is the input feature vector [feature1 feature2].
Output: y = 1 if you decide input x is from class 2, and y=0 if not.
Error checking: input must be a vector with 2 values, or else y = [ ] and an error message is displayed.

See attached

e. (8 pts) Using the given data and your *classify2* function, write code to determine the number of false accepts and the number of false rejects for this *Class 2* or *NOT-Class 2* problem, as well as the FAR (%) and FRR (%) and record them:

FA= 0 FR= 3

FAR= 0% FRR= 4%

f. (8 pts) Starting with your answer from part b-c, create a new linear decision boundary that will expand your decision rule to incorporate all three classes; i.e., the two linear boundaries will allow recognition of all three classes. Note: the linear boundary you already created will remain as is. Create another well-labeled scatter plot that includes all of your decision boundaries. Write your overall decision rule below.

Decision rule: $testvalue = f_2 - (0.2872 f_1 + 3.8557)$

if $testvalue \geq 0$, decide class 2
else if $f_1 \geq 5$, decide class 1
else decide class 3

g. (6 pts) Write a MATLAB function called *classify.m* that will incorporate both of your decision boundaries and will decide whether the input feature vector is from class 1, 2 or 3.

Usage: `y = classify(x)`

Input: x is the input feature vector [feature1 feature2]
Output: y = 1 if you decide input x is from class 1, y = 2 if from class 2, y = 3 if from class 3.

See attached

Error checking: input must be a vector with 2 values, or else y = [ ] and an error message is displayed.

h. (8 pts) Write code that implements your 3-class decision rule, using your *classify* function, and computes the total number of errors and overall percent accuracy. Record your results below. Fill in the *data.dat* file's 2nd column with your classification of each sample as class 1, 2 or 3, and email that to me. To write out the file, use:

```
>> save data.dat data -ascii
```

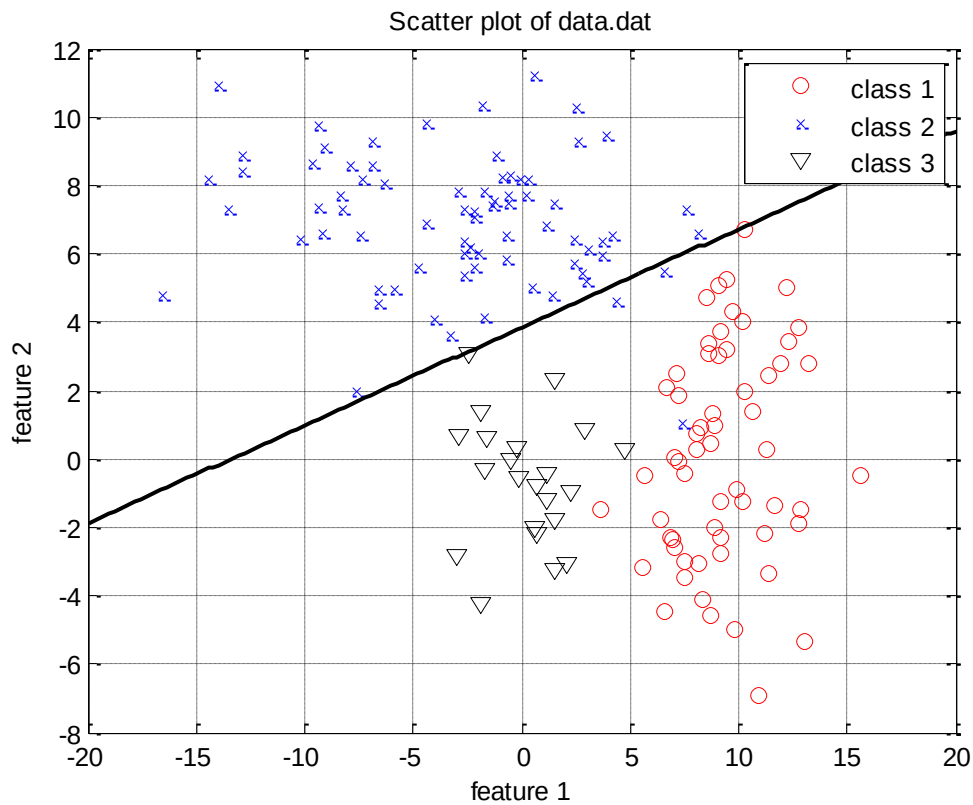
Total # errors = 4

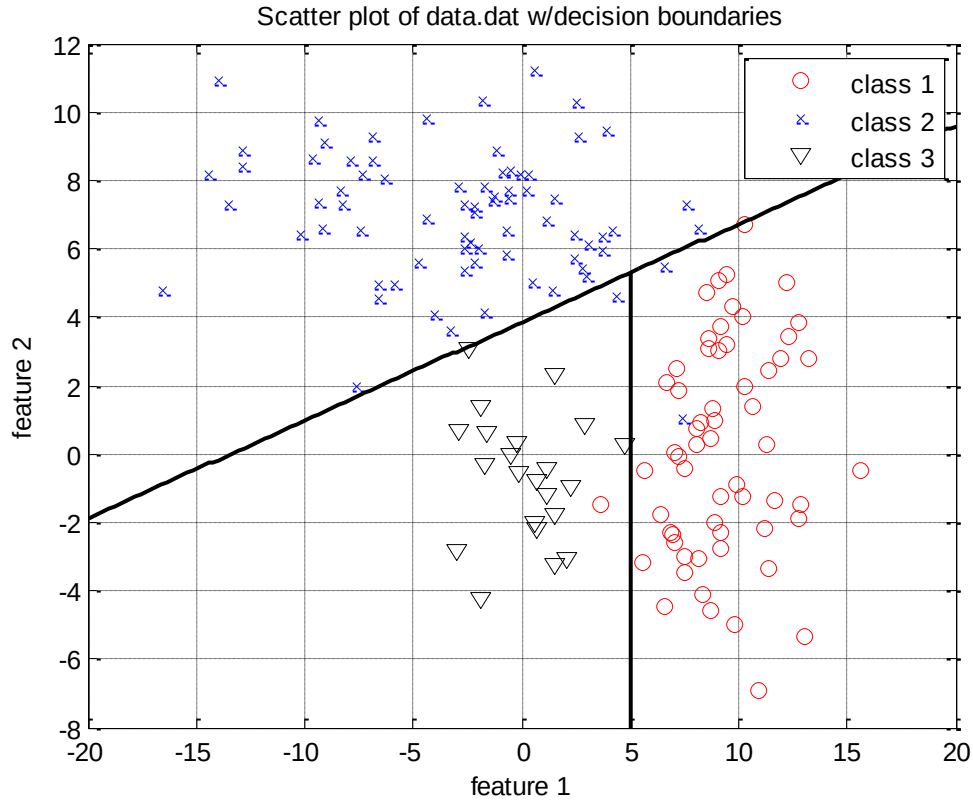
% Accuracy = 97.45 % (show how you computed this)

total # samples

errors

$$\frac{157-4}{157} \times 100\%$$





&&&&&&&&&&&&&&&&&& *classify2* function &&&&&&&&&&&&&&&&&&

```
function class2=classify2(x)
%
%   function class2=classify2(x)
%       This function will classify input 2-element vector x as either
%       class 2 or not class 2 for EE435 Exam 01, Spring 2012, problem 2d.
%       If x is from class 2, class2=1; if not, class2=0.
%       If x is not a 2-element vector, class2 = [].

class2=[];
if ~isvector(x)
    disp('Error in classify2: input not a vector');
    return
end

x1=10.24;y1=6.697;x2=-2.429;y2=3.058;
m=(y2-y1)/(x2-x1);  % slope of line
b=y2-m*x2;
b=b+0.1;  % move line up a little bit

test=x(2)-(m*x(1)+b);
class2=0;                % default...assume NOT class 2
if test >= 0
    class2=1;
end
```

&&&&&&&&&&&&&&&&&& *classify* function &&&&&&&&&&&&&&&&&&

```
function class=classify(x)
%
%   function class=classify(x)
%       This function will classify input 2-element vector x as either
%       class 1,2,3 for EE435 Exam 01, Spring 2012, problem 2g.
%       If x is from class 2, class2=1; if not, class2=0.
%       If x is not a 2-element vector, class2 = [].

class=[];
if ~isvector(x)
    disp('Error in classify2: input not a vector');
    return
end
% For the decision boundary for class 2/not class 2, draw a line that
% connects the samples of class
% 1 and class 3 that have the largest values for feature 2. These
% coordinates are: (f1, f2) = (10.24,6.697) and (-2.429,3.058)
x1=10.24;y1=6.697;x2=-2.429;y2=3.058;
m=(y2-y1)/(x2-x1);  % slope of line
b=y2-m*x2;
b=b+0.1;  % move line up a little bit

test=x(2)-(m*x(1)+b);
if test >= 0
    class=2;
    return
end

% at this point, x is NOT class 2, so determine if class 1 or 3
if x(1) >= 5
    class=1;
  else
    class=3;
end
```

&&&&&&&&&&&&&&&&&& Program to answer problem 2 &&&&&&&&&&&&&&&&&&

```
% EE435, Spring 2012, Exam 01, problem 2

% create scatter plot
load data.dat -ascii
c1=find(data(:,1)==1);
c2=find(data(:,1)==2);
c3=find(data(:,1)==3);

c1f1=data(c1,3);,c1f2=data(c1,4);
c2f1=data(c2,3);,c2f2=data(c2,4);
c3f1=data(c3,3);,c3f2=data(c3,4);

figure(22),plot(c1f1,c1f2,'ro',c2f1,c2f2,'bx',c3f1,c3f2,'kv')

xlabel('feature 1'),ylabel('feature 2'),title('Scatter plot of data.dat')
```

```
% now add in the class 2/NOT class 2 decision boundary
x1=10.24;y1=6.697;x2=-2.429;y2=3.058;
m=(y2-y1)/(x2-x1);  % slope of line
b=y2-m*x2;
b=b+0.1;  % move line up a little bit
x=-20:.2:20;
y=m*x+b;
hold on,plot(x,y,'k','linewidth',2),grid on, hold off
legend('class 1','class 2', 'class 3')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Problem 2, 2-class problem
%make copy of data matrix
data1=data;
N=length(data1(:,1));   % total number of pattern vectors
% since this is a 2-class problem, change class 1 and 3
%   ground truth to class "0", which will mean NOT class 2
data1(c1)=0; data1(c3)=0;
data1(c2)=1;    % class 2 indicated by a 1 in this column

data1(:,2)=0;  % change all values in column 2, and the next loop will change
these
                % to a 2 if above the decision boundary
for k=1:length(data1(:,1))
    data1(k,2)=classify2(data1(k,3:4));
end

correct=0;
FA=0;
FR=0;

for k=1:length(data1(:,1))
    if data1(k,1)==data1(k,2);
        correct=correct+1;
    else
        if data1(k,1)==1 && data1(k,2)==0
            FR=FR+1;
        else
            if data1(k,1)==0 && data1(k,2)==1
                FA=FA+1;;
            end
        end
    end
end
Nimposters=length(find(data1(:,1)==0));
Ngenuines=length(find(data1(:,1)==1));
FAR=FA/Nimposters;
FRR=FR/Ngenuines;

disp(sprintf('2-class problem: FA = %d, FAR=%f, FR=%d,
FRR=%f\n',FA,FAR*100,FR,FRR*100))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Problem 2, 3-class problem
%make a fresh copy of data matrix
data1=data;
% cycle through each row and determine classification
```

```
for k=1:length(data1(:,1))
    test=classify(data1(k,3:4));
    data1(k,2)=test;
end

% determine # errors and accuracy
errors=length(find(data1(:,1) ~= data1(:,2)));

disp(sprintf('3-class problem: # errors = %d, Accuracy = %f %%\n',errors,(N-
errors)/N*100))

% now create 2nd scatter plot with both decision boundaries
figure(23),plot(c1f1,c1f2,'ro',c2f1,c2f2,'bx',c3f1,c3f2,'kv')
legend('class 1','class 2', 'class 3')


% now add in the class 2/NOT class 2 decision boundary
x1=10.24;y1=6.697;x2=-2.429;y2=3.058;
m=(y2-y1)/(x2-x1);  % slope of line
b=y2-m*x2;
b=b+0.1;  % move line up a little bit
x=-20:.2:20;
y=m*x+b;
hold on,plot(x,y,'k','linewidth',2),grid on
plot([5 5],[-8 5.3],'k','linewidth',2),grid on,hold off
xlabel('feature 1'),ylabel('feature 2'),title('Scatter plot of data.dat
w/decision boundaries')
legend('class 1','class 2', 'class 3')
```