Homework 8: Log() part - nicholas hardy

```
%%MATLAB Log Likelihood Script for Homework 8
```

Answers

dataset0 = benignfull.csv

dataset1 = malignant.csv

Probability of error for identity covariance is 0.085

Probability of error for same covariance is 0.5

Probability of error for different covariances is 0.015

CATS AND DOGS

Probability of error for identity covariance is 0.19

Probability of error for same covariance is 0.5

Probability of error for different covariances is 0.5

```
%dataset0 = readmatrix("syntheticH0.csv");
%dataset1 = readmatrix("syntheticH1.csv");
%dataset0 = readmatrix("benignfull.csv");
%dataset1 = readmatrix("malignantfull.csv");
[dataset0 ,dataset1] = read_cats_dogs;
[n0, d0] = size(dataset0);
[n1, d1] = size(dataset1);
if (d0 == d1)
    d = d0;
else
    error("dataset0 and dataset1 have a different number of columns.")
end

%Split dataset into training and test data.
train0 = dataset0(1:floor(n0/2),:);
test0 = dataset0(floor(n0/2)+1:n0,:);
train1 = dataset1(1:floor(n1/2),:);
test1 = dataset1(floor(n1/2)+1:n1,:);
n0test = size(test0,1);
n1test = size(test1,1);

%Estimate mean vectors and covariance matrices from training data.
mu0 = mean(train0);
mu1 = mean(train1);
sigma0 = cov(train0);
sigma1 = cov(train1);
```

```
%Apply decision rules.
H0guesses_idcov = zeros(n0test,1);
H1guesses_idcov = zeros(n1test,1);

for i = 1:n0test
    currentdata= test0(i,:);
    %%%Problem 8.4(e) code goes here.  Call the
    %%% function closest average to classify currentdata and store the
    %%% output in the reight array

    H0guesses_idcov(i) = closest_average(currentdata,mu0 ,mu1);
end

for i = 1:n1test
    currentdata = test1(i,:);
    %%%Problem 8.4(e) code goes here.  Repeat the above for data from test1
    %%%

    H1guesses_idcov(i) = closest_average(currentdata,mu0 ,mu1);
end

Pe_idcov = proberror(H0guesses_idcov,H1guesses_idcov);
fprintf('Probability of error for identity covariance is %.2g.\n',Pe_idcov);
```

```
Probability of error for identity covariance is 0.19.
```

```
H0guesses_samecov = zeros(n0test,1);
H1guesses_samecov = zeros(n1test,1);


%%%Problem 8.4(f) code goes here.  First, compute the pooled covariance
%%%  and its pseudoinverse %%%

S0 = cov(train0);
S1 = cov(train1);
n = n0train + n1train;
Sp = ((n0train-1)/n)*S0 + ((n1train-1)/n)*S1;

Sp_inv = pinv(Sp);

PooledVar = Sp;

d = size(test0,2);
alpha = 0.05;  % significance level

for i = 1:n0test
    currentdata = test0(i,:);
    %%%Problem 8.4(f) code goes here.  Implement the log llikelihood
    %%% test and apply it to currentdata, storing the answers in the array
    %%% below.

     log_ratio = (currentdata - mu1) * Sp_inv * (currentdata - mu1)' ...
                - (currentdata - mu0) * Sp_inv * (currentdata - mu0)';
```

```
     t = chi2inv(1-alpha,d);
    % Classify test point
    H0guesses_samecov(i) = (log_ratio < t);
end

for i = 1:n1test
    currentdata = test1(i,:);
    % Compute log-likelihood ratio for H0 and H1
    log_ratio = (currentdata - mu1) * Sp_inv * (currentdata - mu1)' ...
                - (currentdata - mu0) * Sp_inv * (currentdata - mu0)';
    % Compute threshold value
    t = chi2inv(1-alpha,d);
    % Classify test point
    H1guesses_samecov(i) = (log_ratio < t);
end

Pe_samecov = proberror(H0guesses_samecov,H1guesses_samecov);
fprintf('Probability of error for same covariance is %.2g.\n',Pe_samecov);
```

```
Probability of error for same covariance is 0.5.
```

```
H0guesses_diffcov = zeros(n0test,1);
H1guesses_diffcov = zeros(n1test,1);

%%% Problem 8.4(g) code goes here.  Compute the pseudoinverses of the
%%% covariances here, outside of the loop. Do as much algebra as you can
%%% that does not depend on the data.

inv_sigma0 = inv(sigma0);
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
 1.735923e-22.
```

```
inv_sigma1 = inv(sigma1);
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
 3.737169e-22.
```

```
P0 = n0train / (n0train + n1train);
P1 = n1train / (n0train + n1train);

g0 = @(x) -0.5*log(det(sigma0))-0.5*(x-mu0)*inv_sigma0*(x-mu0)'+log(P0);
g1 = @(x) -0.5*log(det(sigma1))-0.5*(x-mu1)*inv_sigma1*(x-mu1)'+log(P1);

%%% loop for testing H0 data
for i = 1:n0test
    currentdata = test0(i,:);
    %%%Problem 8.4(g) code goes here.  Implement the log-likelihood test
    %%% with different covariances, store the decision in the array

    H0guesses_diffcov(i) = g1(currentdata) > g0(currentdata);
end
```

```
%%% loop for testing H1 data
for i = 1:n1test
    currentdata = test1(i,:);
    %%%Problem 8.3(g) code goes here.  Implement the log-likelihood test
    %%% with different covariances, store the decision in the array

    H1guesses_diffcov(i) = g1(currentdata) > g0(currentdata);
end

Pe_diffcov = proberror(H0guesses_diffcov,H1guesses_diffcov);
fprintf('Probability of error for different covariances is %.2g.\n',Pe_diffcov);
```

```
Probability of error for different covariances is 0.5.
```