CS6140 Machine Learning Fall 2014 Homework 3, Wei Luo

PROBLEM 1

Using Gaussian Discriminant Analysis, I got:
average train error rate: 0.091212 average test error rate: 0.096498
Since the average accuracy is over 0.9, it seems to me that the gaussian assumption holds for this data set.

PROBLEM 2

The Error Table for Naïve Bayes Classifier, Model with Bernoulli Random Variables:

fold#	false positive rate	false negative rate	overall error rate
1	0.142322	0.067010	0.110629
2	0.134021	0.112426	0.126087
3	0.123288	0.089286	0.110870
4	0.104651	0.084158	0.095652
5	0.119718	0.062500	0.097826
6	0.127208	0.056497	0.100000
7	0.118467	0.127168	0.121739
8	0.114695	0.049724	0.089130
9	0.128571	0.083333	0.110870
10	0.123596	0.062176	0.097826
average	0.12365375	0.07942785	0.10606291

The Error Table for Naïve Bayes Classifier, Model with Gaussian Random Variables:

fold#	false positive rate	false negative rate	overall error rate
1	0.186207	0.140351	0.169197
2	0.121622	0.091463	0.110870
3	0.102273	0.056122	0.082609
4	0.138996	0.084577	0.115217
5	0.175000	0.077778	0.136957
6	0.178832	0.112903	0.152174
7	0.136842	0.085714	0.117391
8	0.138686	0.096774	0.121739
9	0.199301	0.137931	0.176087
10	0.157143	0.094444	0.132609
average	0.15349013	0.09780588	0.13148496

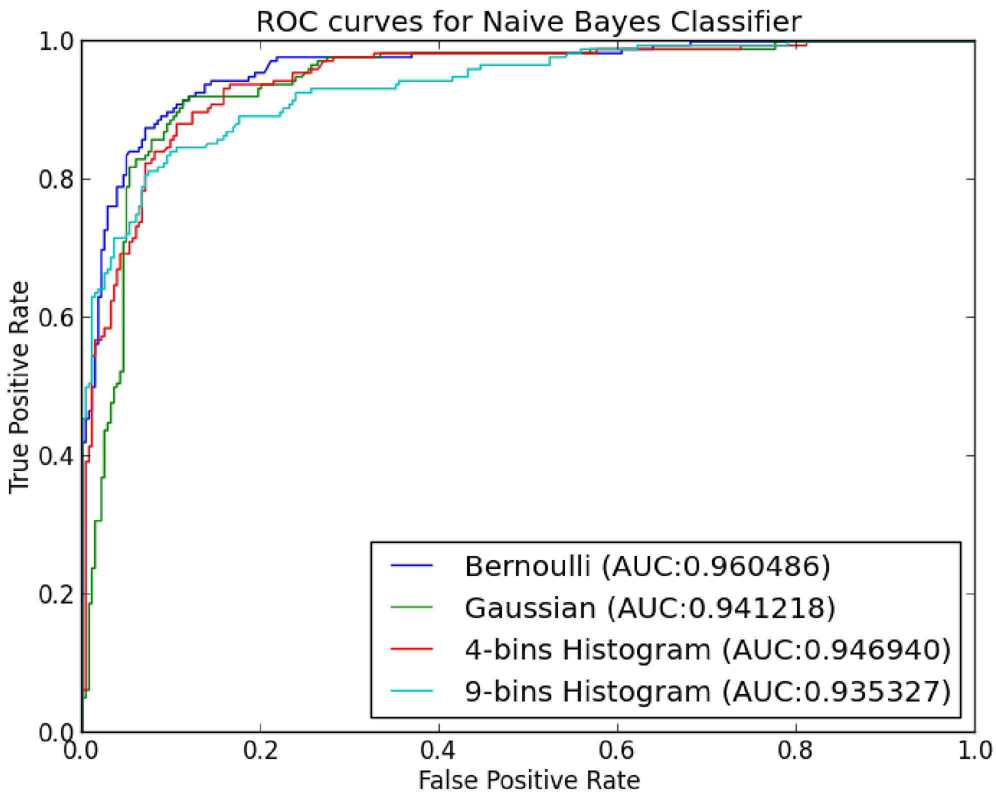
The Error Table for Naïve Bayes Classifier (4-bins Histogram):

fold#	false positive rate	false negative rate	overall error rate
1	0.086957	0.221622	0.140998
2	0.097222	0.151163	0.117391
3	0.076642	0.139785	0.102174
4	0.103321	0.190476	0.139130
5	0.064982	0.218579	0.126087
6	0.084559	0.196809	0.130435
7	0.078292	0.195531	0.123913
8	0.081851	0.201117	0.128261
9	0.094737	0.194286	0.132609
10	0.063604	0.225989	0.126087
average	0.08321663	0.19353558	0.12670848

The Error Table for Naïve Bayes Classifier (9-bins Histogram):

fold#	false positive rate	false negative rate	overall error rate
1	0.032374	0.251366	0.119306
2	0.025180	0.340659	0.150000
3	0.025271	0.371585	0.163043
4	0.007692	0.485000	0.215217
5	0.021818	0.356757	0.156522
6	0.027119	0.400000	0.160870
7	0.027027	0.358209	0.171739
8	0.049123	0.291429	0.141304
9	0.023649	0.329268	0.132609
10	0.017544	0.365714	0.150000
average	0.02567962	0.3549987	0.15606102

The ROC curves and AUC for each classifier is:



PROBLEM 3

(next page)

```
function [label, model, llh] = emgm(X, init)
% Perform EM algorithm for fitting the Gaussian mixture model.
%   X: d x n data matrix
%   init: k (1 x 1) or label (1 x n, 1<=label(i)<=k) or center (d x k)
% Written by Michael Chen (sth4nth@gmail.com).
%% initialization
fprintf('EM for Gaussian mixture: running ... \n');
R = initialization(X,init);
[~,label(1,:)] = max(R,[],2);
R = R(:,unique(label));

tol = 1e-10;          converge threshold 10^-10
maxiter = 500;        maximum 500 steps
llh = -inf(1,maxiter);     initialize loglikelihoods with -inf s
converged = false;    init converged
t = 1;                init t
while ~converged && t < maxiter       loop while not converged and not reach maxiter
    t = t+1;
    model = maximization(X,R);        m-step
    [R, llh(t)] = expectation(X,model);      e-step

    [~,label(:)] = max(R,[],2);
    u = unique(label);   % non-empty components
    if size(R,2) ~= size(u,2)
        R = R(:,u);   % remove empty components
    else
        converged = llh(t)-llh(t-1) < tol*abs(llh(t));  converged if difference between current and previous likelihood less than threshold.
    end
end
llh = llh(2:t);     llh value starts from llh(2), first value not set in loop.
if converged                 converged
    fprintf('Converged in %d steps.\n',t-1);
else                         not converged in maxiter steps
    fprintf('Not converged in %d steps.\n',maxiter);
end

function R = initialization(X, init)
[d,n] = size(X);      get X (dxn data) dimensions
if isstruct(init)  % initialize with a model
    R  = expectation(X,init);      return expectation for X from given model
elseif length(init) == 1  % random initialization
    k = init;
    idx = randsample(n,k);      randomly pick sample k indexes from 1 to n
    m = X(:,idx);               get sample data
    [~,label] = max(bsxfun(@minus,m'*X,dot(m,m,1)'/2),[],1);
    [u,~,label] = unique(label);
    while k ~= length(u)
        idx = randsample(n,k);      pick random samples
        m = X(:,idx);
        [~,label] = max(bsxfun(@minus,m'*X,dot(m,m,1)'/2),[],1);
        [u,~,label] = unique(label);
    end
    R = full(sparse(1:n,label,1,n,k,n));    make a full (1xn) sparse matrix which is a random membership matrix
elseif size(init,1) == 1 && size(init,2) == n  % initialize with labels
    label = init;
    k = max(label);
    R = full(sparse(1:n,label,1,n,k,n));
elseif size(init,1) == d  %initialize with only centers
    k = size(init,2);
    m = init;
    [~,label] = max(bsxfun(@minus,m'*X,dot(m,m,1)'/2),[],1);
    R = full(sparse(1:n,label,1,n,k,n));
else      otherwise, init is not valid.
```

```
    error('ERROR: init is not valid.');
end

function [R, llh] = expectation(X, model)
mu = model.mu;
Sigma = model.Sigma;          } get parameters from model.
w = model.weight;

n = size(X,2);                get size parameters
k = size(mu,2);
logRho = zeros(n,k);          initialize log gaussian pdf values

for i = 1:k
    logRho(:,i) = loggausspdf(X,mu(:,i),Sigma(:,:,i)); calculate log gaussian pdf values
end
logRho = bsxfun(@plus,logRho,log(w));     add weights
T = logsumexp(logRho,2);
llh = sum(T)/n; % loglikelihood
logR = bsxfun(@minus,logRho,T);
R = exp(logR);                       update Z

function model = maximization(X, R)
[d,n] = size(X);
k = size(R,2);                get size parameters

nk = sum(R,1);                number of points for each gaussian
w = nk/n;                      weights for each gaussian
mu = bsxfun(@times, X*R, 1./nk);   means for each gaussian

Sigma = zeros(d,d,k);
sqrtR = sqrt(R);
for i = 1:k
    Xo = bsxfun(@minus,X,mu(:,i));     x - μ
    Xo = bsxfun(@times,Xo,sqrtR(:,i)');     z_i (x-μ)
    Sigma(:,:,i) = Xo*Xo'/nk(i);       covariance for each gaussian
    Sigma(:,:,i) = Sigma(:,:,i)+eye(d)*(1e-6); % add a prior for numerical stability
end

model.mu = mu;
model.Sigma = Sigma;          } update parameters.
model.weight = w;

function y = loggausspdf(X, mu, Sigma)
d = size(X,1);
X = bsxfun(@minus,X,mu);      x-μ
[U,p]= chol(Sigma);
if p ~= 0
    error('ERROR: Sigma is not PD.');
end
Q = U'\X;
q = dot(Q,Q,1);  % quadratic term (M distance)
c = d*log(2*pi)+2*sum(log(diag(U)));   % normalization constant
y = -(c+q)/2;            return pdf value.
```

PROBLEM 4

A) The results for 2 gaussian:

Gaussian$_1$: points: 1981; mean: [2.94846981, 3.0421113]
covariance: [[0.8992528, 0.01219505], [0.01219505, 2.96950283]]
Gaussian$_2$: points: 4019; mean: [7.0078808, 3.98162176]
covariance: [[0.95936573, 0.4865028], [0.4865028, 0.99568567]]

B) The results for 3 gaussian:

Gaussian$_1$: points: 3098; mean: [6.98857865, 4.03307065]
covariance: [[1.00480526, 0.49076192], [0.49076192, 1.0080817]]
Gaussian$_2$: points: 1945; mean: [2.91656552, 2.8932281]
covariance: [[0.86404608, -0.20507606], [-0.20507606, 2.9738529]]
Gaussian$_3$: points: 4957; mean: [4.99043274, 7.03134783]
covariance: [[0.91578388, 0.18253052], [0.18253052, 0.90038684]]

PROBLEM 5

a)

$$\frac{P(B|A,C)P(A|C)}{P(B|C)} = \frac{P(A,B,C)/P(A,C) * P(A,C)/P(C)}{P(B,C)/P(C)} = \frac{P(A,B,C)}{P(B,C)} = P(A|B,C)$$

b) The prior probability of the coin being fair is $P(F) = \frac{F}{F+1}$. So the prior probability of the coin being double-headed is $P(D) = 1 - P(F) = \frac{1}{F+1}$. If we see n heads in a row, the probability that the coon is fair is: $P(F|nH) = \frac{P(nH|F)P(F)}{P(nH)}$; the probability that the coon is fair is: $P(D|nH) = \frac{P(nH|D)P(D)}{P(nH)}$. If the coin is fair, the probability of seeing a head is $P(H|F) = \frac{1}{2}$. the probability of seeing n heads in a row is $P(nH|F) = P(H|F)^n = (\frac{1}{2})^n$. If the coin is double-headed, the probability of seeing a head and n head in a row is $P(H|D) = P(nH|D) = 1$. We want a better than even chance that the coin is double-headed, which means $P(D|nH) > P(F|nH)$. So $\frac{P(nH|D)P(D)}{P(nH)} > \frac{P(nH|F)P(F)}{P(nH)}$. So $1 * \frac{1}{F+1} > (\frac{1}{2})^n * \frac{F}{F+1}$.
We get $n > \log_2 F$. So we need to see more than $\log_2 F$ heads in a row to become convinced that there is a better than even chance that the coin is double-headed.

PROBLEM 6

Flip different number of coins with different probabilities. Then run em algorithm:

π	0.8	0.2
em_π	0.84163594	0.15836406
probs	0.75	0.4
em_probs	0.74193735	0.386823

π	0.4	0.6
em_π	0.41665103	0.58334897
probs	0.9	0.2
em_probs	0.89850985	0.19634036

π	0.1	0.9
em_π	0.29550855	0.70449145
probs	0.45	0.51
em_probs	0.45889733	0.51653134

π	0.2	0.3	0.5
em_π	0.19472662	0.31741561	0.48785777
probs	0.99	0.47	0.08
em_probs	0.99373054	0.48164992	0.07176479

π	0.1	0.1	0.2	0.3	0.3
em_π	0.13627108	0.0884572	0.11475013	0.3240638	0.33645778
probs	0.98	0.69	0.38	0.21	0.03
em_probs	0.97007034	0.64665238	0.30068529	0.28223276	0.03697732

PROBLEM 7

Tried with K = {1,2,4,9}. The average error rates are as follows:

K	1	2	4	9
Error Rate	0.132321	0.130152	0.097614	0.091106

And the ROC curve for each K is: