```
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);    Calling the initialization function
[~,label(1,:)] = max(R,[],2);
R = R(:,unique(label));

tol = 1e-10;       defining Threshold for convergence
maxiter = 500;     Defining the number of iterations
llh = -inf(1,maxiter);
converged = false;
t = 1;
while ~converged && t < maxiter     run the loops until convergence or number of iterations
    t = t+1;
    model = maximization(X,R);
    [R, llh(t)] = expectation(X,model);  calling expectation function

    [~,label(:)] = max(R,[],2);
    u = unique(label);   % non-empty components    unique classes
    if size(R,2) ~= size(u,2)   If size of 2nd dim of R is not equal to size of 2nd dim of u
        R = R(:,u);   % remove empty components
    else
        converged = llh(t)-llh(t-1) < tol*abs(llh(t)); Condition for converged i.e value less than
    end                                                defined threshold
    figure(gcf); clf;
    spread(X,label);
    muA = model.mu;
    SigmaA = model.Sigma;
    wA = model.weight;
    k = size(muA,2);        Size of 2nd dim of muA matrix
    % figure(12); clf;
    % for i=1:k
    %   mu1 =muA(i,:)
    %   Sigma1=SigmaA(i,:)
    %   w1=wA(i)
    %   xx= mvnrnd(mu1, Sigma1, 1000);
    %   yy= mvnpdf(xx,mu1,Sigma1);
    %   plot3(xx(:,1), xx(:,2), yy, '.b'); hold on;
    % end

    pause;



end
llh = llh(2:t);
if converged
    fprintf('Converged in %d steps.\n',t-1);
else
    fprintf('Not converged in %d steps.\n',maxiter);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function R = initialization(X, init)     Function for initialization
[d,n] = size(X);        Rows and columns of the X data matrix
if isstruct(init)  % initialize with a model     if init is multi-dimentional
    R  = expectation(X,init);
elseif length(init) == 1  % random initialization     if length of init is 1
    k = init;                   k = init
    idx = randsample(n,k);      k samples in the range 1 to n
    m = X(:,idx);       Taking a part of X data all rows and idx columns into m
    [~,label] = max(bsxfun(@minus,m'*X,dot(m,m,1)'/2),[],1);  assign samples to nearest gaussian mix
    [u,~,label] = unique(label);    remove empty gaussian misture
```

```
    while k ~= length(u)
        idx = randsample(n,k);   k random samples in range 1 to n
        m = X(:,idx);      Taking a part of X data all rows and idx columns into m
        [~,label] = max(bsxfun(@minus,m'*X,dot(m,m,1)'/2),[],1);   assign samples to nearest gaussian mix
        [u,~,label] = unique(label);
    end
    R = full(sparse(1:n,label,1,n,k,n));
elseif size(init,1) == 1 && size(init,2) == n  % initialize with labels  get the size of 1st and 2nd dim of matrix
    label = init;
    k = max(label);                         Returns maximum element
    R = full(sparse(1:n,label,1,n,k,n));     convert to full matrix
elseif size(init,1) == d  %initialize with only centers
    k = size(init,2);       Size of 2nd dim of init
    m = init;
    [~,label] = max(bsxfun(@minus,m'*X,dot(m,m,1)'/2),[],1);  assign samples to nearest gaussian mix
    R = full(sparse(1:n,label,1,n,k,n));    convert to full matrix
else
    error('ERROR: init is not valid.');
end

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

n = size(X,2);
k = size(mu,2);
logRho = zeros(n,k);

for i = 1:k
    logRho(:,i) = loggausspdf(X,mu(:,i),Sigma(:,:,i));
end
logRho = bsxfun(@plus,logRho,log(w));          Addition
T = logsumexp(logRho,2);          will sum across rows instead of columns.
llh = sum(T)/n; % loglikelihood
logR = bsxfun(@minus,logRho,T);    subtract logRho and T
R = exp(logR);


function model = maximization(X, R)
[d,n] = size(X);                    Row, column size of matrix X
k = size(R,2);           size of 2nd dime of R matrix

nk = sum(R,1);
w = nk/n;
mu = bsxfun(@times, X*R, 1./nk);      Multiplication

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

model.mu = mu;
model.Sigma = Sigma;
model.weight = w;

function y = loggausspdf(X, mu, Sigma)
d = size(X,1);                 get the size of 1st dimension of matrix
X = bsxfun(@minus,X,mu);   Subtract X and mu
[U,p]= chol(Sigma);            Cholesky factorization
if p ~= 0        Sigma is not positive definite
    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;

```