Problem 3: EM Algorithm

1 Main Function

```
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).
```

Implements EM algorithm for Gaussian Mixture Models.
Inputs:

- `X`: Data matrix ($d \times n$) where $d =$ dimensions, $n =$ samples
- `init`: Flexible initialization:
  - Single number k (number of clusters)
  - Label vector ($1 \times n$) with assignments
  - Center matrix ($d \times k$) with initial centers

Outputs:

- `label`: Final cluster assignments
- `model`: Learned GMM parameters (μ, Σ, π)
- `llh`: Log-likelihood history

2 Initialization

```
%% initialization
fprintf('EM for Gaussian mixture: running ... \n');
R = initialization(X,init);
[~,label(1,:)] = max(R,[],2);
R = R(:,unique(label));
```

Create initial responsibility matrix R. Convert soft assignments to hard labels: take maximum responsibility to each cluster. Then, remove empty clusters.

3 Algorithm Parameters

```

tol = 1e-10;
maxiter = 500;
llh = -inf(1,maxiter);
converged = false;
t = 1;
```

Set convergence tolerance, maximum number of iterations, and initialize log-likelihood array.

4 Main EM Iteration

```
while ~converged && t < maxiter
    t = t+1;
    model = maximization(X,R);
    [R, llh(t)] = expectation(X,model);
```

Continue while not converged AND not reach max iteration count.
M-step (Maximization): update model parameters based on current responsibilities.
E-step (Expectation): compute new responsibilities R and log-likelihood history.

5 Cluster Management

```

    [~,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));
    end
    figure(gcf); clf;
    spread(X,label);
    muA = model.mu;
    SigmaA = model.Sigma;
    wA = model.weight;
    k = size(muA,2);
    % 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
```

Update hard cluster assignments based on maximum responsibilities.
Remove empty clusters from R if any.
Update convergence status based on relative change in log-likelihood and tolerance.
Extract model parameters: number of clusters `k`, means `mu`, covariances `Sigma`, and weights w.
Trim log-likelihood array.

6 Initialization Function

```
function R = initialization(X, init)
[d,n] = size(X);
if isstruct(init)  % initialize with a model
    R = expectation(X,init);
elseif length(init) == 1  % random initialization
    k = init;
    idx = randsample(n,k);
    m = X(:,idx);
    [~,label] = max(bsxfun(@minus,m'*X,dot(m,m,1)'/2),[],1);
    [u,~,label] = unique(label);
    while k ~= length(u)
        idx = randsample(n,k);
        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));
elseif size(init,1) == 1 && size(init,2) == n
    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
    error('ERROR: init is not valid.');
end
```

Extract data dimensions

- d = number of features
- n = number of data points

6.1 Initialize with a model:

If init is a model struct, use E-step to compute responsibilities R.

6.2 Random initialization:

select k random points as initial centers, then assign points to nearest center using the trick $\arg\max(m'X - \|m\|^2/2) = \arg\min(\|X - m\|^2)$, ensuring consecutive labels (1,2,3...).

Handle empty clusters:

- Keep resampling until all k clusters have members
- Prevents degenerate initialization

Create responsibility matrix R: Using sparse matrix and creates binary $n \times k$ matrix.

6.3 Initialize with labels:

Provides cluster assignments then convert to matrix format.

6.4 Initialize with centers

Provides $d \times k$ center matrix and assign points to nearest center

7 E-step Function

```
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
```

Extract model parameters

- `mu`: cluster means
- `Sigma`: covariance matrices
- `w`: mixture weights

Compute log probabilities: For each cluster i, compute $\log p(x|\mu_i, \Sigma_i)$

8 Calculate Responsibilities

```
logRho = bsxfun(@plus,logRho,log(w));
T = logsumexp(logRho,2);
llh = sum(T)/n; % loglikelihood
logR = bsxfun(@minus,logRho,T);
R = exp(logR);
```

Add log weights

- $\log(\pi_k \times p(x|\theta_k))$

Logsumexp trick

- $T = \log \sum_k \pi_k \times p(x|\theta_k)$

Compute average log-likelihood.
Calculate responsibilities $R(n, k)$ that represent the posterior probability of point n belonging to cluster k:

9 M-step Function

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

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

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

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

Compute effective number of points per cluster.
Update mixture weights.

- Weights: $\pi_k = \frac{n_k}{n}$

Update means μ_k:

- $\mu_k = \frac{1}{n_k} \sum_{i=1}^{n} r_{ik} x_i$
- r_{ik} is the responsibility of point i for cluster k

Update covariance matrices Σ_k:

- $\Sigma_k = \frac{1}{n_k} \sum_{i=1}^{n} r_{ik}(x_i - \mu_k)(x_i - \mu_k)^T + \epsilon I$
- Add small ϵ to ensure positive definiteness

Assign `mu`, `Sigma`, and `weight` to model struct.

10 Log Gaussian PDF function

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

Center data by subtracting mean, then compute the Cholesky decomposition of covariance matrix Σ.
Solve $U'Q = X$ efficiently
Compute quadratic term
Compute log normalization constant and final log probability.