

Problem 3: EM Algorithm

1 Main Function

```
1 function [label, model, llh] = emgm(X, init)
2 % Perform EM algorithm for fitting the Gaussian
   mixture model.
3 % X: d x n data matrix
4 % init: k (1 x 1) or label (1 x n, 1<=label(i)<=k
   ) or center (d x k)
5 % 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

```
6 %% initialization
7 fprintf('EM for Gaussian mixture: running ... \n'
   );
8 R = initialization(X,init);
9 [~,label(1,:)] = max(R,[],2);
10 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

```
11
12 tol = 1e-10;
13 maxiter = 500;
14 llh = -inf(1,maxiter);
15 converged = false;
16 t = 1;
```

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

4 Main EM Iteration

```
17 while ~converged && t < maxiter
18     t = t+1;
19     model = maximization(X,R);
20     [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

```
21
22     [~,label(:)] = max(R,[],2);
23     u = unique(label); % non-empty components
24     if size(R,2) ~= size(u,2)
25         R = R(:,u); % remove empty
components
26     else
27         converged = llh(t)-llh(t-1) < tol*abs(llh
(t));
28     end
29     figure(gcf); clf;
30     spread(X,label);
31     muA = model.mu;
32     SigmaA = model.Sigma;
33     wA = model.weight;
34     k = size(muA,2);
35     % figure(12); clf;
36     % for i=1:k
37     %     mu1 =muA(i,:)
38     %     Sigma1=SigmaA(i,:)
39     %     w1=wA(i)
40     %     xx= mvnrnd(mu1, Sigma1, 1000);
41     %     yy= mvnpdf(xx,mu1,Sigma1);
42     %     plot3(xx(:,1), xx(:,2), yy, '.b'); hold
on;
43     % end
44
45     pause;
46
47
48 end
49 llh = llh(2:t);
50 if converged
51     fprintf('Converged in %d steps.\n',t-1);
52 else
53     fprintf('Not converged in %d steps.\n',
maxiter);
54 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 μ , covariances Σ , and weights w .

Trim log-likelihood array.

6 Initialization Function

```
56 function R = initialization(X, init)
57 [d,n] = size(X);
58 if isstruct(init) % initialize with a model
59     R = expectation(X,init);
60 elseif length(init) == 1 % random initialization
61     k = init;
62     idx = randsample(n,k);
63     m = X(:,idx);
64     [~,label] = max(bsxfun(@minus,m'*X,dot(m,m,1)
65         '/2),[],1);
66     [u,~,label] = unique(label);
67     while k ~= length(u)
68         idx = randsample(n,k);
69         m = X(:,idx);
70         [~,label] = max(bsxfun(@minus,m'*X,dot(m,
71             m,1)/2),[],1);
72         [u,~,label] = unique(label);
73     end
74     R = full(sparse(1:n,label,1,n,k,n));
75 elseif size(init,1) == 1 && size(init,2) == n
76     label = init;
77     k = max(label);
78     R = full(sparse(1:n,label,1,n,k,n));
79 elseif size(init,1) == d %initialize with only
80     centers
81     k = size(init,2);
82     m = init;
83     [~,label] = max(bsxfun(@minus,m'*X,dot(m,m,1)
84         '/2),[],1);
85     R = full(sparse(1:n,label,1,n,k,n));
86 else
87     error('ERROR: init is not valid.');
```

7 E-step Function

```
87 function [R, llh] = expectation(X, model)
88 mu = model.mu;
89 Sigma = model.Sigma;
90 w = model.weight;
91
92 n = size(X,2);
93 k = size(mu,2);
94 logRho = zeros(n,k);
95
96 for i = 1:k
97     logRho(:,i) = loggausspdf(X,mu(:,i),Sigma
98         (:,:,i));
99 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

Extract model parameters

- μ : cluster means
- Σ : covariance matrices
- w : mixture weights

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

8 Calculate Responsibilities

```
99 logRho = bsxfun(@plus,logRho,log(w));
100 T = logsumexp(logRho,2);
101 llh = sum(T)/n; % loglikelihood
102 logR = bsxfun(@minus,logRho,T);
103 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

```
106 function model = maximization(X, R)
107 [d,n] = size(X);
108 k = size(R,2);
109
110 nk = sum(R,1);
111 w = nk/n;
112 mu = bsxfun(@times, X*R, 1./nk);
113
114 Sigma = zeros(d,d,k);
115 sqrtR = sqrt(R);
116 for i = 1:k
117     Xo = bsxfun(@minus,X,mu(:,i));
118     Xo = bsxfun(@times,Xo,sqrtR(:,i)');
119     Sigma(:, :, i) = Xo*Xo'/nk(i);
120     Sigma(:, :, i) = Sigma(:, :, i)+eye(d)*(1e-6);
121 end
122
123 model.mu = mu;
124 model.Sigma = Sigma;
125 model.weight = w;
```

Compute effective number of points per cluster.

Update mixture weights and means μ_k .

Update covariance matrices Σ_k :

Assign mu, Sigma, and weight to model struct.

10 Log Gaussian PDF function

```
127 function y = loggausspdf(X, mu, Sigma)
128 d = size(X,1);
129 X = bsxfun(@minus,X,mu);
130 [U,p]= chol(Sigma);
131 if p ~= 0
132     error('ERROR: Sigma is not PD. ');
133 end
134 Q = U'\X;
135 q = dot(Q,Q,1); % quadratic term
136 c = d*log(2*pi)+2*sum(log(diag(U)));
137 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.