

```

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).
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));
11
12 tol = 1e-10;
13 maxiter = 500;
14 llh = -inf(1,maxiter);
15 converged = false;
16 t = 1;
17 while ~converged && t < maxiter
18     t = t+1;
19     model = maximization(X,R);
20     [R, llh(t)] = expectation(X,model);
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
49 end
50 llh = llh(2:t);
51 if converged
52     fprintf('Converged in %d steps.\n',t-1);
53 else
54     fprintf('Not converged in %d steps.\n',maxiter);
55 end
56 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
57 function R = initialization(X, init)
58 [d,n] = size(X);
59 if isstruct(init) % initialize with a model
60     R = expectation(X,init);
61 elseif length(init) == 1 % random initialization
62     k = init;
63     idx = randsample(n,k);
64     m = X(:,idx);
65     [~,label] = max(bsxfun(@minus,m'*X,dot(m,m,1)'/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,m,1)'/2),[],1);
71         [u,~,label] = unique(label);
72     end
73     R = full(sparse(1:n,label,1,n,k,n));
74 elseif size(init,1) == 1 && size(init,2) == n % initialize with labels
75     label = init;
76     k = max(label);
77     R = full(sparse(1:n,label,1,n,k,n));
78 elseif size(init,1) == d %initialize with only centers
79     k = size(init,2);
80     m = init;
81     [~,label] = max(bsxfun(@minus,m'*X,dot(m,m,1)'/2),[],1);
82     R = full(sparse(1:n,label,1,n,k,n));
83 else
84     error('ERROR: init is not valid. ');
85 end
86
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(:, :, i));
98 end
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);
104
105 function model = maximization(X, R)
106 [d,n] = size(X);
107 k = size(R,2);
108
109 nk = sum(R,1);
110 w = nk/n;
111 mu = bsxfun(@times, X*R, 1./nk);
112
113 Sigma = zeros(d,d,k);
114 sqrtR = sqrt(R);
115 for i = 1:k
116     Xo = bsxfun(@minus,X,mu(:,i));
117     Xo = bsxfun(@times,Xo,sqrtR(:,i)');
118     Sigma(:, :, i) = Xo*Xo'./nk(i);
119     Sigma(:, :, i) = Sigma(:, :, i)+eye(d)*(1e-6); % add a prior for numerical stability
120 end
121
122 model.mu = mu;
123 model.Sigma = Sigma;
124 model.weight = w;
125
126 function y = loggausspdf(X, mu, Sigma)
127 d = size(X,1);
128 X = bsxfun(@minus,X,mu);
129 [U,p]= chol(Sigma);
130 if p ~= 0
131     error('ERROR: Sigma is not PD. ');

```

```
133 end
134 Q = U'\X;
135 q = dot(Q,Q,1); % quadratic term (M distance)
136 c = d*log(2*pi)+2*sum(log(diag(U))); % normalization constant
137 y = -(c+q)/2;
138
```