Abdulkadir Çelikkanat

Term Project

Introduction

The variational Bayes(VB) algorithm is derived for learining continuous hidden Markov models(CHMMs), examined on the generated data set, and the performances of VB and ML is compared on this synthetic data set for discrete HMM model. Unlike the ML, and MAP, which yield a single point estimate of the model parameters, VB algorithm yields an estimate of the full posteriour of the model parameters. Since it gives the measure of confidence in the accuracy of learned model, it is particularly significant for small sets.

Variational Bayes For Continuous Hidden Markov Models

Consider a N-state HMM model $\Phi = \{\pi, A, C, \theta\}$ where π is the initial state probability distribution, $\boldsymbol{A}$ is the transition matrix, $\boldsymbol{C}$ is the mixture coefficient matrix, and θ is the parameter matrix consisting of the Gaussian parameters $\theta_{ik} = \{\mu_{ik}, R_{ik}\}$ for the kth mixture component of the ith state, with mean μ_{ik} and precision R_{ik}.
Let Y=$\{y_1, y_2, ..., y_T\}$ be an observation sequence, X=$\{x_1, x_2, ..., x_T\}$ be the unobserved state sequence, and L=$\{l_1, l_2, ..., l_T\}$ be the indicator sequence, which shows which mixture component generates the observation y_i. Therefore, related complete-data is Z={X,Y,L}, and proability of Z can be written as

$$p(Z|\Phi) = p(X, Y, L|\Phi) = \pi_{x_1} \left[\prod_{t=1}^{T-1} a_{x_t x_{t+1}}\right] \left[\prod_{t=1}^{T} c_{x_t l_t} f(y_t|\theta_{x_t l_t})\right] \tag{1}$$

the likelihood of the model paremeters ϕ given the data $\boldsymbol{Y}$ is

$$p(Y|\Phi) = \sum_{X,L} \pi_{x_1} \left[\prod_{t=1}^{T-1} a_{x_t x_{t+1}}\right] \left[\prod_{t=1}^{T} c_{x_t l_t} f(y_t|\theta_{x_t l_t})\right] \tag{2}$$

and the marginal likelihood can be expressed as

$$p(Y) = \frac{p(X, Y, L, \phi)}{p(X, L, \phi|Y)} \tag{3}$$

Now, taking the logarithm of equation (3) above, the expectation with respect to the distribition q(X,L,ϕ), then rearranging it, we obtain

$$logp(Y) = F(q) + KL(q||p) \tag{4}$$

where the F(q) known as the negative free energy is

$$F(q) = \int q(X, L, \phi) log \frac{p(Y, X, L, \phi)}{q(X, L, \phi)} dX dL d\phi \tag{5}$$

and the Kullback-Leibler(KL) divergence between approximate and true posterior densities is

$$KL(q||p) = \int q(X, L, \phi) log \frac{q(X, L, \phi)}{p(X, L, \phi|Y)} dX dL d\phi \tag{6}$$

Since the KL divergence is nonnegative and 0 if p(x)=q(x) almost everywhere, F(q) is strict lower bound for logp(Y).
The main aim of the VB is to maximize the lower bound F(q) by tuning distribution q(X,L,ϕ) so that it converges the true posterior distribution. However, here two problems arises: the first one is choosing the suitable form of variational density which is tractable and will make good approximation to the true posterior $p(X, L, \phi|Y)$, the second problem is choosing the prior distributions of model parameters $\phi = \{\pi, A, C, \theta\}$
The factorized form

$$q(X, L, \phi) = q(X)q(L)q(\pi)q(A)q(C)q(\theta) \tag{7}$$

was preferred and has been successfully applied in many applications of variational methods [2]. Since the Dirichlet distribution is the conjugate prior of the multinomial distribution, the prior over π the rows of $\boldsymbol{A}$, and the rows of C was choosen as Dirichlet distribution, and Normal-Wishart distribution was choosen for the prior over θ because it is the conjugate prior of a multivariate normal distribution with unknown mean and precision. Hence, we have

$$q(\phi) = q(\pi)q(A)q(C)q(\theta) \tag{8}$$

where

$$\begin{aligned} p(\pi) &= Dir(\pi_1, ..., \pi_N | u_1^\pi, ..., u_N^\pi), \\ p(A) &= \prod_{i=1}^{N} Dir(a_{i1}, ..., a_{iN} | u_{i1}^A, ..., u_{iN}^A), \\ p(C) &= \prod_{i=1}^{N} Dir(c_{i1}, ..., c_{iN} | u_{i1}^C, ..., u_{iK}^C), \\ p(\theta) &= \prod_{i=1}^{N}\prod_{k=1}^{K} NW(\mu_{ik}, R_{ik} | a_{ik}, b_{ik}, \lambda_{ik}, m_{ik}). \end{aligned} \tag{9}$$

M-step

To maximize F(q), the variational posterior $q(X, L, \phi)$ is updated with respect to $q(\phi)$ on fixed hidden variables at q(X,L). If the equation (5) is subsituted with equations (8) and (9), we obtain

$$\begin{aligned} &F(q) = \int q(X)q(L)q(\pi)q(A)q(C)q(\theta)\Big[log\pi_{x_1} + \textstyle\sum_{t=1}^{T-1} loga_{x_t x_{t+1}} + \sum_{t=1}^{T} logc_{x_t l_t} + \sum_{t=1}^{T} logf(y_t|\theta_{x_t l_t}) \\ &+ logp(\pi) + logp(A) + logp(C) + logp(\theta) - logq(X) - logq(L) - logq(\pi) - logq(A) - logq(C) \\ &- logq(\theta)\Big] dX dL d\phi \\ &F(q) = F(q(A)) + F(q(C)) + F(q(\theta)) + const \end{aligned}$$

Optimization of q(A), $q(\pi)$ and q(C)

By collecting all the terms related to q(A) together, we can write

$$F(q(A)) = \int q(A) \sum_X q(X) \sum_{t=1}^{T-1} loga_{x_t x_{t+1}} dA + \int q(A) logp(A) dA - \int q(A) logq(A) dA \tag{10}$$

Then, we have

$$F(q(A)) = -\int q(A) log \left[\frac{q(A)}{\prod_{i,j=1}^{N} a_{ij}^{W_{ij}^A - 1}} \right] dA \tag{11}$$

where the hyperparameter $W_{ij}^A = \sum_{t=1}^{T-1} w_{ij}^t + u_{ij}^A$ and $w_{ij}^t = q(x_t = i, x_{t+1} = j)$. Then, F(q(A)) is maximized with respect to q(A) by Gibbs inequality, so

$$q(A) = \prod_{i=1}^{N} Dir(a_{i1}, ..., a_{iN} | W_{i1}^A, ..., W_{iN}^A) \tag{12}$$

Similarly, F(q) can be optimized by using similar operations, and acquire the optimal distributions

$$\begin{aligned} q(\pi) &= Dir(\pi_1, ..., \pi_N | W_1^\pi, ..., W_N^\pi) \\ q(C) &= \prod_{i=1}^{N} Dir(c_{i1}, ..., c_{iK} | W_{i1}^C, ..., W_{iK}^C) \end{aligned} \tag{13}$$

where $W_i^\pi = w_i^\pi + u_i^\pi$, $w_i^\pi = q(x_1 = i)$, $W_{ik}^C = \sum_{t=1}^{T} w_{ik}^t + u_{ik}^C$, and $w_{ik}^t = q(x_t = i, l_t = k)$

Optimization of $q(\theta)$

By collecting all the terms related to $q(\theta)$ together, we can write

$$F(q(\theta)) = \int q(\theta) \sum_{X,L} q(X)q(L) \sum_{t=1}^{T} logf(y_t|\theta_{x_t l_{x_t}})d\theta + \int q(\theta)logp(\theta)d\theta - \int q(\theta)logq(\theta)d\theta \quad (14)$$

Then we obtain

$$F(q(\theta)) = -\int q(\theta)log\left[\frac{q(\theta)}{\prod_{i=1}^{N}\prod_{k=1}^{K}\left[\prod_{t=1}^{T} f^{w_{ik}^t}(y_t|\theta_{ik})p(\theta_{ik})\right]}\right]d\theta \quad (15)$$

Then the optimal distribution $q(\theta_{ik})$ is obtained as

$$q(\theta_{ik}) = \frac{(\lambda_{ik}/2\pi)^{d/2}}{Z(a_{ik},b_{ik})(2\pi)^{dw_{ik}/2}}|R_{ik}|^{\frac{a_{ik}+w_{ik}-d}{2}} exp\left[-(\lambda_{ik}^{'}/2)(\mu_{ik}-m_{ik}^{'})^T R_{ik}(\mu_{ik}-m_{ik}^{'}) - \frac{1}{2}Tr(b_{ik}^{'}R_{ik})\right] \quad (16)$$

where

$$w_{ik} = \sum_{t=1}^{T} w_{ik}^t,\ x_{ik} = \sum_{t=1}^{T} w_{ik}^t x_t/w_{ik},\ S_{ik} = \sum_{t=1}^{T} w_{ik}^t(x_t - x_{ik})(x_t - x_{ik})^T,\ a_{ik}^{'} = a_{ik} + w_{ik},$$

$$b_{ik}^{'} = b_{ik} + S_{ik} + \frac{\lambda_{ik}w_{ik}}{\lambda_{ik}+w_{ik}}(m_{ik}-x_{ik})(m_{ik}-x_{ik})^T,\ \lambda_{ik}^{i} = \lambda_{ik} + w_{ik},\ m_{ik}^{'} = \frac{\lambda_{ik}m_{ik} + w_{ik}x_{ik}}{\lambda_{ik}+w_{ik}}$$

E-step

To maximize F(q), the variational posterior $q(X, L, \phi)$ on hidden variables $q(X, L)$ is updated on fixed model parameters $q(\phi)$. If the equation (5) is subsituted with equations (8) and (9),and then rearranged, we can get the equation

$$F(q) = F(q(X,L)) - KL(q(\phi)||p(\phi)) \quad (17)$$

where

$$\begin{aligned} F(q(X,L)) = &\sum_{X} q(X) \int q(\pi) log\pi_{x_1} d\pi + \sum_{X} q(X) \int q(A) \sum_{t=1}^{T-1} log(a_{x_t x_{t+1}})dA \\ &+ \sum_{X,L} q(X,L) \int q(C) \sum_{t=1}^{T} log(c_{x_t l_t})dC \\ &+ \sum_{X,L} q(X,L) \int q(\theta) \sum_{t=1}^{T} logf(y_t|\theta_{x_t l_t})d\theta - \sum_{X,L} logq(X,L), \end{aligned} \quad (18)$$

Since the second term $KL(q(\phi)||p(\phi))$ in equation (17) is constant, only the first term $F(q(X, L))$ need to be optimized.
Now, defining

$$log\pi_{x_1}^{*} = \int q(\pi) log\pi_{x_1} d\pi = \psi(W_{x_1}^{\pi}) - \psi(W_0^{\pi}) \quad (19)$$

$$loga_{x_t x_{t+1}}^{*} = \int q(A) loga_{x_t x_{t+1}} dA = \psi(W_{x_t x_{t+1}}^{A}) - \psi(W_{x_t 0}^{A}) \quad (20)$$

$$logc_{x_t l_t}^{*} = \int q(C) logc_{x_t l_t} dC = \psi(W_{x_t l_t}^{C}) - \psi(W_{x_t 0}^{C}) \quad (21)$$

$$\begin{aligned} logf^{*}(y_t|\theta_{x_t|l_t}) &= \int q(\theta) logf(y_t|\theta_{x_t l_t})d\theta \\ &= \frac{-d}{2}log2\pi - \frac{1}{2}log\left|\frac{b_{x_t l_t}}{2}\right| + \frac{1}{2}\sum_{i=1}^{d}\psi\left(\frac{a_{x_t l_t}+1-i}{2}\right) \\ &\quad - \frac{1}{2}a_{x_t l_t}(x_t - m_{x_t l_t})^T b_{x_t l_t}^{-1}(x_t - m_{y_t l_t}) - \frac{d}{2\lambda_{y_t l_t}} \end{aligned} \quad (22)$$

where $\psi(.)$ is the digamma function and the parameters $W_0^{\pi} = \sum_{i=1}^{i=N} W_i^{\pi}$, $W_{x_t 0}^{A} = \sum_{i=1}^{i=N} W_{x_t i}^{A}$, and $W_{x_t 0}^{C} = \sum_{i=1}^{i=N} W_{x_t i}^{C}$. Then, after subsituting (19)-(22) into (18), we obtain

$$F(q(X,L)) = \sum_{X,L} q(X,L) log \frac{\pi^*_{x_1} \prod_{t=1}^{T-1} a_{x_t} a_{x_t+1} \prod_{t=1}^{T} c^*_{x_t l_t} f^*(y_t|\theta_{x_t l_t})}{q(X,L)} \tag{23}$$

so the optimized

$$q(X,L) = \frac{1}{Z} \left[\pi^*_{x_1} \prod_{t=1}^{T} a^*_{x_t x_{t+1}} \prod_{t=1}^{T} c^*_{x_t l_t} f^*(y_t|\theta_{x_t l_t}) \right] \tag{24}$$

It can be noticed that $Z = q(X|\phi^*)$, after comparing it with 2. It is the approximate likelihood of the optimized mode ϕ^*, which can be computed efficiently by the forward backward algorithm.

Convergence

The variational Bayes approach is a generalization of the conventional EM algorithm [2]. Each iteration increases the negative free energy $F(q)$, or it remains as unchanged. The negative free energy is of significance to approximate the marginal likelihood, so the algorithm is terminated when the change in $F(q)$ is negligibly small, and this quantity can be computed by using the equations (17) and (23)-(24) in the following way

$$\begin{aligned} F(q) &= F(q(X,L)) - KL(q(\phi)||p(\phi)) \\ &= logq(Y||\phi^*) - KL_{Dir}(q(\pi)||p(\pi)) - KL_Dir(q(A)||p(A)) - KL_{Dir}(q(C)||p(C)) \\ &- KL_{NW}(q(\theta)||p(\theta)) \end{aligned} \tag{25}$$

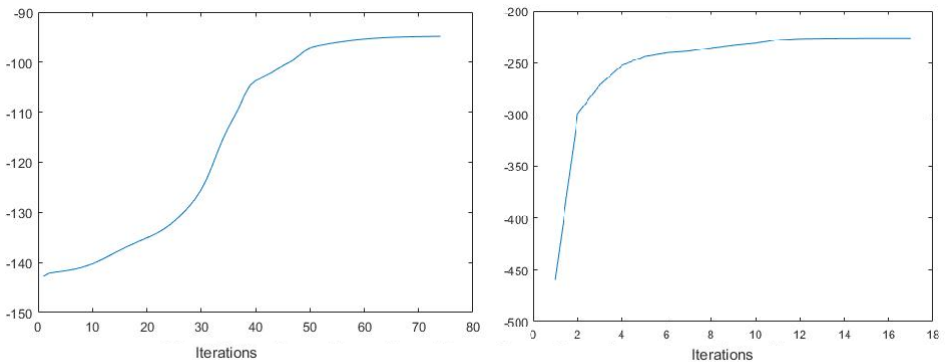
Application

For an experiment, a synthetic data consisting of four different letter $\{a, b, c, d\}$ sequences has been generated. The data set contains 19 letter sequences and the longest sequence contains 426 letters, smallest contains 14 letters. Some instances of the generated data set as follows:

$$\begin{aligned} y_{1:T_1} &= (bbcccbdcdbbbcc...acdbcacddabcbbbbcadab) \\ y_{1:T_2} &= (cddadcdbaacabd...aacadbbadacbdccbbddac) \\ &... \\ &... \\ y_{1:T_{18}} &= (ccdabccbbdabdb...cbcaccaddccccbdbbdbbbcb) \\ y_{1:T_{19}} &= (bdaadbbbbacabadbdbcb...cadadaaddaaccd) \end{aligned}$$

In the figures below, the performances of ML and VB algorithms are compared over this generated data set. It is also assumed that the number of hidden states of HMM is 15.

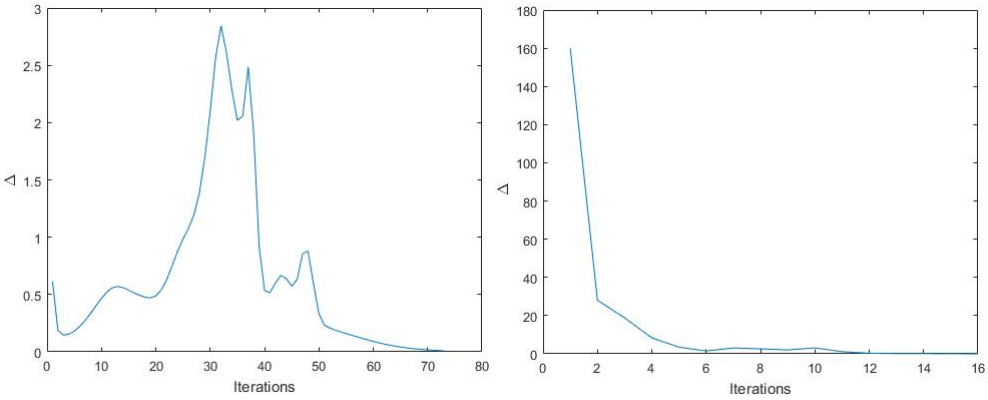
Figure 1:



(a) ML: plot of the log likelihood of the data $p(Y|\Phi)$ (b) VB: plot of the lower bound $F(q)$

The figures show the likelihood of the data under the ML algorithm and the lower bound on the marginal likelihood under the VB algorithm. As shown in figures, it takes ML about 70 iterations to converge, and takes rougly 12 iterations for VB.

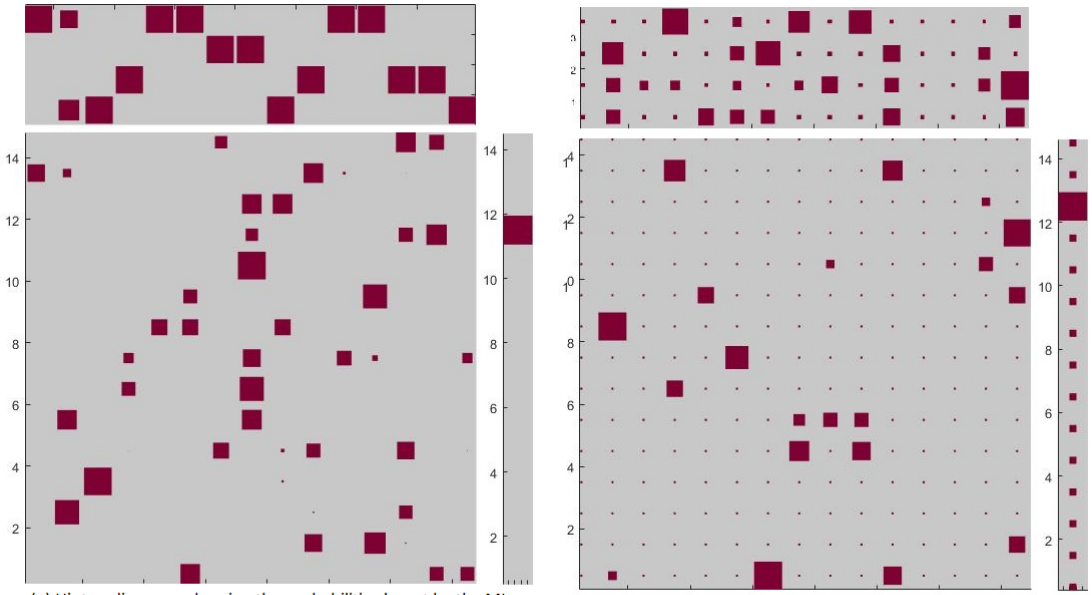
Figure 2: Derivatives



(a) ML: plot of the derivative of log likelihood of the data $p(Y|\Phi)$ (b) VB: plot of the derivative of lower bound $F(q)$

The derivatives of the functions given in the previous figures for ML and VB algorithms.

Figure 3: Hinton Diagrams



(a) Hinton diagrams showing the probabilities learnt by the ML model, for initial state prior **π**, transition matrix **A**, and emission matrix **C**

(b) Hinton diagrams for the variational parameters describing the posterior distributions over the parameters q(π), q(A), and q(C) respectively

The hinton diagram in the left side shows the probabilities learnt by ML, for initial state prior π, transition matrix A, and emission matrix C. In the right side, hinton diagram for the variational parameters describing the posterior distributions over $q(\pi)$, $q(A)$, and $q(C)$, respectively.

References

[1] S. Ji , B. Krishnapuram and L. Carin , *Variational Bayes for continuous hidden Markov models and its application to active learning* , IEEE Trans. Pattern Anal. Mach. Intell. vol. 28 , no. 4 , pp.522 -532, 2006

[2] M. J. Beal, *Variational algorithms for approximate Bayesian inference*, PhD thesis, Gatsby Computational Neuroscience Unit, University College, London, 2003.

Implementation Codes

```
1  function main()
2
3      file = load('data seq.mat');
```

```
4       data = file.data;
5       numOfSeqs = length(data);
6
7           data = symbol2numeric(data,'abcd');
8
9       N = 15; L = 4;
10
11      [F,predict] = vbhmm(data,N,L,100,0.0001);
12
13      figure
14      grid;
15      plot(1:length(F),F,' ')
16
17      figure,
18      grid;
19      plot(diff(F),' ');
20      xlabel('Iterations'), ylabel('\Delta F')
21      title('\Delta')
22
23   %  hinton(predict.WA), hinton(predict.WB'), hinton(predict.WPi')
24
25
26  end
27
28
29  function [F, predict] = vbhmm(data, N, L, maxIter, epsilon)
30  % Input parameters:
31  %   N > number of states
32  %   L > number of observations
33
34      numOfSeqs = size(data,2);
35
36      totalSeqLength = 0;
37      for i=1:numOfSeqs
38          totalSeqLength = totalSeqLength + size(data{i},2);
39      end
40
41      % Initialise the pseudo counts
42      uA = ones(1,N)*(1/N);
43      uB = ones(1,L)*(1/L);
44      uPi = ones(1,N)*(1/N);
45      % Pick an HMM from the prior to initialize the counts
46      sum_wa = zeros(N,N); sum_wb = zeros(N,L);
47      for n=1:N, % loop over hidden states
48          sum_wa(n,:) = (dirichlet(uA',1))*totalSeqLength;
49          sum_wb(n,:) = (dirichlet(uB',1))*totalSeqLength;
50      end;
51      w_pi = (dirichlet(uPi',1))*numOfSeqs;
52
53      A = zeros(N,N); B = zeros(N,L); Pi = zeros(N,1);
54      for i=1:maxIter
55
56          WA = sum_wa + repmat(uA,N,1);
57          WB = sum_wb + repmat(uB,N,1);
58          WPi = w_pi + uPi;
59
60          A = exp(   psi(WA)    repmat( psi(sum(WA,2)) ,[1 N])   );
61          B = exp(   psi(WB)    repmat( psi(sum(WB,2)) ,[1 L])   );
62          Pi = exp(   psi(WPi)    psi(sum(WPi,2))   );
63
64
```

```
65          % E Step
66          [alpha, beta, gamma, xsi, gammak, a_estt, gammaInit, logZ] =
                forwardBackward(A', B, Pi', data);
67          sum_wa = a_estt;
68          sum_wb = gammak;
69          w_pi = gammaInit;
70         % [sum_wA, sum_wB, w_Pi, logZ(i), lnZv] = forwback(A,B,Pi,data)
71
72          % Compute F, straight after E Step.
73          Fa(i)=0; Fb(i)=0; Fpi(i)=0;
74          for kk = 1:N,
75              Fa(i) = Fa(i)    KL_Dirichlet(WA(kk,:),uA);
76              Fb(i) = Fb(i)    KL_Dirichlet(WB(kk,:),uB);
77          end;
78          Fpi(i) =    KL_Dirichlet(WPi,uPi);
79
80          F(i) = logZ + Fa(i)+Fb(i)+Fpi(i);
81
82          if i>2
83              %if( (F(i)   F(i 1)) < epsilon )
84              if( (F(i)   F(2)) < (1+epsilon)*(F(i 1) F(2)))
85                  fprintf('Converged at %d th iteration.\n',i);
86                  break;
87              end
88          end
89      end
90
91      predict = struct('A', A, 'B', B, 'Pi', Pi, 'WA',WA, 'WB',WB, 'WPi',WPi
            );
92
93  end
94
95  function [result] = KL_Dirichlet(alpha, beta)
96  % Input parameters :
97  % alpha(1xN), and beta(1xN) are paramaeters of Dirichlet distribution
98
99  result = gammaln(sum(alpha)) gammaln(sum(beta))    sum(gammaln(alpha)
        gammaln(beta)) ...
100             + (alpha beta) * (psi(alpha) psi(sum(alpha)))' ;
101
102 end
103
104
105 % Generates samples from Drichlet distribution
106 function [x] = dirichlet(alpha, N)
107 % Paramaters :
108 %      alpha  > Nx1 matrix,
109
110     L = size(alpha,1);
111     x = gamrnd(repmat(alpha',N,1),1,N,L);
112     x = bsxfun(@rdivide, x, sum(x,2));
113
114 end
```

```
1   function [alpha, beta, gamma, xsi ,B_est, A_est, Pi_est,logZ] =
        forwardBackward(A, B, Pi, dataSeq)
2   % Assumption dataSeq consisting of positive integers
3   %      A is NxN, B is NxL, Pi is Nx1
4
5   xsi = zeros(size(A));
6   B_est = zeros(size(B));
```

```
7  A_est = zeros(size(A));
8  Pi_est = zeros(size(Pi));
9      for inx=1:size(dataSeq,2)
10
11         obs = dataSeq{inx};
12
13         T = size(obs,2);
14         [N, L] = size(B);
15
16         % Compute alpha
17         alpha(:,1) = Pi .* B(:,obs(1,1));
18         scale(1) = sum(alpha(:,1));
19         alpha(:,1) = alpha(:,1) / scale(1);
20         for t=2:T
21             alpha(:,t) = (A * alpha(:,t-1)) .* B(:,obs(1,t));
22             scale(t) = sum(alpha(:,t));
23             alpha(:,t) = alpha(:,t) / scale(t);
24         end
25
26         % Compute beta
27         beta(:,T) = ones(N,1) / scale(T);
28         for t=T-1:-1:1
29             beta(:,t) = A' * ( beta(:,t+1) .* B(:,obs(1,t+1)) );
30             beta(:,t) = beta(:,t) / scale(t);
31         end
32
33         % Compute gamma
34         gamma = alpha .* beta;
35         gamma = bsxfun(@rdivide, gamma, sum(gamma,1));
36
37         % Compute xsi
38         for t=1:T-1
39             xsi = xsi + repmat(alpha(:,t)',N,1) .* A .* repmat(B(:,obs
                   (:,t+1)),1,N) .* repmat(beta(:,t+1),1,N);
40         end
41
42
43         % Compute the sums of Gamma conditioned on k
44         for t = 1:T
45                 B_est(:,obs(t)) = B_est(:,obs(t)) + gamma(:,t);
46         end;
47
48
49         Pi_est = Pi_est + gamma(:,1);
50
51         logZsums(inx) = sum(log(scale));
52
53     end
54     A_est = xsi';
55     Pi_est = Pi_est';
56
57     logZ = sum(logZsums);
58 end
```