Bayesian Approach of Hidden Markov Model

Stats 236 Paper

Jiatong Chen

Department of Materials Science and Engineering, UCLA

Abstract

Hidden Markov Models (HMM) have been widely used in diverse fields during the last twenty years. Inference in HMM is traditionally carried out using the EM algorithm. But Bayesian estimation, in general implemented by Markov Chain Monte Carlo (MCMC) sampling, also appears quite often in HMM literature. In this paper, I will brief review the learning and inference of HMM from Bayesian perspective and demonstrate it by numerical simulation.

I. INTRODUCTION

Hidden Markov Models are a class of statistics models that have been standard in applied statistics nowadays. HMM provide a handy extension of i.i.d. mixture models and therefore allow for an analysis of phenomena that arise from dependent clusters or sequential procedure. For a usual mixture model, the observations are drawn independently from the distribution with density,

$$p(y|\theta) = \sum_{k=1}^{M} \lambda_m f(y|\theta_m) \tag{1}$$

where $f(\cdot|\theta_m)$ is a given distribution family. All weights λ_m sum up to one. The parameters are $(\lambda_1, ..., \lambda_M, \theta_1, ..., \theta_M)$. We further introduce hidden variables $X = (x_{im})$ to indicate which cluster the i-th individual y_i comes from.

$$x_{im} = \begin{cases} 1, & if\ y_i\ from\ f(\cdot|\theta_m) \\ 0, & otherwise \end{cases} \tag{2}$$

All $x_{i\cdot} = X_i$ are sampled i.i.d. from a multinomial distribution $\mathcal{M}(1, (\lambda_1, ..., \lambda_M))$. However, in Hidden Markov Model, independent assumption is removed. All hidden variables X_i are correlated so that they constitute a Markov chain with state space $X_i \in \{1, \cdots, M\}$ and transition matrix $A = (a_{ij})$. Suppose we consider a time homogenous Markov Chain,

$$a_{ij} = P(X_t = j|X_{t-1} = i) = P(X_2 = j|X_1 = i) \tag{3}$$

If we denote probability of initial state as parameters, i.e. $P(X_1 = i_1) = \rho_{1i_1}$, the Joint probability of hidden variables becomes,

$$\begin{aligned} P(X_1 = i_1, \cdots, X_n = i_n) =& P(X_1 = i_1)P(X_2 = i_2|X_1 = i_1) \\ & \cdots P(X_n = i_n|X_{n-1} = i_{n-1}) \\ =& \rho_{1i_1} \prod_{t=2}^{n} a_{i_{t-1}i_t} \end{aligned} \tag{4}$$

What's more, given all the hidden variables $\{X_i\}$, the conditional distribution of Y_k depends on X_k, but not on any other hidden variables. This is called emission probability $P(Y_k = y|X_k = m) = f(y|\theta_m)$. Therefore, the joint probability of hidden variables and

observed variables are,

$$\begin{aligned} & P(X_1, \cdots, X_n, Y_1, \cdots, Y_n | A, \{\theta_m\}_{m=1}^M) \\ & \quad = P(X_1) P(Y_1|X_1) \prod_{t=2}^{n} P(X_t|X_{t-1}) P(Y_t|X_t) \\ & \quad = \rho_{1i_1} f(Y_1|\theta_{X_1}) \prod_{t=2}^{n} a_{i_{t-1}i_t} f(Y_t|\theta_{X_t}) \end{aligned} \tag{5}$$

The goal is to estimate the parameters $A, \{\theta_m\}_{m=1}^M$ as well as make inference on the hidden variables given observations $\{Y_1, \cdots, Y_n\}$.

The first papers on HMM appeared in the second half of the 1960's (Baum, Petrie and co-workers), mostly dealing with the case when the observed variables Y_k take value from a finite set as well. In a major paper (Baum et al. 1970), a particular version of EM algorithm was implemented for the particular case of HMMs. Since then MLE and EM have been the main theme for inference in HMMs. Then in the early 1990's, Bayesian inference in HMMs, often implemented with Gibbs sampler, has gain more and more interest, especially a paper by Robert, Celeux and Diebolt in 1993. The purpose of this paper is to discuss and perform numerical analysis on an improved version of this approach.

II. BAYESIAN INFERENCE IN HMM

For simplicity and without loss of generality we assume the emission probability is normal distribution, i.e. $f(y|\theta_m) = \phi(y|\mu_m, \sigma^2)$. The most popular method for sampling from the posterior distribution of the parameters is the data augmentation, where we actually sample between model parameters and hidden data from their respective full conditional distributions. This is because given the hidden Markov chain and the data, the parameters are conditionally independent. Their probability distributions are from well-established parametric families as long as we assign conjugate priors for them. On the other hand, given the parameters and the observed data, the hidden Markov chain is a non-homogeneous Markov chain and hence easy to sample.

Now we assign conjugate priors for all model parameters. The hidden variables X_i are all categorical variables. $P(X_k|X_{k-1} = i_{k-1})$ follows a multinomial distribution. Therefore, each row of the transition probability matrix and the initial distribution $(P(X_1 = 1), P(X_1 = 2), \cdots, P(X_n = M)) = (\rho_{11}, \rho_{12}, \cdots, \rho_{1M})$ were given an independent Dirichlet distribution

prior $Dir(1, 1, \cdots, 1)$, each μ_i was given an independent Normal prior $\mathcal{N}(\xi, \kappa^{-1})$ with $\xi = (min(y_k) + max(y_k))/2$ and $\kappa = 1/R^2$ where $R = max(y_k) - min(y_k)$. σ^2 was sampled from an $Inv - \Gamma(\alpha, \beta)$ with $\alpha = 2$ and the hyper-parameter β was given a hyper prior $\Gamma(g, h)$, with $g = 0.2$ and $h = 10/R^2$.

Under this prior specification, the full conditional distributions are given by the following set of equations,

$$(\rho_{11}, \rho_{12}, \cdots, \rho_{1M})|\cdots \sim Dir(I\{X_1 = 1\} + 1, I\{X_1 = 2\} + 1, \cdots, I\{X_1 = M\} + 1) \quad (6)$$

where $I\{\cdot\}$ denotes an indicator function.

Each row of transition matrix A is conditionally independent across $i = 1, \cdots, M$,

$$(a_{i1}, a_{i2}, \cdots, a_{iM})|\cdots \sim Dir(n_{i1} + 1, n_{i2} + 1, \cdots, n_{iM} + 1) \quad (7)$$

where $n_{ij} = \#\{1<k \leqslant n : X_{k-1} = i, X_k = j\}$ is the number of transitions from state i to j in the hidden state sequence.

All μ_i are conditionally independent across $i = 1, \cdots, M$,

$$\mu_i|\cdots \sim \mathcal{N}(\frac{S_i + \kappa\xi\sigma^2}{n_i + \kappa\sigma^2}, \frac{\sigma^2}{n_i + \kappa\sigma^2}) \quad (8)$$

where $S_i = \sum_{k:X_k=i} y_k$, and $n_i = \#\{1 \leqslant k \leqslant n : X_k = i\}$ is the number of visits to state i in the hidden state sequence.

Posterior for σ^2,

$$\sigma^2|\cdots \sim Inv - \Gamma(\alpha + \frac{1}{2}n, \beta + \frac{1}{2}\sum_{k=1}^{n}(y_k - \mu_{X_k})^2) \quad (9)$$

and,

$$\beta|\cdots \sim \Gamma(g + \alpha, h + \sigma^{-2}) \quad (10)$$

Here, in all cases " $\cdots$ " denotes other parameters, the hidden data and the observed data.

However, unlike A and $\{\{\mu_i\}_{i=1}^M, \sigma^2\}$, the hidden variables are not independent given the parameters and observed data. The size of the chain can be fairly large. An obvious way to sample X_k is by Direct Gibbs Sampling (Robert, Celeux and Diebolt 1993), which draws each X_k from its full conditional distribution,

$$\begin{aligned} P(X_k = j|X_{-k}, Y_1, \cdots, Y_n, A, \{\mu_i\}_{i=1}^M, \sigma^2) \\ &\propto P(X_{k-1}|X_k = j)P(X_{k+1}|X_k = j)P(Y_k|X_k = j)P(X_k = j) \\ &= P(X_k = j, X_{k-1})P(X_{k+1}|X_k = j)f(Y_k|\theta_j) \\ &\propto P(X_k = j|X_{k-1})P(X_{k+1}|X_k = j)f(Y_k|\theta_j) \end{aligned} \quad (11)$$

for $k = 1, \cdots, n$, with appropriate adjustments for end effect. This Direct Gibbs Sampling procedure produces a sequence $\{(A, \{\mu_i\}_{i=1}^M, \sigma^2, \{X_k\}_{k=1}^n)^{(j)} : j = 1, 2, \cdots\}$ from a Markov chain and thereafter we can make inference upon this.

Notice that given the parameters and observed data, the hidden chain becomes a non-homogenous Markov chain. Therefore, a modification to the Direct Gibbs Sampling will be a stochastic version of the forward-backward recursion (Chib 1996; Scott 1999), where X_1 is sampled from initial distribution,

$$P(X_1 = j|\cdots) \propto \rho_{1j}\phi(y_1|\mu_j, \sigma^2)P_\theta(y_{2:n}|X_1 = j) \tag{12}$$

and $X_k, k = 2, \cdots, n$ are sampled from transition probabilities sequentially,

$$P(X_k = j|X_{k-1} = i) \propto a_{ij}\phi(y_k|\mu_j, \sigma^2)P_\theta(y_{k+1:n}|X_k = j) \tag{13}$$

where ϕ is the density of a Normal distribution with the indicated mean and variance. Given that the relations are only up to proportionality in j, they need to be normalized in order to be the correct probabilities. We note that the densities of the partial data sequences $P_\theta(y_{k+1:n}|X_k = j)$ (we denote θ as all parameters here) are nothing but the backward variables which can be calculated via backward recursion.

If we denote the backward variables as $B_k(i) = P_\theta(y_{k+1}, \cdots, y_n|X_k = i)$, then,

$$\begin{aligned} B_k(i) =& P_\theta(y_{k+1}, \cdots, y_n|X_k = i) \\ =& \sum_{j=1}^M P_\theta(y_{k+1}, \cdots, y_n, X_{k+1} = j|X_k = i) \\ =& \sum_{j=1}^M P(X_{k+1} = j|X_k = i)P(y_{k+1}|X_{k+1} = j)P_\theta(y_{k+2}, \cdots, y_n|X_{k+1} = j) \\ =& \sum_{j=1}^M a_{ij}f(y_{k+1}|\theta_j)B_{k+1}(j) \end{aligned} \tag{14}$$

The backward recursive summation becomes

- Initialize $B_n(i) = P(y_{n+1}|X_n = i) = 1$, $i = 1, \cdots, M$
- Get consistent recursion $B_k(i) = \sum_{j=1}^M a_{ij}f(y_{k+1}|\theta_j)B_{k+1}(j)$ for $k = n-1, n-2, \cdots, 1$

This backward recursion forward sampling method leads to a more rapidly mixing algorithm because fewer components are introduced into the Gibbs Markov chain, and hence is

considered to be more efficient. In practice, these backward variables tend to zero really fast in the recursions. Therefore, we must normalize the backward variables to sum to one over $j = 1, \cdots, M$.

The full Gibbs sampler then amounts to alternating between updating the parameters conditional on the data and hidden Markov chain, and updating the hidden Markov chain conditional on the data and parameters. So in t-th iteration,

- Update $(\mu_1, \mu_2, \cdots, \mu_M)$ by drawing independently from (8).
- Update σ^2 by drawing from (9).
- Update β by drawing from (10).
- For $i = 1, \cdots, M$, update $(a_{i1}, a_{i2}, \cdots, a_{iM})$ by drawing from (7).
- Update $(\rho_{11}, \rho_{12}, \cdots, \rho_{1M})$ by drawing from (6).
- Update $\{X_k\}_{k=1}^n$ by drawing X_1 from (12) and $\{X_k\}_{k=2}^n$ from (13) respectively.

III. COMPUTATIONAL RESULT

I choose number of states to be $M = 3$ and the transition matrix to be,

$$A = (a_{ij}) = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{2}{3} & \frac{1}{3} \\ \frac{2}{3} & 0 & \frac{1}{3} \end{bmatrix} \tag{15}$$

The observations are generated from a conditional normal $Y_k|X_k \sim \mathcal{N}(\mu_i, \sigma^2)$ with $\mu_1 = -2$, $\mu_2 = 0$ and $\mu_3 = 2$. I further assume $\sigma^2 = 0.25$. Starting from an initial guess, we end up having samples of all parameters and hidden variables. The posterior samples of all μ_i are shown in Fig. 1. The posterior samples of all σ^2 are shown in Fig. 2. As we can see, the first 300 steps can be discarded as burn-in iterations. Again I sample up to 10000 iterations. Then only calculate the posterior mean using samples from step 300 to step 10000. The final posterior mean estimates are,

$$\hat{A} = (\hat{a}_{ij}) = \begin{bmatrix} 0.357 & 0.289 & 0.353 \\ 0.001 & 0.645 & 0.354 \\ 0.683 & 0.005 & 0.312 \end{bmatrix} \tag{16}$$

$$\hat{\mu} = \begin{bmatrix} \hat{\mu_1} \\ \hat{\mu_2} \\ \hat{\mu_3} \end{bmatrix} = \begin{bmatrix} -2.009 \\ 0.0229 \\ 2.035 \end{bmatrix} \tag{17}$$

$$\hat{\sigma^2} = 0.243 \tag{18}$$

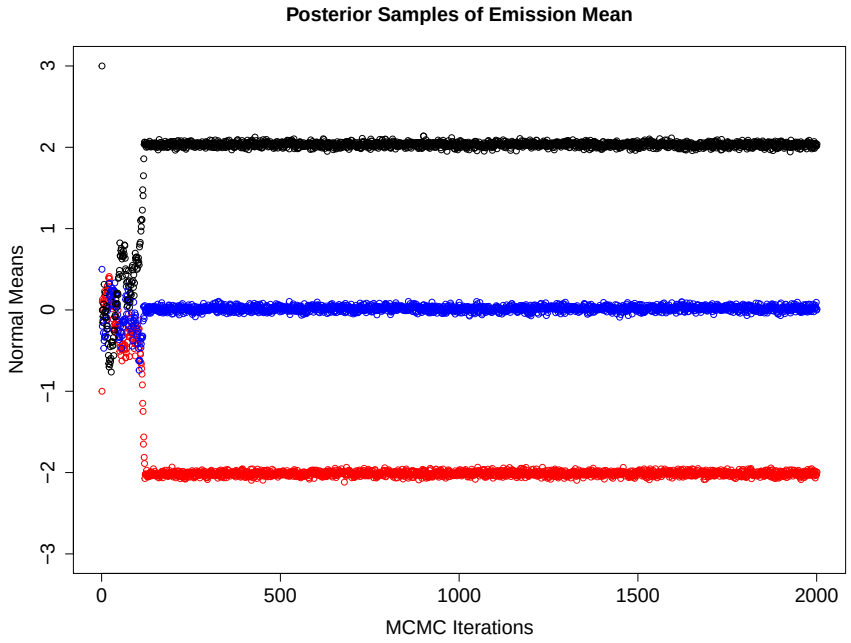

FIG. 1: Sequence of $\mu_i, i = 1, \ldots, 3$ from Posterior Dist.

which are quite accurate compared to true values.

For Hidden states X_k, I use posterior majority vote from iteration 300 to 10000, then evaluate the accuracy by,

$$Accuracy = \frac{\#(X_i^{pred} == X_i^{true})}{\#(X_i)} \tag{19}$$

I finally get the accuracy of the averaged chain to be 99.1%. If I only use the last chain, it is 98.8%. So model averaging does improve the performance.

IV. APPENDIX: R CODE

Below is my R code.

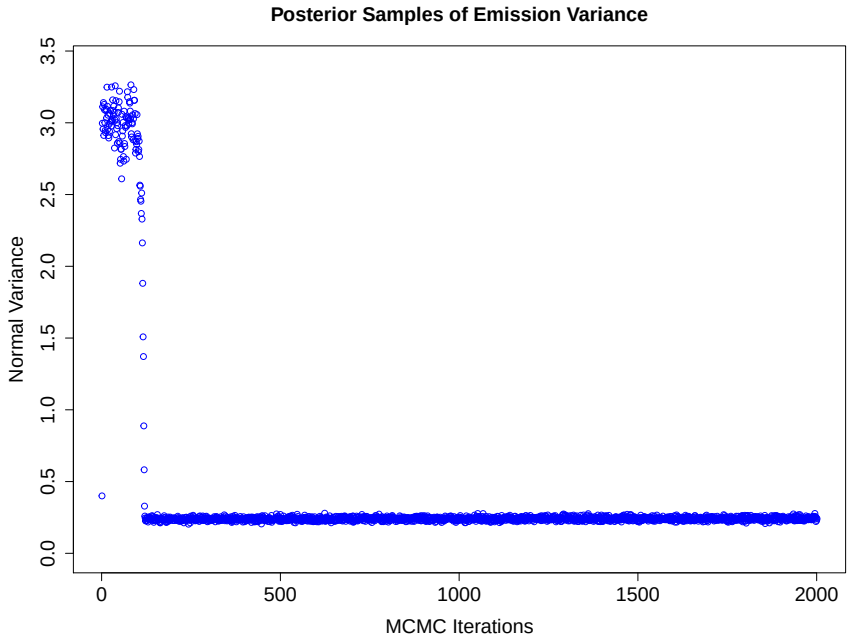

FIG. 2: Sequence of σ^2 from Posterior Dist.

```
################### setup ####################
set.seed(100)
library(DirichletReg)
library(invgamma)

######### generate simulated data ###########
## HMM generating function
HMM_generate <- function(trans,size,initial_state,sigma,mu){
  hidden_var = initial_state
  obs_var = rnorm(1, mean = mu[initial_state], sd = sigma)
  state = initial_state
  # Make n-1 steps through the markov chain
  for (i in 1:(size-1)){
    p = 0
    u = runif(1, 0, 1)
    #cat("> Dist:", paste(round(c(trans[state,]), 2)), "\n")
    #cat("> Prob:", u, "\n")
```

```
    # generate a new state from previous one
    for (j in 1:ncol(trans)){
      p = p + trans[state, j]
      if (p >= u){
        newState = j
        break
      }
    }
    #cat("*", state, "->", newState, "\n")
    hidden_var = c(hidden_var,newState)
    #make a emission from this new state
    obs = rnorm(1, mean = mu[newState], sd = sigma)
    obs_var = c(obs_var,obs)
    state = newState
  }
  return(list(hidden_var=hidden_var,obs_var=obs_var))
}

n = 1000 # size of the Markov chain
trans = matrix(c(1/3,1/3,1/3, # The probability transition
   matrix
                 0,2/3,1/3,
                 2/3,0,1/3), ncol=3, byrow=TRUE)
state = ceiling(3 * runif(1, 0, 1)) # initial prob for the 3
   states are same here
cat("Starting state:", state, "\n")
# normal obs
sigma0 = 0.5
mu0 = c(-2,0,2)
list = HMM_generate(trans,n,state,sigma0,mu0)
hidden_var = list$hidden_var
```

```
obs_var = list$obs_var



############ sampling parameters ############
## set up the prior
xi = (min(obs_var)+max(obs_var))/2
R = max(obs_var)-min(obs_var)
alpha = 2
kapa = 1/(R^2)
g = 0.2
h = 10/(R^2)

## parameters
niters = 10000 # number of iterations
M = 3 # state space
mu= mat.or.vec(niters+1,M)
sigsq = mat.or.vec(niters+1,1)
beta = mat.or.vec(niters+1,1)
trans_est = mat.or.vec(niters+1,M^2) # each row represent the
   matrix ordered by row
rho = mat.or.vec(niters+1,M) # initial probability

# hidden chain
X = mat.or.vec(niters+1,n)
cluster_size = mat.or.vec(M,1) #n_i
adj_size = mat.or.vec(M,M) #n_{ij}
sum_y = mat.or.vec(M,1) #S_i

# initialize parameters
mu[1,1] = -1
mu[1,2] = 0.5
```

```
mu[1,3] = 3
sigsq[1] = 0.4
trans_est[1,] = matrix(c(1/3+0.15,1/3-0.075,1/3-0.075,
                         0+0.075,2/3-0.15,1/3+0.075,
                         2/3-0.15,0+0.075,1/3+0.075), nrow=1)
rho[1,] =  matrix(c(1/3,1/3,1/3), nrow=1)
beta[1] = g/h

# initialize the chain
trans_initial = matrix(trans_est[1,],3,byrow = TRUE)
X[1,] = HMM_generate(trans_initial,n,state,sqrt(sigsq[1]),mu
   [1,])$hidden_var
#X[1,] = hidden_var #matrix(ceiling(3 * runif(n, 0, 1)),nrow=1)
for (i in 1:M){
  cluster_size[i] = sum(X[1,] == i)
}
for (i in 1:M){
  for(j in 1:M){
    for (k in 2:n){
      if(X[1,k-1] == i & X[1,k] == j){
        adj_size[i,j] = adj_size[i,j] +1
      }
    }
  }
}
for (i in 1:M){
  sum_y[i] = sum(obs_var[X[1,] == i])
}


########## Gibbs sampling iteration ###########
```

```
## backward recursion
backward_summation <- function(A,mu,sigsq,y){
  M = dim(A)[1]
  n = length(y)
  B = mat.or.vec(n,M) #backward variables
  B[n,] = 1
  for (m in (n-1):1){
    for (i in 1:M){
      #print(m+1)
      B[m,i]  = sum(A[i,]*dnorm(y[m+1],mean=mu,sd = sqrt(sigsq)
         )*B[m+1,])
    }
    sumBrow = sum(B[m,])
    B[m,] = B[m,]/sumBrow # normalize the row to avoid decaying
        to 0
  }
  return(B)
}

## main iteration
for (t in 1:niters){

  if(t %% 100 == 0){
    cat("iter=",t,"\n")
  }

  # step 1: update mu, given hidden data, and sigma^2
  for (i in 1:M){
    mean_i = (sum_y[i] + kapa*xi*sigsq[t]) / (cluster_size[i]+
       kapa*sigsq[t])
    std_i = sqrt(sigsq[t] / (cluster_size[i]+kapa*sigsq[t]))
```

```
  mu[t+1,i] = rnorm(1,mean_i,std_i)
}

# step 2: update sigma^2, given hidden data, and beta
shape1 = alpha + 0.5*n
rate1 = beta + 0.5* sum((matrix(obs_var,nrow=1) - mu[t+1,X[t
   ,]])^2)
inv = rgamma(1,shape1,rate1)
sigsq[t+1] = 1/inv

# step 3: update beta, given sigma^2
beta[t+1] = rgamma(1,g+alpha,h+inv)

# step 4: update A, given sigma^2
A = mat.or.vec(M,M)
for (i in 1:M){
  A[i,] = rdirichlet(1, adj_size[i,]+1)
}
trans_est[t+1,] = matrix(t(A),nrow=1,byrow = FALSE)
# matrix(trans_est[t+1,],3,byrow=TRUE)

# step 5: update rho, given X1
rho[t+1,] = rdirichlet(1, (X[t,1]==1:M)+1)

# step 6: update X, given all parameters
#X[t+1,1]
B = backward_summation(A,mu[t+1,],sigsq[t+1],obs_var)
w = mat.or.vec(n,M)
for (i in 1:n){
  for (m in 1:M){
    if(i==1){
```

```
      w[i,m] =rho[m] * dnorm(obs_var[i],mean=mu[t+1,m],sd=
         sqrt(sigsq[t+1])) *B[i,m]
    }else{
      #print(i-1)
      w[i,m] =A[ X[t+1,i-1],m] * dnorm(obs_var[i],mean=mu[t
         +1,m],sd=sqrt(sigsq[t+1])) *B[i,m]
    }
  }

  w_sum = sum(w[i,])
  w_sample = runif(1, min = 0, max = w_sum)
  for (m in 1:M){
    if( w_sample<sum(w[i,1:m]) ){
      X[t+1,i] = m
      break
    }
  }
}

# step 7: update counting parameters
for (i in 1:M){
  cluster_size[i] = sum(X[t+1,] == i)
}
for (i in 1:M){
  for(j in 1:M){
    for (k in 2:n){
      if(X[t+1,k-1] == i & X[t+1,k] == j){
        adj_size[i,j] = adj_size[i,j] +1
      }
    }
  }
```

```
  }
  for (i in 1:M){
    sum_y[i] = sum(obs_var[X[t+1,] == i])
  }

}

#################### Plotting ######################
plot(1:(niters+1),mu[1:(niters+1),1],col = 'red',ylim = c(-3,3)
   ,xlab ="MCMC Iterations", ylab = "Normal Means",
     main = "Posterior Samples of Emission Mean", cex.lab=1.5,
       cex.axis=1.5, cex.main=1.5, cex.sub=1.5 )
points(1:(niters+1),mu[1:(niters+1),2],col = 'blue')
points(1:(niters+1),mu[1:(niters+1),3],col = 'black')


plot(1:(niters+1),sigsq[1:(niters+1)],col = 'blue',ylim = c
   (0,3.4),xlab ="MCMC Iterations", ylab = "Normal Variance",
     main ="Posterior Samples of Emission Variance", cex.lab
       =1.5, cex.axis=1.5, cex.main=1.5, cex.sub=1.5 )

######### calculate posterior statistics ##########
trans_mean = matrix(colMeans(trans_est[300:niters+1,]),M,byrow
   = TRUE)
sigsq_mean = mean(sigsq[300:niters+1])
mu_mean = colMeans(mu[300:niters+1,])

hidden_samples = X[300:niters+1,]
Xsumcount = mat.or.vec(M,n)
maj_vote_X = mat.or.vec(n,1)
for (j in 1:n){
```

```
  for (m in 1:M){
    Xsumcount[m,j] = sum(hidden_samples[,j] == m)
  }
  maj_vote_X[j] =which.max(Xsumcount[,j])
}
Accuracy_averaged = sum(maj_vote_X == hidden_var)/n
Accuracy_last_chain = sum(X[niters+1,] == hidden_var)/n
```