Mixture Distributions and Hidden Markov Models

Vasileios (Bill) Katsianos

August 2023

Contents

1 Finite Mixtures of Distributions

Let $y_1, \ldots, y_n$ be a sample of independent observations from a mixture distribution with PMF or PDF:

$$f_\vartheta(y_i) = \sum_{k=1}^{K} p_k f_{\theta_k}(y_i).$$

The observed-data likelihood for $\vartheta = (p_1, \ldots, p_K, \theta_1, \ldots, \theta_K)$ is given by:

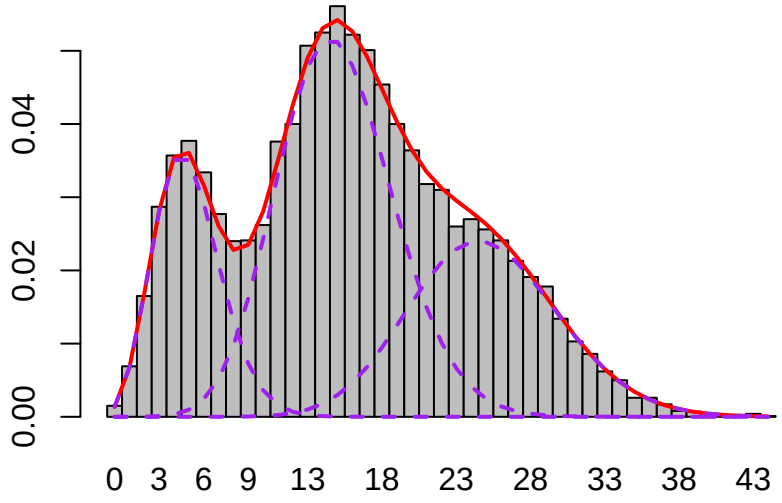
$$L(\vartheta \mid y) = \prod_{i=1}^{n} \left[\sum_{k=1}^{K} p_k f_{\theta_k}(y_i) \right],$$

so it's impossible to find a closed form solution to its maximization problem.

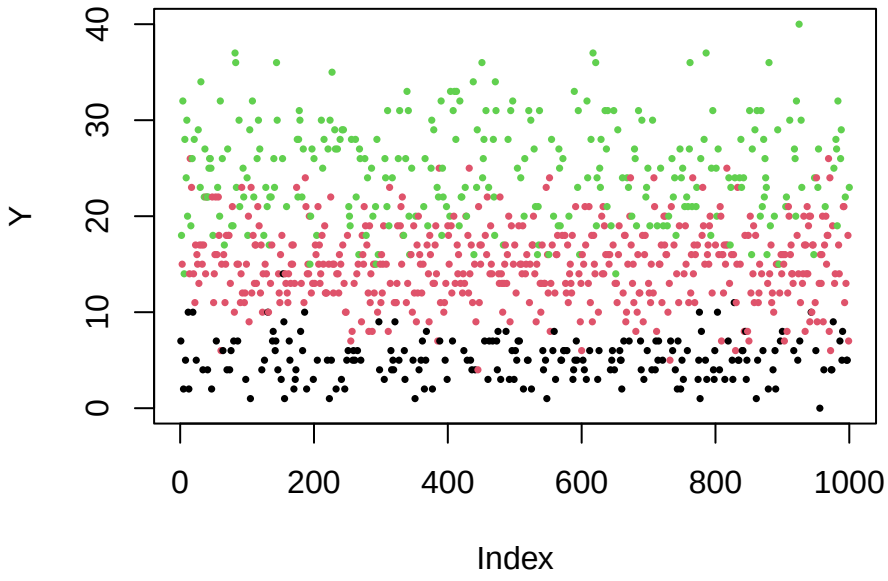
We insert the latent variables $X_1, \ldots, X_n$ with $\mathbb{P}_p(X_i = k) = p_k$. Then, $(Y_i \mid X_i = k) \sim f_{\theta_k}(\cdot)$. The complete-data likelihood, i.e. the joint likelihood of the observed variables y_i and the latent variables x_i, is given by:

$$\begin{aligned} L(\vartheta \mid y, x) &= \prod_{i=1}^{n} f_\vartheta(y_i, x_i) = \prod_{i=1}^{n} \left[\mathbb{P}_p(X_i = x_i) f_\theta\left(y_i \mid X_i = x_i\right) \right] \\ &= \prod_{i=1}^{n} \left[p_{x_i} f_{\theta_{x_i}}(y_i) \right] = \prod_{i=1}^{n} \prod_{k=1}^{K} \left[p_k f_{\theta_k}(y_i) \right]^{\mathbb{1}\{x_i = k\}}. \end{aligned}$$

```
rpoismix = function(n, theta, p) {
    K = length(theta)
    X = sample(K, n, replace = TRUE, p)
    Y = rpois(n, theta[X])
    return(list(Y = Y, X = X))
}

n = 10000
K = 3
theta = c(5, 15, 25)
p = c(0.2, 0.5, 0.3)
mix = rpoismix(n, theta, p)
Y = mix$Y
X = mix$X
barplot(table(factor(Y, levels = 0:max(Y)))/n, space = 0)
lines(0:max(Y) + 0.5, p[1] * dpois(0:max(Y), theta[1]) + p[2] * dpois(0:max(Y),
    theta[2]) + p[3] * dpois(0:max(Y), theta[3]), col = "red", lwd = 2)
lines(0:max(Y) + 0.5, p[1] * dpois(0:max(Y), theta[1]), col = "purple", lty = 2,
    lwd = 2)
lines(0:max(Y) + 0.5, p[2] * dpois(0:max(Y), theta[2]), col = "purple", lty = 2,
    lwd = 2)
lines(0:max(Y) + 0.5, p[3] * dpois(0:max(Y), theta[3]), col = "purple", lty = 2,
    lwd = 2)
```




```r
n = 1000
mix = rpoismix(n, theta, p)
Y = mix$Y
X = mix$X
plot(Y, pch = 16, col = X, cex = 0.5)
```



2 Expectation-Maximization (EM) Algorithm

Instead of directly maximizing the observed-data likelihood, we work iteratively. We start with some initial estimate $\vartheta^{(0)}$. We calculate the conditional expectation of the complete-data log-likelihood $\ell(\vartheta \mid y, x) = \log L(\vartheta \mid y, x)$ given the observed variables under the parameter value $\vartheta^{(0)}$, i.e. the following function:

$$Q_{\vartheta^{(0)}}(\vartheta) = \mathbb{E}_{\vartheta^{(0)}}\left[\ell(\vartheta \mid y, X) \mid y\right].$$

This function is called the intermediate quantity of the EM algorithm. This step of the EM algorithm is called the Expectation step (E-step). Afterwards, we maximize the function $Q_{\vartheta^{(0)}}(\vartheta)$ with respect to ϑ and we get a new estimate $\vartheta^{(1)}$. This step of the EM algorithm is called the Maximization step (M-step). We iterate these two steps

until the algorithm converges to some estimate ϑ^*.

Theorem 1. If the observed-data likelihood $L(\vartheta \mid y)$ is bounded, then the value ϑ^* to which the EM algorithm converges is one of its local maxima.

Proof. First, we observe that:

$$L(\vartheta \mid y, x) = f_\vartheta(x, y) = f_\vartheta(y) f_\vartheta(x \mid y) = L(\vartheta \mid y) f_\vartheta(x \mid y).$$

We take the logs of both sides in the above equation:

$$\ell(\vartheta \mid y, x) = \ell(\vartheta \mid y) + \log f_\vartheta(x \mid y).$$

If the random variable X is absolutely continuous, we multiply both sides of the above equation by the density $f_{\vartheta^{(0)}}(x \mid y)$ and integrate with respect to x:

$$\int \ell(\vartheta \mid y, x) f_{\vartheta^{(0)}}(x \mid y) dx = \int \ell(\vartheta \mid y) f_{\vartheta^{(0)}}(x \mid y) dx + \int \log f_\vartheta(x \mid y) f_{\vartheta^{(0)}}(x \mid y) dx.$$

If the random variable X is discrete, we would instead multiply both sides by the probability $\mathbb{P}_{\vartheta^{(0)}}(X = x \mid y)$ and sum over all possible values of x. We observe that:

$$\int \ell(\vartheta \mid y, x) f_{\vartheta^{(0)}}(x \mid y) dx = \mathbb{E}_{\vartheta^{(0)}}\left[\ell(\vartheta \mid y, X) \mid y\right] = Q_{\vartheta^{(0)}}(\vartheta),$$

$$\int \ell(\vartheta \mid y) f_{\vartheta^{(0)}}(x \mid y) dx = \ell(\vartheta \mid y) \int f_{\vartheta^{(0)}}(x \mid y) dx = \ell(\vartheta \mid y).$$

Additionally, we define:

$$\mathcal{H}_{\vartheta^{(0)}}(\vartheta) = -\int \log f_\vartheta(x \mid y) f_{\vartheta^{(0)}}(x \mid y) dx = -\mathbb{E}_{\vartheta^{(0)}}\left[\log f_\vartheta(X \mid y) \mid y\right].$$

Therefore, we have succeeded in decomposing the observed-data likelihood into two parts:

$$\ell(\vartheta \mid y) = Q_{\vartheta^{(0)}}(\vartheta) + \mathcal{H}_{\vartheta^{(0)}}(\vartheta).$$

Making use of Jensen's inequality, we can show that:

$$\begin{aligned}
\mathcal{H}_{\vartheta^{(0)}}\left(\vartheta^{(1)}\right) - \mathcal{H}_{\vartheta^{(0)}}\left(\vartheta^{(0)}\right) &= -\mathbb{E}_{\vartheta^{(0)}}\left[\log f_{\vartheta^{(1)}}(X \mid y) \mid y\right] + \mathbb{E}_{\vartheta^{(0)}}\left[\log f_{\vartheta^{(0)}}(X \mid y) \mid y\right] \\
&= -E_{\vartheta^{(0)}}\left[\log \frac{f_{\vartheta^{(1)}}(X \mid y)}{f_{\vartheta^{(0)}}(X \mid y)} \middle| y\right] \geqslant -\log E_{\vartheta^{(0)}}\left[\frac{f_{\vartheta^{(1)}}(X \mid y)}{f_{\vartheta^{(0)}}(X \mid y)} \middle| y\right] \\
&= -\log \int \frac{f_{\vartheta^{(1)}}(x \mid y)}{\cancel{f_{\vartheta^{(0)}}(x \mid y)}} \cancel{f_{\vartheta^{(0)}}(x \mid y)} dx = -\log \int f_{\vartheta^{(1)}}(x \mid y) dx = -\log 1 = 0.
\end{aligned}$$

This inequality is known as the fundamental inequality of the EM algorithm. If $\vartheta^{(0)}$ is our current estimate of ϑ, then this inequality shows that whatever our next estimate $\vartheta^{(1)}$ is going to be, the function $\mathcal{H}_{\vartheta^{(0)}}(\cdot)$ is certainly not going to drop below the current value $\mathcal{H}_{\vartheta^{(0)}}\left(\vartheta^{(0)}\right)$. Since the value of the function $\mathcal{H}$ is guaranteed to increase in each step of the EM algorithm, we can completely ignore it and solely focus on the function Q.

If we select any value $\vartheta^{(1)}$ which increases the value of the function $Q_{\vartheta^{(0)}}(\cdot)$, i.e. $Q_{\vartheta^{(0)}}\left(\vartheta^{(1)}\right) > Q_{\vartheta^{(0)}}\left(\vartheta^{(0)}\right)$, then we

would get that $\ell\left(\vartheta^{(1)} \middle| y, x\right) > \ell\left(\vartheta^{(0)} \middle| y, x\right)$. By repeating this process, we produce a sequence of estimates which increase the value of the observed-data likelihood at each step and finally converges to one of its local maxima.

Obviously, if we select the value $\vartheta^{(1)}$ which exactly maximizes the function $Q_{\vartheta^{(0)}}(\cdot)$, i.e. $\vartheta^{(1)} = \operatorname{argmax}_{\vartheta} Q_{\vartheta^{(0)}}(\vartheta)$, then the algorithm will achieve the maximum possible rate of convergence. This is exactly the goal of the EM algorithm. However, even if it's not possible to directly maximize the function $Q_{\vartheta^{(0)}}(\cdot)$, the algorithm is still guaranteed to converge to some local maximum of the observed-data likelihood, provided that we select a new estimate which even slightly increases the value of the function Q at each step. For this reason, many variations of the classical EM algorithm have been proposed over the years.

3 Application of the EM algorithm to Mixture Distributions

First, we take the log of the complete-data likelihood of the mixture model:

$$\ell(\vartheta \mid y, x) = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\{x_i = k\} \left[\log p_k + \log f_{\theta_k}(y_i)\right].$$

Next, we calculate the intermediate quantity of the EM algorithm:

$$\begin{aligned} Q_{\vartheta^{(0)}}(\vartheta) = \mathbb{E}_{\vartheta^{(0)}}\left[\ell(\vartheta \mid y, X) \mid y\right] &= \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{E}_{\vartheta^{(0)}}\left[\mathbb{1}\{X_i = k\} \mid \cancel{y_1}, \ldots, y_i, \ldots, \cancel{y_n}\right] \left[\log p_k + \log f_{\theta_k}(y_i)\right] \\ &= \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{P}_{\vartheta^{(0)}}\left(X_i = k \mid y_i\right) \left[\log p_k + \log f_{\theta_k}(y_i)\right]. \end{aligned}$$

Therefore, the E-step of the EM algorithm for mixture distributions reduces to the calculation of the following conditional probabilities:

$$w_{ik} = \mathbb{P}_{\vartheta}\left(X_i = k \mid y_i\right).$$

These conditional probabilities are given by Bayes' theorem:

$$w_{ik} \propto \mathbb{P}_p(X_i = k) f_{\theta}\left(y_i \mid X_i = k\right) = p_k f_{\theta_k}(y_i).$$

Additionally, we define the corresponding normalization constants:

$$c_i(\vartheta) = \sum_{\ell=1}^{K} p_\ell f_{\theta_\ell}(y_i), \quad i = 1, 2, \ldots, n.$$

Returning to the formula for the observed-data likelihood of a mixture model, we observe that:

$$L(\vartheta \mid y) = \prod_{i=1}^{n} c_i(\vartheta), \quad \ell(\vartheta \mid y) = \log L(\vartheta \mid y) = \sum_{i=1}^{n} \log c_i(\vartheta).$$

According to the theory of the EM algorithm, the value of the observed-data likelihood must increase at each step of the algorithm.

The probability vectors and the observed-data likelihood, must always be calculated on the log scale for reasons of

numerical stability. For this reason, we make use of the Log-Sum-Exp trick. For example, we define:

$$v_{ik} = \log p_k + \log f_{\theta_k}(y_i), \quad m_i = \max_{k \in \{1,\dots,K\}} v_{ik}.$$

Then, we get that:

$$w_{ik} = \frac{e^{v_{ik}-m_i}}{\sum_{\ell=1}^{K} e^{v_{i\ell}-m_i}}, \quad \log c_i(\vartheta) = m_i + \log \sum_{\ell=1}^{K} e^{v_{i\ell}-m_i}.$$

Having calculated the conditional probabilities probabilities w_{ik} for $\vartheta = \vartheta^{(0)}$, we have completely specified the intermediate quantity of the EM algorithm. Therefore, we can move on to the M-step, which depends on the choice of PMFs or PDFs $f_{\theta_k}(\cdot)$.

4 Application of the EM Algorithm to Mixtures of Poisson Distributions

Suppose that $(Y_i \mid X_i = k) \sim \text{Poisson}(\theta_k)$. According to the previous paragraph, we calculate that:

$$Q_{\vartheta^{(0)}}(\vartheta) = \sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik} \left[\log p_k - \theta_k + y_i \log \theta_k - \log(y_i!)\right].$$

First, we maximize the intermediate quantity of the EM algorithm with respect to the probability vector p, taking into account the fact that $\sum_{k=1}^{K} p_k = 1$. We define the following Lagrangian function:

$$\mathcal{L}(p, \lambda) = \sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik} \log p_k - \lambda \left(\sum_{k=1}^{K} p_k - 1 \right).$$

By differentiating with respect to p_k, we calculate that:

$$\frac{\partial \mathcal{L}(p, \lambda)}{\partial p_k} = \frac{1}{p_k} \sum_{i=1}^{n} w_{ik} - \lambda \quad \Rightarrow \quad p_k^{(1)} = \frac{1}{\lambda} \sum_{i=1}^{n} w_{ik}.$$

We observe that $\sum_{k=1}^{K} w_{ik} = 1$. By applying the constraint $\sum_{k=1}^{K} p_k^{(1)} = 1$, we calculate the value of λ as follows:

$$1 = \sum_{k=1}^{K} p_k^{(1)} = \frac{1}{\lambda} \sum_{k=1}^{K} \sum_{i=1}^{n} w_{ik} = \frac{1}{\lambda} \sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik} = \frac{1}{\lambda} \sum_{i=1}^{n} 1 = \frac{n}{\lambda} \quad \Rightarrow \quad \lambda = n.$$

Therefore, we infer that:

$$p_k^{(1)} = \frac{1}{n} \sum_{i=1}^{n} w_{ik},$$

i.e. the new estimate of p_k is the average conditional posterior probability of each observation belonging to group k. Obviously, this maximization procedure remains the same for any finite mixture distribution.

Next, we differentiate the intermediate quantity of the EM algorithm with respect to θ_k:

$$\frac{\partial Q_{\vartheta^{(0)}}(\vartheta)}{\partial \theta_k} = \sum_{i=1}^{n} w_{ik} \left(-1 + \frac{y_i}{\theta_k} \right) \quad \Rightarrow \quad \theta_k^{(1)} = \frac{\sum_{i=1}^{n} w_{ik} y_i}{\sum_{i=1}^{n} w_{ik}},$$

i.e. the new estimate of θ_k is the weighted average of all observations, where each observation is weighted by the conditional probability of it belonging to group k.

Having calculated the MLE $\widehat{\vartheta}$ via the EM algorithm, we can also calculate the probability distributions w_{ik} for $\vartheta = \widehat{\vartheta}$. Then, we can calculate the maximum a posteriori estimates of the latent variables $X_1, \ldots, X_n$ as follows:

$$\widehat{X}_i = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \, w_{ik}, \quad i = 1, 2, \ldots, n.$$

```
loglikpoismix = function(Y, theta, p) {
    n = length(Y)
    K = length(theta)
    w = matrix(0, n, K)
    loglik = 0
    for (i in 1:n) {
        logw = log(p) + dpois(Y[i], theta, log = TRUE)
        maximum = max(logw)
        unnormalized = exp(logw - maximum)
        c = sum(unnormalized)
        w[i, ] = unnormalized/c
        loglik = loglik + maximum + log(c)
    }
    return(list(w = w, loglik = loglik))
}

EMpoismix = function(Y, theta, p, tol = 1e-05) {
    steps = 1
    poismix = loglikpoismix(Y, theta, p)
    w = poismix$w
    loglik = poismix$loglik
    err = Inf
    while (err > tol) {
        steps = steps + 1
        p = colMeans(w)
        theta = colMeans(w * Y)/p
        poismix = loglikpoismix(Y, theta, p)
        w = poismix$w
        loglik[steps] = poismix$loglik
        err = loglik[steps] - loglik[steps - 1]
    }
    return(list(theta = theta, p = p, w = w, loglik = loglik))
}

MLE = EMpoismix(Y, mean(Y) + sd(Y) * ((1 - K)/2):((K - 1)/2), rep(1, K)/K)
```

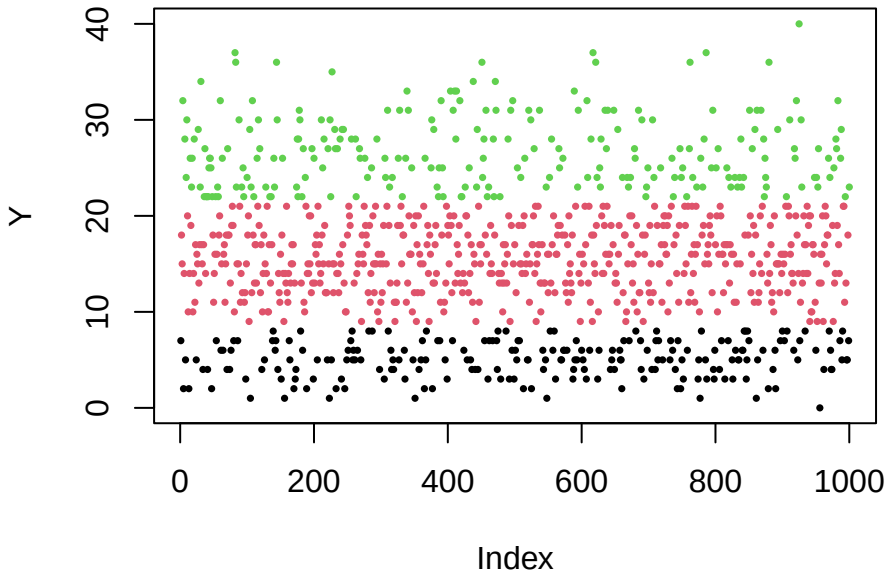
```
print(MLE$theta)
```

```
## [1]  5.281834 15.588634 25.617278
```

```
print(MLE$p)
```

```
## [1] 0.2171178 0.5270992 0.2557829
```

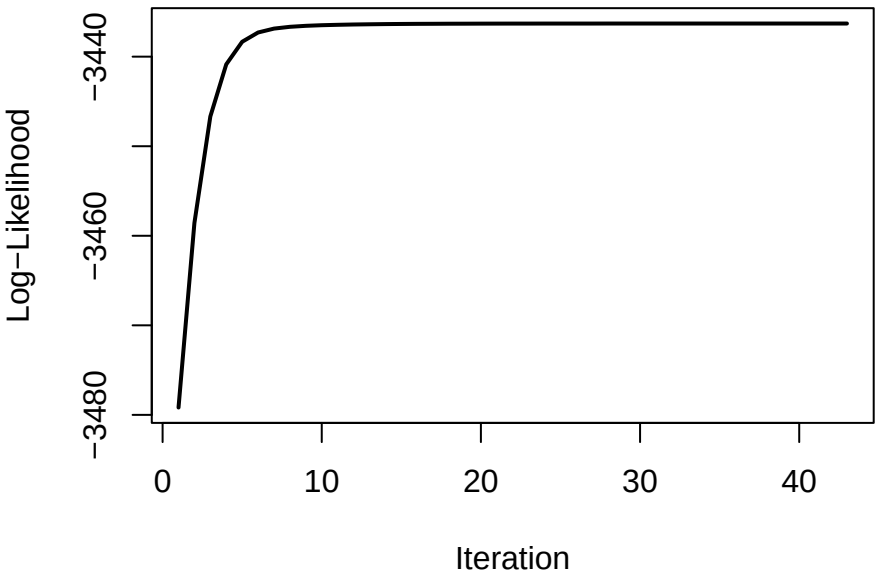
```
XMAP = apply(MLE$w, 1, which.max)
plot(Y, col = XMAP, pch = 16, cex = 0.5)
```




```
table(X, XMAP)
```

```
##    XMAP
## X     1   2   3
##   1 191  13   0
##   2  24 447  28
##   3   0  86 211
```

```
plot(MLE$loglik[-1], type = "l", xlab = "Iteration", ylab = "Log-Likelihood",
    lwd = 2)
```



5 Application of the Gibbs Sampler to Mixture Distributions

Consider the conditionally conjugate Dirichlet(α) prior distribution for the probability vector p with probability density function:

$$f(p) = \frac{\Gamma(K\alpha)}{[\Gamma(\alpha)]^K} \prod_{k=1}^{K} p_k^{\alpha-1} \propto \prod_{k=1}^{K} p_k^{\alpha-1}.$$

Then, the conditional posterior distribution of p is given by:

$$f(p \mid x) \propto f(p) \prod_{i=1}^{n} \mathbb{P}(X_i = x_i \mid p) \propto \prod_{k=1}^{K} \left[p_k^{\alpha-1} \prod_{i=1}^{n} p_k^{\mathbb{1}_{\{x_i=k\}}} \right] = \prod_{k=1}^{K} p_k^{N_k+\alpha-1},$$

i.e. $p \mid x \sim \text{Dirichlet}\,(N_1+\alpha, N_2+\alpha, \ldots, N_K+\alpha)$, where $N_k = \sum_{i=1}^{n} \mathbb{1}_{\{x_i=k\}}$ for $k = 1, 2, \ldots, K$. As a special case, we can consider Jeffreys' prior for p, which results for $\alpha = 0.5$. In order to simulate from the conditional posterior distribution of p, we can first simulate random variables $W_k \sim \text{Gamma}(N_k+\alpha, 1)$ for $k = 1, 2, \ldots, K$ and then let $p = \frac{1}{W}(W_1, W_2, \ldots, W_K)$, where $W = \sum_{k=1}^{K} W_k$.

Next, we consider the conditionally conjugate Gamma(β, λ) prior distribution for the parameter θ_k with probability density function:

$$f(\theta_k) = \frac{\lambda^\beta}{\Gamma(\beta)} \theta_k^{\beta-1} e^{-\lambda\theta_k} \propto \theta_k^{\beta-1} e^{-\lambda\theta_k}.$$

Then, the conditional posterior distribution of θ_k is given by:

$$f(\theta_k \mid x, y) \propto f(\theta_k) \prod_{i=1}^{n} f(y_i \mid x_i, \theta_k) \propto \theta_k^{\beta-1} e^{-\lambda\theta_k} \prod_{i=1}^{n} \left(e^{-\theta_k} \frac{\theta_k^{y_i}}{y_i!} \right)^{\mathbb{1}_{\{x_i=k\}}} \propto \theta_k^{S_k+\beta-1} e^{-(N_k+\lambda)\theta_k},$$

i.e. $\theta_k \mid x, y \sim \text{Gamma}(S_k+\beta, N_k+\lambda)$, where $S_k = \sum_{i=1}^{n} y_i \mathbb{1}_{\{x_i=k\}}$ for $k = 1, 2, \ldots, K$. As a special case, we can consider the improper Jeffreys' prior for θ_k, which is given by $f(\theta_k) \propto \theta_k^{-0.5}$ and results for $\beta = 0.5$, $\lambda = 0$.

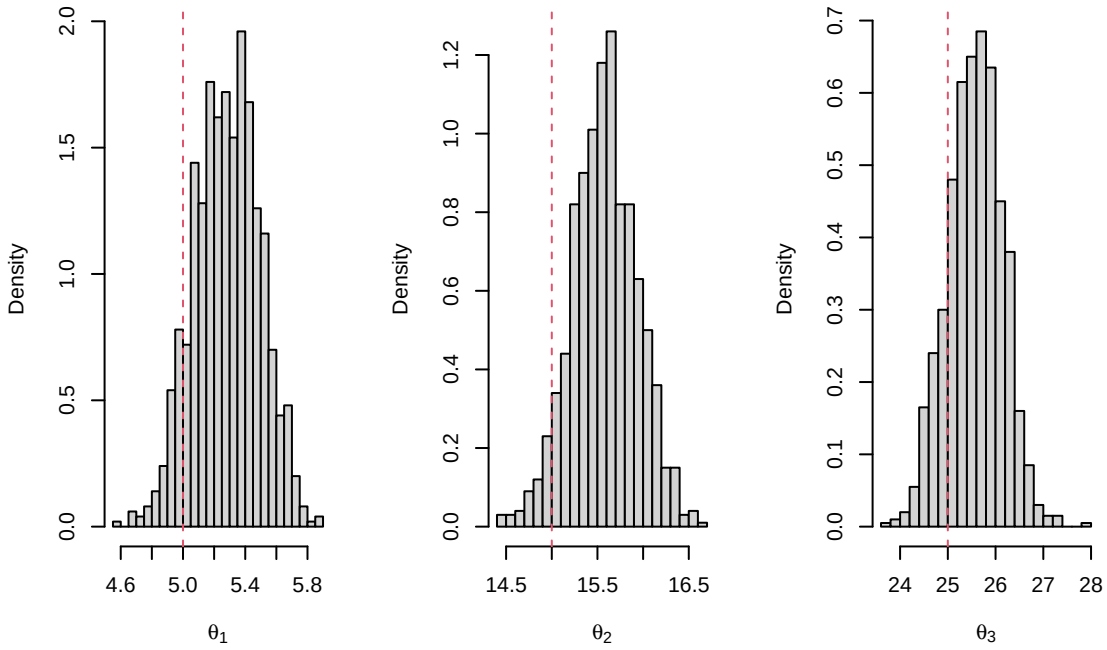
Finally, we know that the conditional posterior distribution of the latent variables X_i is given by:

$$\mathbb{P}(X_i = k \mid y_i, \vartheta) = w_{ik} \propto p_k f_{\theta_k}(y_i).$$

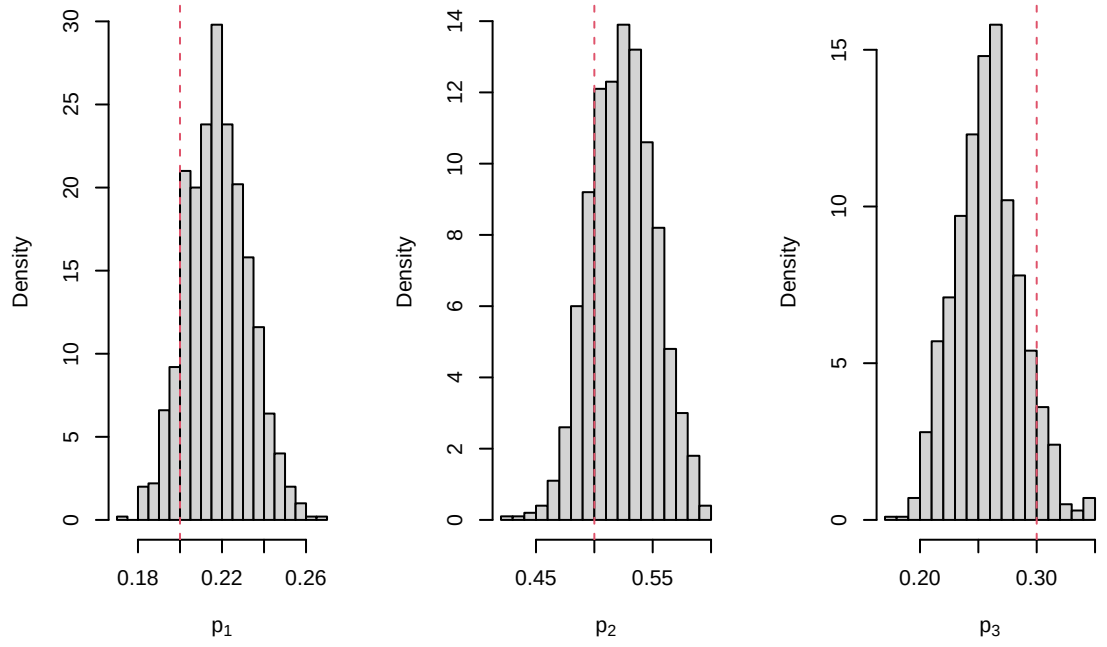
Since the likelihood of the mixture model and the prior distributions we're considering treat the parameters which correspond to different groups symmetrically, the phenomenon of label switching can sometimes appear in the application of the MCMC algorithms to mixture distributions. In order to deal with this phenomenon, we can try imposing some identifiability constraint such as $\theta_1 < \theta_2 < \cdots < \theta_K$ at each step of the MCMC algorithm.

```
MCMCpoismix = function(Y, theta0, p0, alpha, beta, lambda, niter, nburn) {
    K = length(theta)
    theta = matrix(0, niter, K)
    p = matrix(0, niter, K)
    X = matrix(0, niter, n)
    theta[1, ] = theta0
    p[1, ] = p0
    w = loglikpoismix(Y, theta[1, ], p[1, ])$w
    X[1, ] = apply(w, 1, function(x) {
        sample(K, 1, prob = x)
    })
    for (j in 2:niter) {
        x = factor(X[j - 1, ], levels = 1:K)
        N = table(x)
        p[j, ] = rgamma(K, N + alpha)
        p[j, ] = p[j, ]/sum(p[j, ])
        S = aggregate(Y ~ x, FUN = sum, drop = FALSE)[, 2]
        S[is.na(S)] = 0
        theta[j, ] = rgamma(K, S + beta, N + lambda)
        w = loglikpoismix(Y, theta[j, ], p[j, ])$w
        X[j, ] = apply(w, 1, function(x) {
            sample(K, 1, prob = x)
        })
        I = order(theta[j, ])
        theta[j, ] = theta[j, I]
        p[j, ] = p[j, I]
        X[j, ] = I[X[j, ]]
    }
    return(list(theta = theta[-(1:nburn), ], p = p[-(1:nburn), ], X = X[-(1:nburn),
        ]))
}

posterior = MCMCpoismix(Y, mean(Y) + sd(Y) * ((1 - K)/2):((K - 1)/2), rep(1,
    K)/K, 0.5, 0.5, 0, 2000, 1000)
par(mfrow = c(1, 3))
hist(posterior$theta[, 1], "FD", freq = FALSE, main = NA, xlab = expression(theta[1]))
abline(v = theta[1], col = 2, lty = 2)
hist(posterior$theta[, 2], "FD", freq = FALSE, main = NA, xlab = expression(theta[2]))
abline(v = theta[2], col = 2, lty = 2)
```

```r
hist(posterior$theta[, 3], "FD", freq = FALSE, main = NA, xlab = expression(theta[3]))
abline(v = theta[3], col = 2, lty = 2)
```

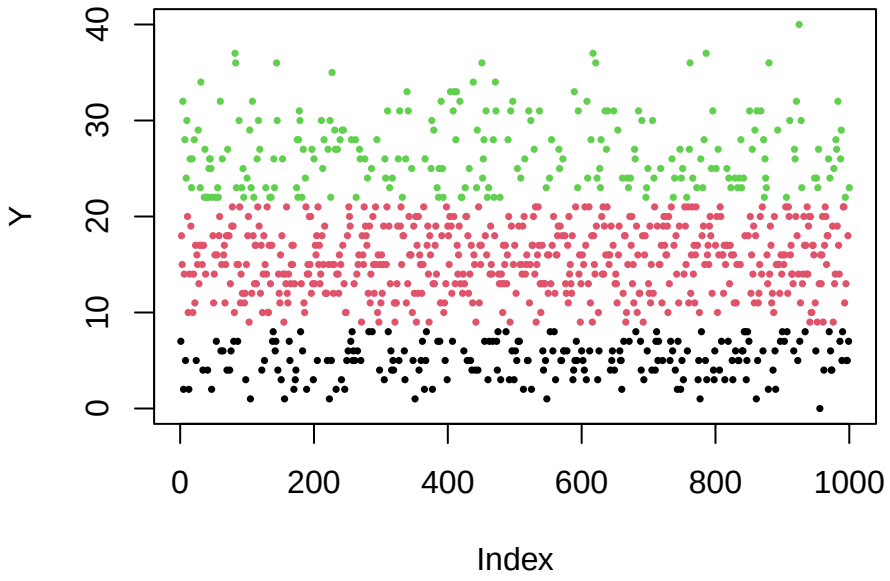



```r
hist(posterior$p[, 1], "FD", freq = FALSE, main = NA, xlab = expression(p[1]))
abline(v = p[1], col = 2, lty = 2)
hist(posterior$p[, 2], "FD", freq = FALSE, main = NA, xlab = expression(p[2]))
abline(v = p[2], col = 2, lty = 2)
hist(posterior$p[, 3], "FD", freq = FALSE, main = NA, xlab = expression(p[3]))
abline(v = p[3], col = 2, lty = 2)
```




```r
XMAP = apply(posterior$X, 2, function(x) {
    which.max(table(factor(x, levels = 1:K)))
```

```
})
plot(Y, col = XMAP, pch = 16, cex = 0.5)
```




```
table(X, XMAP)
```

```
##    XMAP
## X     1   2   3
##   1 191  13   0
##   2  24 447  28
##   3   0  86 211
```

6 Application of Generalized Likelihood Ratio Testing to Mixture Distributions

Suppose that we want to compare two nested mixture models with different numbers of groups. Consider the null model $\mathcal{M}_0$ with $K-1$ groups and the alternative model $\mathcal{M}_1$ with K groups. We calculate the MLE $\widehat{\vartheta}_0$ of the null model $\mathcal{M}_0$ and the MLE $\widehat{\vartheta}_1$ of the alternative model $\mathcal{M}_1$ by use of the EM algorithm. Then, we get the observed value of the generalized likelihood ratio statistic as follows:

$$\text{LR}^{\text{obs}} = -2\left[\ell\left(\widehat{\vartheta}_0 \mid y\right) - \ell\left(\widehat{\vartheta}_1 \mid y\right)\right].$$

We generally know that the generalized likelihood ratio statistic follows the χ^2_ν distribution under the null model $\mathcal{M}_0$, where ν is the number of restrictions that the null model $\mathcal{M}_0$ sets upon the alternative model $\mathcal{M}_1$. However, in the case of nested mixture models with different numbers of groups, the number of restrictions ν is not uniquely defined, so the regularity conditions of Wilks' theorem aren't satisfied.

For this reason, we may choose to make use of the parametric bootstrap method in order to estimate the distribution of the generalized likelihood ratio statistic under the null model $\mathcal{M}_0$. In other words, we simulate samples $y^{(1)}, y^{(2)}, \ldots, y^{(n_{\text{boot}})}$ from the null model $\mathcal{M}_0$ with parameter vector $\widehat{\vartheta}_0$. Next, we calculate the MLE $\widehat{\vartheta}_0^{(j)}$ of the null model $\mathcal{M}_0$ and the MLE $\widehat{\vartheta}_1^{(j)}$ of the alternative model $\mathcal{M}_1$ by use of the EM algorithm based on the

sample $y^{(j)}$ for $j = 1, 2, \ldots, n_{\text{boot}}$. Finally, we calculate the value $\text{LR}_j^{\text{boot}}$ of the bootstrapped generalized likelihood ratio statistic, in the same way we calculated the observed value LR^{obs}. Then, we can estimate the p-value of the generalized likelihood ratio test as follows:

$$\text{p-value} = \frac{1 + \sum_{j=1}^{n_{\text{boot}}} \mathbb{1}_{\{\text{LR}_j^{\text{boot}} > \text{LR}^{\text{obs}}\}}}{1 + n_{\text{boot}}}.$$

```
LRpoismix = function(n, theta, p, LRobs, nboot, tol = 1e-05) {
    K = length(theta)
    LRboot = numeric(nboot)
    for (i in 1:nboot) {
        Y = rpoismix(n, theta, p)$Y
        loglik0 = EMpoismix(Y, mean(Y) + sd(Y) * ((1 - K)/2):((K - 1)/2), rep(1,
            K)/K, tol)$loglik
        loglik1 = EMpoismix(Y, mean(Y) + sd(Y) * (-K/2):(K/2), rep(1, K + 1)/(K +
            1), tol)$loglik
        LRboot[i] = -2 * (tail(loglik0, 1) - tail(loglik1, 1))
    }
    pval = (1 + sum(LRboot > LRobs))/(1 + nboot)
    return(pval)
}

MLE0 = EMpoismix(Y, mean(Y) + sd(Y) * ((1 - K/2):(K/2 - 1)), rep(1, K - 1)/(K -
    1))
LRobs = -2 * (tail(MLE0$loglik, 1) - tail(MLE$loglik, 1))
print(LRobs)
```

```
## [1] 257.3839
```

```
LRpoismix(n, MLE0$theta, MLE0$p, LRobs, 100)
```

```
## [1] 0.00990099
```

7 Hidden Markov Models (HMMs)

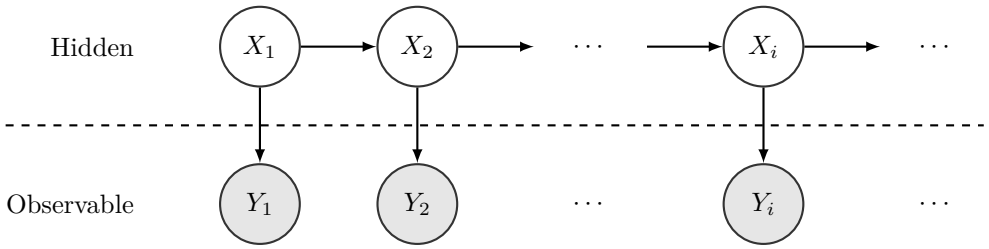
Let $\{X_i\}$ be a time-homogeneous discrete-time Markov chain with initial distribution $p_k = \mathbb{P}(X_1 = k)$ and first-order transition probabilities $p_{k,\ell} = \mathbb{P}(X_{i+1} = \ell \mid X_i = k)$ for $k, \ell = 1, 2, \ldots, K$.

Consider a stochastic process $\{Y_i\}$. Suppose that the stochastic process $\{Y_i\}$ is conditionally independent given the Markov chain $\{X_i\}$, that the conditional distribution of the random variable Y_i given the Markov chain $\{X_i\}$ solely depends on the random variable X_i and that $(Y_i \mid X_i = k) \sim f_{\theta_k}(\cdot)$. In other words,

$$\begin{aligned} f_\theta(y_1, \ldots, y_n \mid X_1 = x_1, \ldots, X_n = x_n) &= \prod_{i=1}^{n} f_\theta(y_i \mid X_1 = x_1, \ldots, X_n = x_n) \\ &= \prod_{i=1}^{n} f_\theta(y_i \mid X_i = x_i) = \prod_{i=1}^{n} f_{\theta_{x_i}}(y_i). \end{aligned}$$

Then, the bivariate stochastic process $\{(X_i, Y_i)\}$ is a time-homogeneous discrete-time hidden Markov model with finite state-space.

The Markov chain $\{X_i\}$ is latent, i.e. non-observable. We insert it in order to explain the serial autocorrelation of the observable stochastic process $\{Y_i\}$. A graphical representation of the dependence structure of a HMM is shown in the following figure.



In contrast with the Markov chain $\{X_i\}$ whose every random variable X_i only depends on the value of the immediately preceding random variable X_{i-1}, every random variable Y_i of the stochastic process $\{Y_i\}$ depends on the values of all preceding random variables $Y_1, \ldots, Y_{i-1}$. In other words, even though the stochastic process $\{Y_i\}$ is conditionally independent given the Markov chain $\{X_i\}$, it displays "infinite memory" without this conditioning.

Conditional independence plays a central role in all calculations pertaining to HMMs. We say that a random variable X and a random variable Y are conditionally independent given a random variable Z, denoted by $(X \perp\!\!\!\perp Y) \mid Z$, if the conditional distribution of X given Z and the conditional distribution of Y given Z are independent.

We can easily infer the conditional independence from the graphical representation of the dependence structure of a HMM. More specifically, if we condition on a random variable Z which "cuts" the path between the random variables X and Y, then $(X \perp\!\!\!\perp Y) \mid Z$. For example,

$$((Y_1, \ldots, Y_i) \perp\!\!\!\perp (Y_{i+1}, \ldots, Y_n)) \mid X_i,$$

$$((Y_1, \ldots, Y_i) \perp\!\!\!\perp X_{i+1}) \mid X_i,$$

$$((Y_i, \ldots, Y_n) \perp\!\!\!\perp X_{i-1}) \mid X_i.$$

Let $y_1, \ldots, y_n$ be a sample from a stochastic process. We insert the hidden Markov chain $\{X_i\}$. The complete-data likelihood, i.e. the joint likelihood of the observed variables y_i and the latent variables x_i, for parameter vector $\vartheta = (p_1, \ldots, p_K, p_{1,1}, \ldots, p_{K,K}, \theta_1, \ldots, \theta_K)$ is given by:

$$\begin{aligned}
L(\vartheta \mid y, x) &= f_\vartheta(x, y) = \mathbb{P}_\vartheta(X = x) f_\vartheta(y \mid x) \\
&= \mathbb{P}_p\left(X_1 = x_1\right) \cdot \prod_{i=2}^{n} \mathbb{P}_p\left(X_i = x_i \mid X_{i-1} = x_{i-1}\right) \cdot \prod_{i=1}^{n} f_\theta\left(y_i \mid X_i = x_i\right) \\
&= p_{x_1} \cdot \prod_{i=2}^{n} p_{x_{i-1}, x_i} \cdot \prod_{i=1}^{n} f_{\theta_{x_i}}(y_i) \\
&= \prod_{k=1}^{K} p_k^{\mathbb{1}\{x_i = k\}} \cdot \prod_{i=2}^{n} \prod_{k=1}^{K} \prod_{\ell=1}^{K} p_{k,\ell}^{\mathbb{1}\{x_{i-1} = k, x_i = \ell\}} \cdot \prod_{i=1}^{n} \prod_{k=1}^{K} f_{\theta_k}(y_i)^{\mathbb{1}\{x_i = k\}}.
\end{aligned}$$

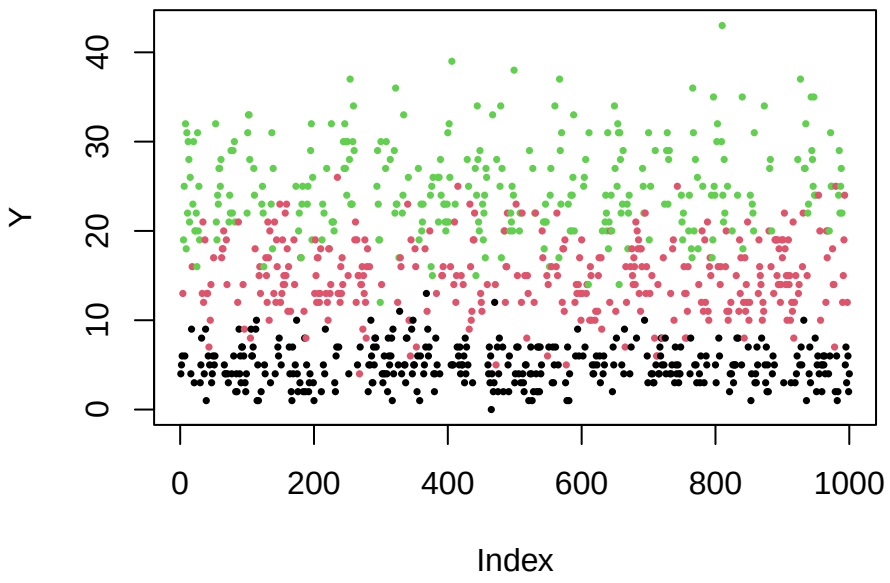
In contrast, the observed-data likelihood is given by the chain rule as follows:

$$\begin{aligned} L(\vartheta \mid y) &= f_\vartheta(y_1) \prod_{i=2}^{n} f_\vartheta(y_i \mid y_1, \ldots, y_{i-1}) \\ &= \left[\sum_{k=1}^{K} \mathbb{P}_p \left(X_1 = k \right) f_\theta \left(y_1 \mid X_1 = k \right) \right] \\ &\quad \times \prod_{i=2}^{n} \left[\sum_{k=1}^{K} \mathbb{P}_\vartheta \left(X_i = k \mid y_1, \ldots, y_{i-1} \right) f_\theta \left(y_i \mid \cancel{y_1, \ldots, y_{i-1}}, X_i = k \right) \right] \\ &= \left[\sum_{k=1}^{K} p_k f_{\theta_k}(y_1) \right] \prod_{i=2}^{n} \left[\sum_{k=1}^{K} \mathbb{P}_\vartheta \left(X_i = k \mid y_1, \ldots, y_{i-1} \right) f_{\theta_k}(y_i) \right]. \end{aligned}$$

Once again, it's impossible to find a closed form solution to the maximization problem of this observed-data likelihood.

In contrast with the straightforward calculation of the observed-data likelihood in mixture distributions for given parameter values, the calculation of the observed-data likelihood in HMMs requires the calculation of the conditional probabilities $\phi_{i|i-1}(k) = \mathbb{P}_\vartheta \left(X_i = k \mid y_1, \ldots, y_{i-1} \right)$. These conditional probabilities are called predictive, since they predict the current state of the hidden Markov chain given the values of all preceding observable variables. The calculation of these probabilities requires a recursive scheme. Such a recursive scheme is involved in every estimation method for the parameters of a HMM, so the calculation of the observed-data likelihood is a direct byproduct of the estimation process.

```
rpoisHMM = function(n, theta, p, P) {
    K = length(theta)
    X = numeric(n)
    X[1] = sample(K, 1, prob = p)
    for (i in 2:n) {
        X[i] = sample(K, 1, prob = P[X[i - 1], ])
    }
    Y = rpois(n, theta[X])
    return(list(Y = Y, X = X))
}

n = 1000
K = 3
theta = c(5, 15, 25)
p = c(1, 0, 0)
P = matrix(c(0.5, 0.3, 0.2, 0.3, 0.6, 0.1, 0.2, 0.1, 0.7), 3)
HMM = rpoisHMM(n, theta, p, P)
Y = HMM$Y
X = HMM$X
plot(Y, col = X, pch = 16, cex = 0.5)
```



8 Application of the EM Algorithm to HMMs

First, we take the log of the complete-data likelihood of the HMM:

$$\ell(\vartheta \mid y, x) = \sum_{k=1}^{K} \mathbb{1}\{x_1 = k\} \log p_k + \sum_{i=2}^{n} \sum_{k=1}^{K} \sum_{\ell=1}^{K} \mathbb{1}\{x_{i-1} = k, x_i = \ell\} \log p_{k,\ell} + \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\{x_i = k\} \log f_{\theta_k}(y_i).$$

Next, we calculate the intermediate quantity of the EM algorithm:

$$Q_{\vartheta^{(0)}}(\vartheta) = \sum_{k=1}^{K} \mathbb{P}_{\vartheta^{(0)}}\left(X_1 = k \mid y_1, \ldots, y_n\right) \log p_k + \sum_{i=2}^{n} \sum_{k=1}^{K} \sum_{\ell=1}^{K} \mathbb{P}_{\vartheta^{(0)}}\left(X_{i-1} = k, X_i = \ell \mid y_1, \ldots, y_n\right) \log p_{k,\ell} + \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{P}_{\vartheta^{(0)}}\left(X_i = k \mid y_1, \ldots, y_n\right) \log f_{\theta_k}(y_i).$$

Therefore, the E-step of the EM algorithm in HMMs reduces to the calculation of the conditional probabilities $\phi_{i|n}(k) = \mathbb{P}_{\vartheta}\left(X_i = k \mid y_1, \ldots, y_n\right)$ and $\phi_{i-1,i|n}(k, \ell) = \mathbb{P}_{\vartheta}\left(X_{i-1} = k, X_i = \ell \mid y_1, \ldots, y_n\right)$. These conditional probabilities are called smoothing, because they smooth out the noise contained in the entire sample to infer the distribution of the hidden Markov chain. These conditional probabilities cannot be directly calculated though use of Bayes' theorem - they require a recursive scheme, which also includes the calculation of the predictive probabilities.

9 Forward-Backward Algorithm

The Forward-Backward algorithm aims at the calculation of the univariate smoothing distributions $\phi_{i|n}(\cdot)$ and the bivariate smoothing distributions $\phi_{i-1,i|n}(\cdot, \cdot)$, so it's incorporated in the E-step of the EM algorithm for the parameter estimation of a HMM. It constitutes of two consecutive recursive schemes.

The first recursive scheme aims at the calculation of the conditional probabilities $\phi_{i|i}(k) = \mathbb{P}_{\vartheta}\left(X_i = k \mid y_1, \ldots, y_i\right)$.

These conditional probabilities are called filtering, because they filter the values of the currently observed variables to infer the distribution of the current state of the hidden Markov chain. This recursive scheme filters the entire observable stochastic process from start to finish. For this reason, it's called forward filtering.

The second recursive scheme makes use of the filtering distributions $\phi_{i|i}(\cdot)$ for the calculation of the smoothing probabilities. This recursive scheme smooths out the entire observable stochastic process from finish to start. For this reason, it's called backward smoothing.

Forward Filtering

Our goal is to calculate the filtering distributions $\phi_{i|i}(\cdot)$ for $i = 1, 2, \ldots, n$. By applying Bayes' theorem, we get that:

$$\begin{aligned}
\phi_{1|1}(k) &= \mathbb{P}_\vartheta\left(X_1 = k \mid y_1\right) \\
&\propto \mathbb{P}_p\left(X_1 = k\right) f_\theta\left(y_1 \mid X_1 = k\right) \\
&= p_k f_{\theta_k}(y_1),
\end{aligned}$$

$$\begin{aligned}
\phi_{i|i}(k) &= \mathbb{P}_\vartheta\left(X_i = k \mid y_1, \ldots, y_{i-1}, y_i\right) \\
&\propto \mathbb{P}_\vartheta\left(X_i = k \mid y_1, \ldots, y_{i-1}\right) f_\theta\left(y_i \mid \cancel{y_1, \ldots, y_{i-1}}, X_i = k\right) \\
&= \phi_{i|i-1}(k) f_{\theta_k}(y_i), \quad i = 2, 3, \ldots, n.
\end{aligned}$$

We see that the calculation of each filtering distribution for $i = 2, 3, \ldots, n$ requires the calculation of the current predictive distribution $\phi_{i|i-1}(\cdot)$. By applying the law of total probability, we get that:

$$\begin{aligned}
\phi_{i|i-1}(k) &= \mathbb{P}_\vartheta\left(X_i = k \mid y_1, \ldots, y_{i-1}\right) \\
&= \sum_{\ell=1}^{K} \mathbb{P}_\vartheta\left(X_{i-1} = \ell \mid y_1, \ldots, y_{i-1}\right) \underbrace{\mathbb{P}_p\left(X_i = k \mid X_{i-1} = \ell, \cancel{y_1, \ldots, y_{i-1}}\right)}_{((Y_1,\ldots,Y_{i-1}) \perp\!\!\!\perp X_i) \mid X_{i-1}} \\
&= \sum_{\ell=1}^{K} \phi_{i-1|i-1}(\ell) p_{\ell,k}, \quad i = 2, 3, \ldots, n.
\end{aligned}$$

We see that the calculation of each predictive distribution $\phi_{i|i-1}(\cdot)$ requires the calculation of the immediately preceding filtering distribution $\phi_{i-1|i-1}(\cdot)$. Therefore, we get a recursive scheme according to which we alternately calculate the filtering and predictive distributions of the hidden Markov chain from start to finish.

Furthermore, we define the normalization constants of the filtering distributions:

$$c_1(\vartheta) = \sum_{\ell=1}^{K} p_\ell f_{\theta_\ell}(y_1),$$

$$c_i(\vartheta) = \sum_{\ell=1}^{K} \phi_{i|i-1}(\ell) f_{\theta_\ell}(y_i), \quad i = 2, 3, \ldots, n.$$

Returning to the formula for the observed-data likelihood of a HMM, we observe that:

$$L(\vartheta \mid y) = c_1(\vartheta) \cdot \prod_{i=2}^{n} c_i(\vartheta) = \prod_{i=1}^{n} c_i(\vartheta), \quad \ell(\vartheta \mid y) = \log L(\vartheta \mid y) = \sum_{i=1}^{n} \log c_i(\vartheta).$$

Therefore, as a byproduct of the forward filtering algorithm, we can calculate the observed-data likelihood of the

HMM for the current estimate of ϑ. According to the theory of the EM algorithm, the value of the observed-data likelihood must increase at each step of the algorithm.

As always, the filtering distributions and observed-data likelihood, must be calculated on the log scale for reasons of numerical stability. For this reason, we make use of the Log-Sum-Exp trick. For example, we define:

$$v_k = \log p_k + \log f_{\theta_k}(y_1), \quad m_1 = \max_{k \in \{1,\dots,K\}} v_k.$$

Then, we get that:

$$\phi_{1|1}(k) = \frac{e^{v_k - m_1}}{\sum_{\ell=1}^{K} e^{v_\ell - m_1}}, \quad \log c_1(\vartheta) = m_1 + \log \sum_{\ell=1}^{K} e^{v_\ell - m_1}.$$

We apply the same logic to the rest of the filtering distributions. The recursive forward filtering scheme is summarized below.

Algorithm 1 Forward Filtering

Input: Observations $y_1, \dots, y_n$ and current estimate ϑ.

1: Calculate the normalization constant $\log c_1(\vartheta)$.

2: Calculate the filtering distribution $\phi_{1|1}(\cdot)$.

3: For $i = 2, 3, \dots, n$, calculate:

i: the predictive distribution $\phi_{i|i-1}(\cdot)$,

ii: the normalization constant $\log c_i(\vartheta)$,

iii: the filtering distribution $\phi_{i|i}(\cdot)$.

4: Calculate the observed-data log-likelihood $\ell(\vartheta \mid y)$.

Output: Filtering distributions $\phi_{i|i}(\cdot)$ and observed-data log-likelihood $\ell(\vartheta \mid y)$.

Backward Smoothing

Our goal is to calculate the smoothing distributions $\phi_{i|n}(\cdot)$ and $\phi_{i-1,i|n}(\cdot,\cdot)$. First, we define the following backward variables:

$$b_i(k) = f_\vartheta\left(y_{i+1}, \dots, y_n \mid X_i = k\right), \quad i = 1, 2, \dots, n-1.$$

By applying Bayes' theorem, we get that:

$$\begin{aligned}
\phi_{i|n}(k) &= \mathbb{P}_\vartheta\left(X_i = k \mid y_1, \dots, y_i, y_{i+1}, \dots, y_n\right) \\
&\propto \mathbb{P}_\vartheta\left(X_i = k \mid y_1, \dots, y_i\right) \underbrace{f_\vartheta\left(y_{i+1}, \dots, y_n \mid \cancel{y_1, \dots, y_i}, X_i = k\right)}_{((Y_1,\dots,Y_i) \perp\!\!\!\perp (Y_{i+1},\dots,Y_n)) \mid X_i} \\
&= \phi_{i|i}(k) b_i(k), \quad i = 1, 2, \dots, n-1,
\end{aligned}$$

$$
\begin{aligned}
\phi_{i-1,i|n}(k,\ell) &= \mathbb{P}_\vartheta\left(X_{i-1}=k, X_i=\ell \mid y_1,\dots,y_{i-1},y_i,\dots,y_n\right)\\
&\propto \underbrace{\mathbb{P}_\vartheta\left(X_{i-1}=k, X_i=\ell \mid y_1,\dots,y_{i-1}\right)}_{\text{chain rule}} \underbrace{f_\vartheta\left(y_i,\dots,y_n \mid \cancel{y_1,\dots,y_{i-1}}, \cancel{X_{i-1}=k}, X_i=\ell\right)}_{((Y_i,\dots,Y_n)\perp\!\!\!\perp X_{i-1})|X_i}\\
&= P_\vartheta\left(X_{i-1}=k \mid y_1,\dots,y_{i-1}\right) \underbrace{P_p\left(X_i=\ell \mid X_{i-1}=k, \cancel{y_1,\dots,y_{i-1}}\right)}_{((Y_1,\dots,Y_{i-1})\perp\!\!\!\perp X_i)|X_{i-1}}\\
&\quad\times \underbrace{f_\theta\left(y_i \mid X_i=\ell\right) f_\vartheta\left(y_{i+1},\dots,y_n \mid X_i=\ell\right)}_{(Y_i\perp\!\!\!\perp(Y_{i+1},\dots,Y_n))|X_i}\\
&= \phi_{i-1|i-1}(k) p_{k,\ell} f_{\theta_\ell}(y_i) b_i(\ell), \quad i=2,3,\dots,n-1,
\end{aligned}
$$

$$
\begin{aligned}
\phi_{n-1,n|n}(k,\ell) &= \mathbb{P}_\vartheta\left(X_{n-1}=k, X_n=\ell \mid y_1,\dots,y_{n-1},y_n\right)\\
&\propto \underbrace{\mathbb{P}_\vartheta\left(X_{n-1}=k, X_n=\ell \mid y_1,\dots,y_{n-1}\right)}_{\text{chain rule}} f_\theta\left(y_n \mid \cancel{y_1,\dots,y_{n-1}}, \cancel{X_{n-1}=k}, X_n=\ell\right)\\
&= P_\vartheta\left(X_{n-1}=k \mid y_1,\dots,y_{n-1}\right) \underbrace{P_p\left(X_n=\ell \mid X_{n-1}=k, \cancel{y_1,\dots,y_{n-1}}\right)}_{((Y_1,\dots,Y_{n-1})\perp\!\!\!\perp X_n)|X_{n-1}} f_{\theta_\ell}(y_n)\\
&= \phi_{n-1|n-1}(k) p_{k,\ell} f_{\theta_\ell}(y_n).
\end{aligned}
$$

We see that the calculation of all smoothing distributions is fully determined by the knowledge of the filtering distributions and the current backward variables. The backward variables are recursively calculated through the law of total probability:

$$
\begin{aligned}
b_i(k) &= f_\vartheta\left(y_{i+1},\dots,y_n \mid X_i=k\right)\\
&= \sum_{\ell=1}^K \mathbb{P}_p\left(X_{i+1}=\ell \mid X_i=k\right) \underbrace{f_\vartheta\left(y_{i+1},\dots,y_n \mid \cancel{X_i=k}, X_{i+1}=\ell\right)}_{((Y_{i+1},\dots,Y_n)\perp\!\!\!\perp X_i)|X_{i+1}}\\
&= \sum_{\ell=1}^K p_{k,\ell} \underbrace{f_\theta\left(y_{i+1} \mid X_{i+1}=\ell\right) f_\vartheta\left(y_{i+2},\dots,y_n \mid X_{i+1}=\ell\right)}_{(Y_{i+1}\perp\!\!\!\perp(Y_{i+2},\dots,Y_n))|X_{i+1}}\\
&= \sum_{\ell=1}^K p_{k,\ell} f_{\theta_\ell}(y_{i+1}) b_{i+1}(\ell), \quad i=1,2,\dots,n-2,
\end{aligned}
$$

$$
\begin{aligned}
b_{n-1}(k) &= f_\vartheta\left(y_n \mid X_{n-1}=k\right)\\
&= \sum_{\ell=1}^K \mathbb{P}_p\left(X_n=\ell \mid X_{n-1}=k\right) f_\theta\left(y_n \mid \cancel{X_{n-1}=k}, X_n=\ell\right)\\
&= \sum_{\ell=1}^K p_{k,\ell} f_{\theta_\ell}(y_n).
\end{aligned}
$$

Therefore, we get a recursive scheme according to which we alternately calculate the smoothing distributions and the backward variables of the hidden Markov chain from finish to start. The final marginal smoothing distribution coincides with the final filtering distribution, which has already been calculated. The rest of the smoothing distributions are calculated using the Log-Sum-Exp trick in the same way as the filtering distributions. The backward smoothing recursive scheme is summarized below.

Algorithm 2 Backward Smoothing

Input: Observations $y_1, \ldots, y_n$, current estimate ϑ and filtering distributions $\phi_{i|i}(\cdot)$.

1: Calculate the bivariate smoothing distribution $\phi_{n-1,n|n}(\cdot,\cdot)$.

2: Calculate the backward variables $b_{n-1}(\cdot)$.

3: For $i = n-1, n-2, \ldots, 2$, calculate:

i: the marginal smoothing distribution $\phi_{i|n}(\cdot)$,

ii: the bivariate smoothing distribution $\phi_{i-1,i|n}(\cdot,\cdot)$,

iii: the backward variables $b_{i-1}(\cdot)$.

4: Calculate the marginal smoothing distribution $\phi_{1|n}(\cdot)$.

Output: Smoothing distributions $\phi_{i|n}(\cdot)$ and $\phi_{i-1,i|n}(\cdot,\cdot)$.

Markovian Backward Smoothing

Markovian backward smoothing is an alternative recursive scheme which aims at the calculation of the smoothing distributions $\phi_{i|n}(\cdot)$ and $\phi_{i-1,i|n}(\cdot,\cdot)$. In contrast with the backward smoothing scheme, it can lead to the calculation of the joint smoothing distribution of the entire hidden Markov chain. This enables the design of efficient MCMC algorithms for the estimation of the joint posterior distribution of the parameters of a HMM. In place of the backward variables, it makes use of the following backward transition probabilities:

$$B_i(k,\ell) = \mathbb{P}_\vartheta\left(X_i = k \mid X_{i+1} = \ell, y_1, \ldots, y_i\right), \quad i = 1, 2, \ldots, n-1.$$

We observe that $\sum_{k=1}^{K} B_i(k,\ell) = 1$. Be applying the chain rule, we calculate that:

$$\begin{aligned}
\phi_{i-1,i|n}(k,\ell) &= \mathbb{P}_\vartheta\left(X_{i-1} = k, X_i = \ell \mid y_1, \ldots, y_n\right) \\
&= \mathbb{P}_\vartheta\left(X_i = \ell \mid y_1, \ldots, y_n\right) \underbrace{\mathbb{P}_\vartheta\left(X_{i-1} = k \mid X_i = \ell, y_1, \ldots, y_{i-1}, \cancel{y_i, \ldots, y_n}\right)}_{((Y_i,\ldots,Y_n) \perp\!\!\!\perp X_{i-1} \mid X_i)} \\
&= \phi_{i|n}(\ell) B_{i-1}(k,\ell), \quad i = 2, 3, \ldots, n.
\end{aligned}$$

We see that the calculation of each bivariate smoothing distribution is fully determined by the calculation of the immediately preceding marginal smoothing distribution and the current backward transition probabilities. Next, by applying the law of total probability, we get that:

$$\begin{aligned}
\phi_{i|n}(k) &= \mathbb{P}_\vartheta\left(X_i = k \mid y_1, \ldots, y_n\right) \\
&= \sum_{\ell=1}^{K} \mathbb{P}_\vartheta\left(X_i = k, X_{i+1} = \ell \mid y_1, \ldots, y_n\right) \\
&= \sum_{\ell=1}^{K} \phi_{i,i+1}(k,\ell), \quad i = 1, 2, \ldots, n-1.
\end{aligned}$$

We see that the calculation of each marginal smoothing distribution is fully determined by the calculation of the current bivariate smoothing distribution. The backward transition probabilities are calculated according to Bayes'

theorem as follows:

$$\begin{aligned} B_i(k,\ell) &= \mathbb{P}_\vartheta\left(X_i = k \mid X_{i+1} = \ell, y_1, \ldots, y_i\right) \\ &\propto \mathbb{P}_\vartheta\left(X_i = k \mid y_1, \ldots, y_i\right) \underbrace{\mathbb{P}_p\left(X_{i+1} = \ell \mid X_i = k, \cancel{y_1, \ldots, y_i}\right)}_{((Y_1,\ldots,Y_i) \perp\!\!\!\perp X_{i+1}) \mid X_i} \\ &= \phi_{i|i}(k) p_{k,\ell}, \quad i = 1, 2, \ldots, n-1. \end{aligned}$$

We see that the calculation of the backward transition probabilities is fully determined by the knowledge of the corresponding filtering distribution. Therefore, we get a recursive scheme according to which we alternately calculate the backward transition probabilities and the smoothing distributions of the hidden Markov chain from finish to start.

Finally, we can decompose the joint smoothing distribution of the entire hidden Markov chain by use of the chain rule:

$$\begin{aligned} \phi_{1,\ldots,n|n}(x_1, \ldots, x_n) &= \mathbb{P}_\vartheta(X_1 = x_1, \ldots, X_n = x_n \mid y_1, \ldots, y_n) \\ &= \mathbb{P}_\vartheta\left(X_n = x_n \mid y_1, \ldots, y_n\right) \\ &\quad \times \prod_{i=1}^{n-1} \underbrace{\mathbb{P}_\vartheta\left(X_i = x_i \mid X_{i+1} = x_{i+1}, \cancel{X_{i+2} = x_{i+2}}, \ldots, \cancel{X_n = x_n}, y_1, \ldots, y_i, \cancel{y_{i+1}, \ldots, y_n}\right)}_{\substack{((X_{i+2},\ldots,X_n)) \perp\!\!\!\perp X_i \mid X_{i+1} \\ ((Y_{i+1},\ldots,Y_n) \perp\!\!\!\perp X_i) \mid X_{i+1}}} \\ &= \phi_{n|n}(x_n) \prod_{i=1}^{n-1} B_i\left(x_i, x_{i+1}\right). \end{aligned}$$

The joint smoothing distribution is essentially the conditional posterior distribution of the entire hidden Markov chain. We observe that the conditional posterior distribution of the hidden Markov chain preserves the Markov property, but is no longer time-homogeneous. The backward probabilities $B_i(\cdot,\cdot)$ essentially determine the transition probability matrix of the conditional posterior distribution of the reverse Markov chain during its $(n-i)$-th transition. We can easily see that this transition probability matrix is different at every transition of the Markov chain.

The backward probabilities are calculated using the Log-Sum-Exp trick. We define:

$$v_i(k,\ell) = \log \phi_{i|i}(k) + \log p_{k,\ell}, \quad m_i(\ell) = \max_{k \in \{1,\ldots,K\}} v_i(k,\ell).$$

Then, we get that:

$$B_i(k,\ell) = \frac{e^{v_i(k,\ell) - m_i(\ell)}}{\sum_{j=1}^K e^{v_i(j,\ell) - m_i(\ell)}}.$$

The Markovian backward smoothing recursive scheme is summarized on the next page.

10 Baum-Welch Algorithm

We start with some initial estimate $\vartheta^{(0)}$. In the E-step of the EM algorithm, we implement the Forward-Backward algorithm for $\vartheta = \vartheta^{(0)}$ and get the smoothing distributions $\phi_{i|n}(\cdot)$, $\phi_{i-1,i|n}(\cdot,\cdot)$. The EM algorithm which incorporates the Forward-Backward algorithm for the calculation of the smoothing distributions is called the Baum-Welch algorithm. In this way, we have fully determined the intermediate quantity of the EM algorithm and

Algorithm 3 Markovian Backward Smoothing

Input: Current estimate ϑ and filtering distributions $\phi_{i|i}(\cdot)$.

1: For $i = n-1, n-2, \ldots, 1$, calculate:

i: the backward transition probabilities $B_i(\cdot, \cdot)$,

ii: the bivariate smoothing distribution $\phi_{i,i+1|n}(\cdot, \cdot)$,

iii: the marginal smoothing distribution $\phi_{i|n}(\cdot)$.

Output: Smoothing distributions $\phi_{i|n}(\cdot)$ and $\phi_{i,i+1|n}(\cdot, \cdot)$.

we can proceed with the M-step.

Suppose that $(Y_i \mid X_i = k) \sim \text{Poisson}(\theta_k)$. In this particular case, we calculate that:

$$Q_{\vartheta^{(0)}}(\vartheta) = \sum_{k=1}^{K} \phi_{1|n}(k) \log p_k + \sum_{i=2}^{n} \sum_{k=1}^{K} \sum_{\ell=1}^{K} \phi_{i-1,i|n}(k,\ell) \log p_{k,\ell} + \sum_{i=1}^{n} \sum_{k=1}^{K} \phi_{i|n}(k) \left[-\theta_k + y_i \log \theta_k - \log(y_i!)\right].$$

First, we note that the estimation of the initial distribution of the hidden Markov chain is solely based on the conditional posterior distribution of X_1. Therefore, there is no hope of obtaining a consistent estimate of the initial distribution based on only one sequence of observations $y_1, \ldots, y_n$. Unless we have some prior information about the initial distribution, the most usual strategy is to consider it known to be the discrete uniform distribution on the state-space $S = \{1, 2, \ldots, K\}$ of the hidden Markov chain.

We know that $\sum_{\ell=1}^{K} p_{k,\ell} = 1$. Therefore, we maximize the intermediate quantity of the EM algorithm with respect to each row of the transition probability matrix separately. We define the following Lagrangian function:

$$\mathcal{L}(p_k, \lambda_k) = \sum_{i=2}^{n} \sum_{\ell=1}^{K} \phi_{i-1,i|n}(k,\ell) \log p_{k,\ell} - \lambda_k \left(\sum_{\ell=1}^{K} p_{k,\ell} - 1 \right).$$

By differentiating with respect to $p_{k,\ell}$, we calculate that:

$$\frac{\partial \mathcal{L}(p_k, \lambda_k)}{\partial p_{k,\ell}} = \frac{1}{p_{k,\ell}} \sum_{i=2}^{n} \phi_{i-1,i|n}(k,\ell) - \lambda_k \quad \Rightarrow \quad p_{k,\ell}^{(1)} = \frac{1}{\lambda_k} \sum_{i=2}^{n} \phi_{i-1,i|n}(k,\ell) = \frac{1}{\lambda_k} \sum_{i=1}^{n-1} \phi_{i,i+1|n}(k,\ell).$$

We observe that $\sum_{\ell=1}^{K} \phi_{i,i+1|n}(k,\ell) = \phi_{i|n}(k)$. By making use of the constraint $\sum_{\ell=1}^{K} p_{k,\ell}^{(1)} = 1$, we calculate the value of the Lagrange multiplier λ_k:

$$1 = \sum_{\ell=1}^{K} p_{k,\ell}^{(1)} = \frac{1}{\lambda_k} \sum_{\ell=1}^{K} \sum_{i=1}^{n-1} \phi_{i,i+1|n}(k,\ell) = \frac{1}{\lambda_k} \sum_{i=1}^{n-1} \sum_{\ell=1}^{K} \phi_{i,i+1|n}(k,\ell) = \frac{1}{\lambda_k} \sum_{i=1}^{n-1} \phi_{i|n}(k) \quad \Rightarrow \quad \lambda_k = \sum_{i=1}^{n-1} \phi_{i|n}(k).$$

Therefore, we infer that:

$$p_{k,\ell}^{(1)} = \frac{\sum_{i=1}^{n-1} \phi_{i,i+1|n}(k,\ell)}{\sum_{i=1}^{n-1} \phi_{i|n}(k)},$$

i.e. the new estimate of $p_{k,\ell}$ is the weighted average of the conditional posterior probabilities of transitioning from

state k to state ℓ, where each probability is weighted by the conditional posterior probability of being in state k. Obviously, the same maximization holds for any finite state-space HMM.

Next, we differentiate the intermediate quantity of the EM algorithm with respect to θ_k:

$$\frac{\partial Q_{\vartheta^{(0)}}(\vartheta)}{\partial \theta_k} = \sum_{i=1}^{n} \phi_{i|n}(k) \left(-1 + \frac{y_i}{\theta_k} \right) \quad \Rightarrow \quad \theta_k^{(1)} = \frac{\sum_{i=1}^{n} \phi_{i|n}(k) y_i}{\sum_{i=1}^{n} \phi_{i|n}(k)},$$

i.e. the new estimate of θ_k is the weighted average of all observations, where each observation is weighted by the conditional posterior probability of it originating from state k. We can see that the estimation of the unknown parameters of the distributions $f_{\theta_k}(\cdot)$ remains the same whether we are working on a mixture distribution or a HMM.

```
forward = function(Y, theta, p, P) {
    n = length(Y)
    K = length(theta)
    filter = matrix(0, n, K)
    logfilter = log(p[p > 0]) + dpois(Y[1], theta[p > 0], log = TRUE)
    maximum = max(logfilter)
    unnormalized = exp(logfilter - maximum)
    c = sum(unnormalized)
    filter[1, p > 0] = unnormalized/c
    loglik = maximum + log(c)
    for (i in 2:n) {
        predict = colSums(filter[i - 1, ] * P)
        logfilter = log(predict) + dpois(Y[i], theta, log = TRUE)
        maximum = max(logfilter)
        unnormalized = exp(logfilter - maximum)
        c = sum(unnormalized)
        filter[i, ] = unnormalized/c
        loglik = loglik + maximum + log(c)
    }
    return(list(filter = filter, loglik = loglik))
}

backward = function(filter, P) {
    n = dim(filter)[1]
    K = dim(P)[1]
    bivariate = array(0, c(K, K, n - 1))
    marginal = matrix(0, n, K)
    marginal[n, ] = filter[n, ]
    for (i in (n - 1):1) {
        for (k in 1:K) {
            logB = log(filter[i, ]) + log(P[, k])
            unnormalized = exp(logB - max(logB))
```

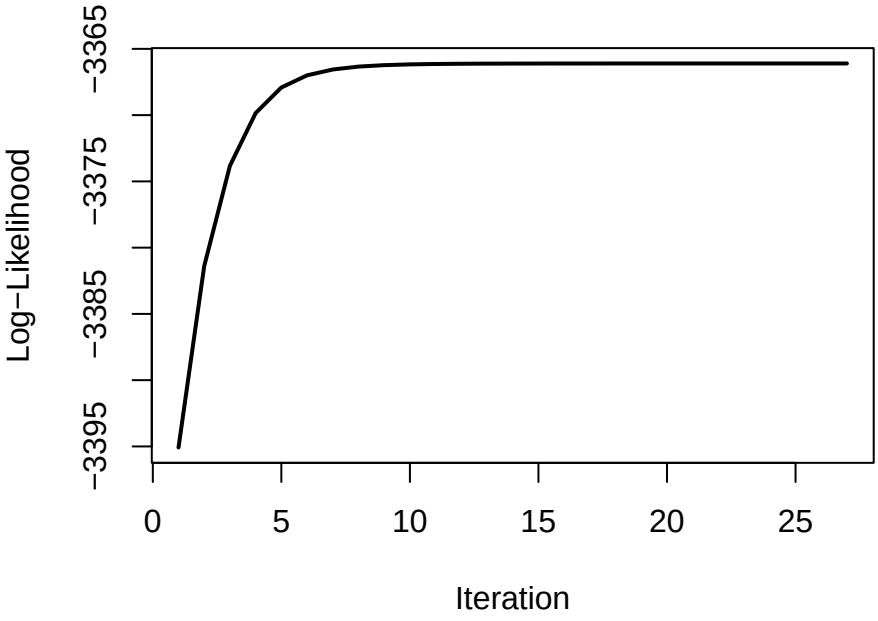
```
            B = unnormalized/sum(unnormalized)
            bivariate[, k, i] = marginal[i + 1, k] * B
        }
        marginal[i, ] = rowSums(bivariate[, , i])
    }
    return(list(marginal = marginal, bivariate = bivariate))
}

EMpoisHMM = function(Y, theta, p, P, tol = 1e-05) {
    steps = 1
    f = forward(Y, theta, p, P)
    loglik = f$loglik
    b = backward(f$filter, P)
    marginal = b$marginal
    bivariate = b$bivariate
    err = Inf
    while (err > tol) {
        steps = steps + 1
        P = apply(bivariate, 1:2, sum)
        P = P/rowSums(P)
        theta = colSums(marginal * Y)/colSums(marginal)
        f = forward(Y, theta, p, P)
        loglik[steps] = f$loglik
        b = backward(f$filter, P)
        marginal = b$marginal
        bivariate = b$bivariate
        err = loglik[steps] - loglik[steps - 1]
    }
    return(list(theta = theta, P = P, marginal = marginal, loglik = loglik))
}

MLE = EMpoisHMM(Y, mean(Y) + sd(Y) * ((1 - K)/2):((K - 1)/2), p, matrix(1, K,
    K)/K)
print(MLE$theta)
```

```
## [1]  4.928938 15.160789 24.837596
```

```
print(MLE$P)
```

```
##           [,1]      [,2]       [,3]
## [1,] 0.5733331 0.2123264 0.21434048
## [2,] 0.2633581 0.6434395 0.09320237
## [3,] 0.2191574 0.1412264 0.63961621
```

```r
plot(MLE$loglik[-1], type = "l", xlab = "Iteration", ylab = "Log-Likelihood",
    lwd = 2)
```



11 Latent State Decoding

Having estimated all unknown parameters of a HMM by use of the Baum-Welch or some other algorithm, we can also estimate the hidden states $x_1, \ldots, x_n$ from which the observations $y_1, \ldots, y_n$ have originated. This process is called decoding of the hidden Markov chain. The decoding can be either global or local.

Global Decoding

Global decoding aims at the maximization of the conditional posterior distribution of the entire hidden Markov chain given the entire observable sequence, i.e. the joint smoothing distribution $\phi_{1,\ldots,n|n}(\cdot,\ldots,\cdot)$. Since this is an n-dimensional distribution, its maximization isn't directly feasible - it requires some algorithm which implements the principle of dynamic programming.

The Viterbi algorithm is based on the principle of dynamic programming and is used in the global decoding of the hidden Markov chain. We want to define the optimal value function and to formulate the Bellman optimality equations. By applying Bayes' theorem, we observe that:

$$
\begin{aligned}
\phi_{1,\ldots,i|i}(x_1,\ldots,x_i) &= \mathbb{P}_\vartheta\left(X_1 = x_1,\ldots,X_i = x_i \mid y_1,\ldots,y_i\right) \\
&= \frac{\mathbb{P}_\vartheta\left(X_1 = x_1,\ldots,X_i = x_i\right) f_\theta\left(y_1,\ldots,y_i \mid X_1 = x_1,\ldots,X_i = x_i\right)}{f_\vartheta(y_1,\ldots,y_i)} \\
&= \frac{1}{L\left(\vartheta \mid y_1,\ldots,y_i\right)} \cdot a_{x_1} \cdot \prod_{j=2}^{i} p_{x_{j-1},x_j} \cdot \prod_{j=1}^{i} f_{\theta_{x_j}}(y_j) \\
&= \frac{L\left(\vartheta \mid y_1,\ldots,y_{i-1}\right)}{L\left(\vartheta \mid y_1,\ldots,y_i\right)} \cdot \frac{p_{x_{i-1},x_i} f_{\theta_{x_i}}(y_i)}{L\left(\vartheta \mid y_1,\ldots,y_{i-1}\right)} \cdot a_{x_1} \cdot \prod_{j=2}^{i-1} p_{x_{j-1},x_j} \cdot \prod_{j=1}^{i-1} f_{\theta_{x_j}}(y_j) \\
&= \frac{L\left(\vartheta \mid y_1,\ldots,y_{i-1}\right)}{L\left(\vartheta \mid y_1,\ldots,y_i\right)} \cdot \phi_{1,\ldots,i-1|i-1}(x_1,\ldots,x_{i-1}) \cdot p_{x_{i-1},x_i} f_{\theta_{x_i}}(y_i), \quad i = 2,3,\ldots,n.
\end{aligned}
$$

Therefore, we define the optimal value function on the log scale as follows:

$$v_i(k) = \ell\left(\vartheta \mid y_1, \ldots, y_i\right) + \max_{(x_1,\ldots,x_{i-1}) \in S^{i-1}} \log \phi_{1,\ldots,i|i}(x_1, \ldots, x_{i-1}, k), \quad i = 1, 2, \ldots, n.$$

This represents the maximal conditional probability of a sequence of hidden states which arrives at state k during time period i given all the observations up to time period i. First, we observe that:

$$\begin{aligned} v_1(k) &= \ell(\vartheta \mid y_1) + \log \phi_{1|1}(k) \\ &= \log\left[f_\vartheta(y_1)\mathbb{P}_\vartheta\left(X_1 = k \mid y_1\right)\right] \\ &= \log\left[\mathbb{P}_p\left(X_1 = k\right) f_\theta\left(y_1 \mid X_1 = x_1\right)\right] \\ &= \log p_k + \log f_{\theta_k}(y_1). \end{aligned}$$

By substituting, we get the Bellman optimality equations:

$$\begin{aligned} v_i(k) &= \cancel{\ell\left(\vartheta \mid y_1, \ldots, y_i\right)} + \ell\left(\vartheta \mid y_1, \ldots, y_{i-1}\right) - \cancel{\ell\left(\vartheta \mid y_1, \ldots, y_i\right)} \\ &\qquad + \max_{(x_1,\ldots,x_{i-1}) \in S^{i-1}} \left[\log \phi_{1,\ldots,i-1|i-1}(x_1, \ldots, x_{i-1}) + \log p_{x_{i-1},k}\right] + \log f_{\theta_k}(y_i) \\ &= \log f_{\theta_k}(y_i) + \ell\left(\vartheta \mid y_1, \ldots, y_{i-1}\right) \\ &\qquad + \max_{\ell \in \{1,\ldots,K\}} \left[\max_{(x_1,\ldots,x_{i-2}) \in S^{i-2}} \log \phi_{1,\ldots,i-1|i-1}(x_1, \ldots, x_{i-2}, \ell) + \log p_{\ell,k}\right] \\ &= \log f_{\theta_k}(y_i) + \max_{\ell \in \{1,\ldots,K\}} \left[v_{i-1}(\ell) + \log p_{\ell,k}\right], \quad i = 2, 3, \ldots, n. \end{aligned}$$

Furthermore, we define:

$$m_i(k) = \operatorname*{argmax}_{\ell \in \{1,\ldots,K\}} \left[v_{i-1}(\ell) + \log p_{\ell,k}\right], \quad i = 2, 3, \ldots, n.$$

Therefore, we recursively calculate all the optimal value functions from start to finish. Next, we calculate the final hidden state which maximizes the final optimal value function, i.e. $x_n^* = \operatorname{argmax}_{k \in \{1,\ldots,K\}} v_n(k)$. This is the final state which maximizes the joint conditional posterior probability of the entire sequence of hidden states. Then, we want to maximize the conditional posterior probability of a sequence of hidden states which ends in state x_n^*. The previous to last state of this hidden Markov chain is given by $x_{n-1}^* = m_n\left(x_n^*\right)$. The rest of the hidden states are recursively estimated in the same manner. The optimal sequence of hidden states $x_1^*, \ldots, x_n^*$ is called a Viterbi path. The steps of the Viterbi algorithm are summarized below.

Algorithm 4 Viterbi

Input: Observations $y_1, \ldots, y_n$ and estimate ϑ.

1: Calculate the optimal value function $v_1(\cdot)$.

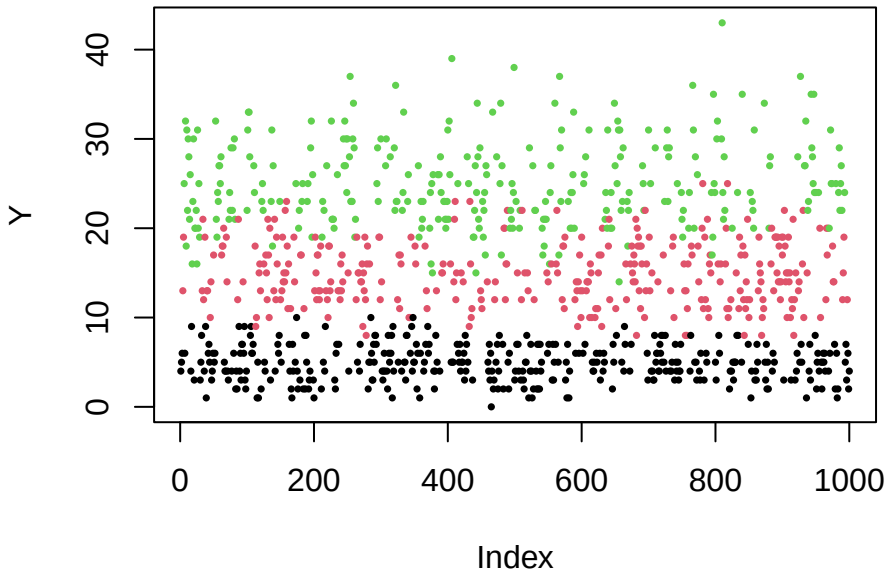
2: For $i = 2, 3, \ldots, n$, recursively calculate the rest of the optimal value functions $v_i(\cdot)$.

3: Calculate the optimal final state $x_n^* = \operatorname{argmax}_{k \in \{1,\ldots,K\}} v_n(k)$.

4: For $i = n-1, n-2, \ldots, 1$, calculate the rest of the optimal states $x_i^* = m_{i+1}\left(x_{i+1}^*\right)$.

Output: Optimal sequence of hidden states $x_1^*, \ldots, x_n^*$.

```
Viterbi = function(Y, theta, p, P) {
    n = length(Y)
    K = length(theta)
    v = matrix(0, n, K)
    m = matrix(0, n - 1, K)
    v[1, ] = log(p) + dpois(Y[1], theta, log = TRUE)
    for (i in 2:n) {
        for (k in 1:K) {
            temp = v[i - 1, ] + log(P[, k])
            m[i - 1, k] = which.max(temp)
            v[i, k] = dpois(Y[i], theta[k], log = TRUE) + max(temp)
        }
    }
    X = numeric(n)
    X[n] = which.max(v[n, ])
    for (i in (n - 1):1) {
        X[i] = m[i, X[i + 1]]
    }
    return(X)
}

XMAPjoint = Viterbi(Y, MLE$theta, p, MLE$P)
plot(Y, col = XMAPjoint, pch = 16, cex = 0.5)
```




```
table(X, XMAPjoint)
```

```
##    XMAPjoint
## X      1   2   3
##   1 347  10   0
```

```
##    2  17 301  21
##    3   0  21 283
```

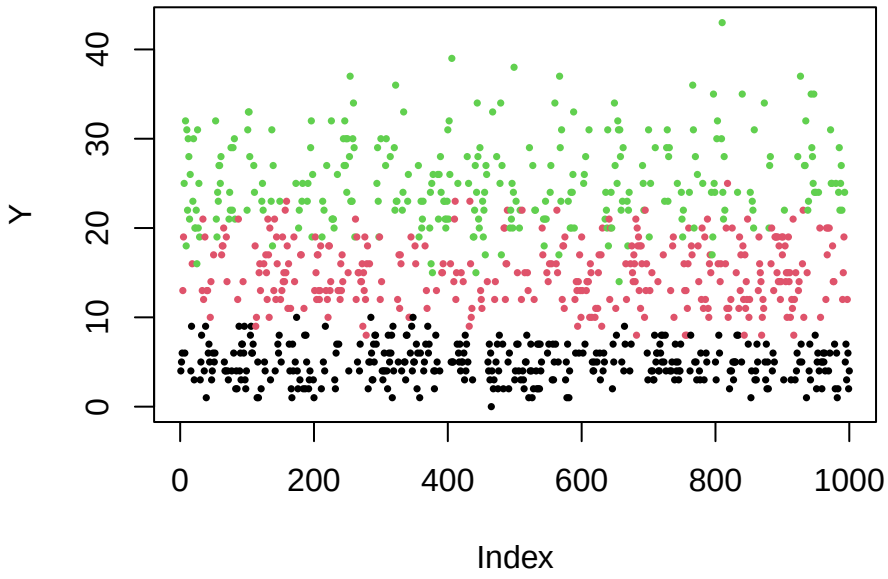
Local Decoding

Local decoding aims at the separate maximization of the marginal conditional posterior distributions of each hidden state, i.e. the marginal smoothing distributions $\phi_{i|n}(\cdot)$. Even though global decoding appears as a much stronger result than local decoding, neither of the two decodings implies the other. For example, local decoding might be preferable if our goal is to most accurately categorize only certain observations. On the other hand, local decoding might even lead to non-feasible optimal paths for the hidden Markov chain.

Suppose that we have some estimate ϑ^* of ϑ. We implement the Forward-Backward algorithm for $\vartheta = \vartheta^*$ and get the marginal smoothing distributions $\phi_{i|n}(\cdot)$ for $i = 1, 2, \ldots, n$. Then, we calculate that:

$$x_i^* = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \ \phi_{i|n}(k), \quad i = 1, 2, \ldots, n.$$

```
XMAPmarginal = apply(MLE$marginal, 1, which.max)
plot(Y, col = XMAPmarginal, pch = 16, cex = 0.5)
```




```
table(X, XMAPmarginal)
```

```
##    XMAPmarginal
## X     1   2   3
##   1 347  10   0
##   2  17 305  17
##   3   0  21 283
```

12 Viterbi Training Algorithm

The Viterbi training algorithm is an alternative to the Baum-Welch algorithm for the estimation of the unknown parameters of a HMM. We start with some initial estimate $\vartheta^{(0)}$. In place of the Forward-Backward algorithm, we

implement the Viterbi algorithm for $\vartheta = \vartheta^{(0)}$ and get a Viterbi path $x_1^{(1)}, \ldots, x_n^{(1)}$. In place of the M-step of the Baum-Welch algorithm, we maximize the complete-data log-likelihood for $x = x^{(1)}$. We repeat these two steps until the Viterbi training algorithm converges to a Viterbi path x^*.

Suppose that $(Y_i \mid X_i = k) \sim \text{Poisson}(\theta_k)$. Then, the complete-data log-likelihood is given by:

$$\ell\left(\vartheta \mid y, x\right) = \sum_{k=1}^{K} \mathbb{1}\left\{x_1 = k\right\} \log p_k + \sum_{i=2}^{n} \sum_{k=1}^{K} \sum_{\ell=1}^{K} \mathbb{1}\left\{x_{i-1} = k, x_i = \ell\right\} \log p_{k,\ell} + \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\left\{x_i = k\right\} \left[-\theta_k + y_i \log \theta_k - \log\left(y_i!\right)\right].$$

We define:

$$M_{k,\ell} = \sum_{i=2}^{n} \mathbb{1}\left\{x_{i-1} = k, x_i = \ell\right\} = \sum_{i=1}^{n-1} \mathbb{1}\left\{x_i = k, x_{i+1} = \ell\right\},$$

$$M_k = \sum_{i=1}^{n-1} \mathbb{1}\left\{x_i = k\right\}, \quad N_k = \sum_{i=1}^{n} \mathbb{1}\left\{x_i = k\right\}, \quad S_k = \sum_{i=1}^{n} \mathbb{1}\left\{x_i = k\right\} y_i.$$

First, we maximize the complete-data likelihood with respect to each row of the transition probability matrix separately. We define the following Lagrangian function:

$$\mathcal{L}(p_k, \lambda_k) = \sum_{\ell=1}^{K} M_{k,\ell} \log p_{k,\ell} - \lambda_k \left(\sum_{\ell=1}^{K} p_{k,\ell} - 1 \right).$$

By differentiating with respect to $p_{k,\ell}$, we calculate that:

$$\frac{\partial \mathcal{L}(p_k, \lambda_k)}{\partial p_{k,\ell}} = \frac{1}{p_{k,\ell}} M_{k,\ell} - \lambda_k \quad \Rightarrow \quad p_{k,\ell}^{(1)} = \frac{1}{\lambda_k} M_{k,\ell}.$$

We observe that $\sum_{\ell=1}^{K} M_{k,\ell} = M_k$. By making use of the constraint $\sum_{\ell=1}^{K} p_{k,\ell}^{(1)} = 1$, we calculate the value of the Lagrange multiplier λ_k:

$$1 = \sum_{\ell=1}^{K} p_{k,\ell}^{(1)} = \frac{1}{\lambda_k} \sum_{\ell=1}^{K} M_{k,\ell} = \frac{1}{\lambda_k} M_k \quad \Rightarrow \quad \lambda_k = M_k.$$

Therefore, we infer that:

$$p_{k,\ell}^{(1)} = \frac{M_{k,\ell}}{M_k},$$

i.e. the new estimate of $p_{k,\ell}$ is the percentage of transitions from state k which lead to state ℓ according to the current Viterbi path. Obviously, this maximization remains the same for any finite state-space HMM.

Next, we differentiate the complete-data log-likelihood with respect to θ_k:

$$\frac{\partial \ell(\vartheta \mid y, x)}{\partial \theta_k} = \sum_{i=1}^{n} \mathbb{1}\left\{x_i = k\right\} \left(-1 + \frac{y_i}{\theta_k}\right) \quad \Rightarrow \quad \theta_k^{(1)} = \frac{S_k}{N_k},$$

i.e. the new estimate of θ_k is the sample average of the observations originating from state k according to the current Viterbi path.

In contrast with the Baum-Welch algorithm, which weighs all possible paths of the hidden Markov chain according to their conditional posterior probability in order to provide new estimates of the unknown parameters of the HMM,

the Viterbi training algorithm only takes the path with the maximum possible conditional posterior probability into account in each iteration. As a result, the Viterbi training algorithm has a much smaller computational cost compared to the Baum-Welch algorithm, but without offering any guarantee that the final estimate of ϑ is indeed going to maximize the observed-data likelihood of the sample.

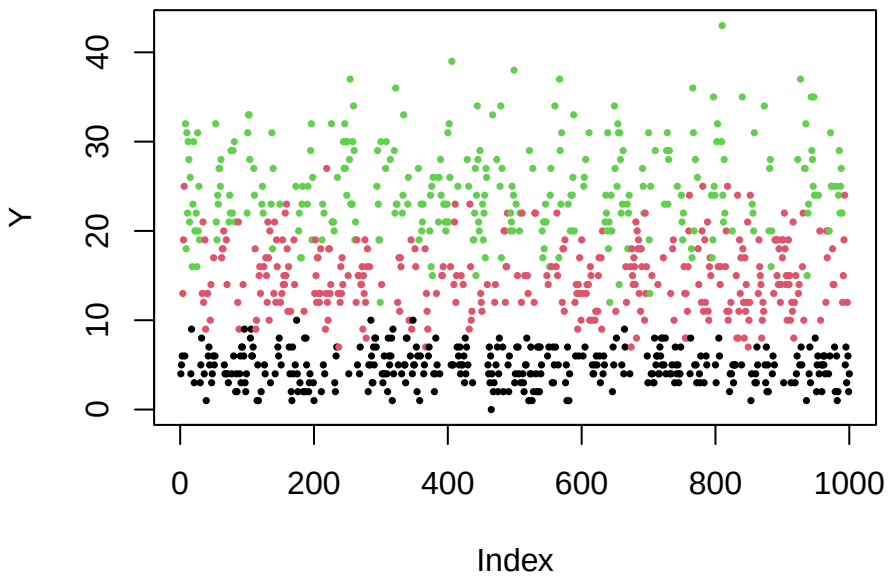
```
VTpoisHMM = function(Y, theta, p, P) {
    K = length(theta)
    steps = 1
    loglik = forward(Y, theta, p, P)$loglik
    Xprev = numeric(n)
    X = Viterbi(Y, theta, p, P)
    while (sum(X != Xprev) > 0) {
        steps = steps + 1
        x = factor(X, levels = 1:K)
        M = matrix(aggregate(numeric(n - 1), data.frame(x[-n], x[-1]), length,
            drop = FALSE)[, 3], K)
        M[is.na(M)] = 0
        P = M/rowSums(M)
        N = table(x)
        S = aggregate(Y ~ x, FUN = sum, drop = FALSE)[, 2]
        S[is.na(S)] = 0
        theta = S/N
        loglik[steps] = forward(Y, theta, p, P)$loglik
        Xprev = X
        X = Viterbi(Y, theta, p, P)
    }
    return(list(theta = theta, P = P, loglik = loglik))
}

VT = VTpoisHMM(Y, mean(Y) + sd(Y) * ((1 - K)/2):((K - 1)/2), p, matrix(1, K,
    K)/K)
print(VT$theta)
```

```
## x
##         1         2         3
##  4.750708 14.899110 24.745161
```

```
print(VT$P)
```

```
##           [,1]      [,2]       [,3]
## [1,] 0.5823864 0.1875000 0.23011364
## [2,] 0.2581602 0.7091988 0.03264095
## [3,] 0.1935484 0.1032258 0.70322581
```

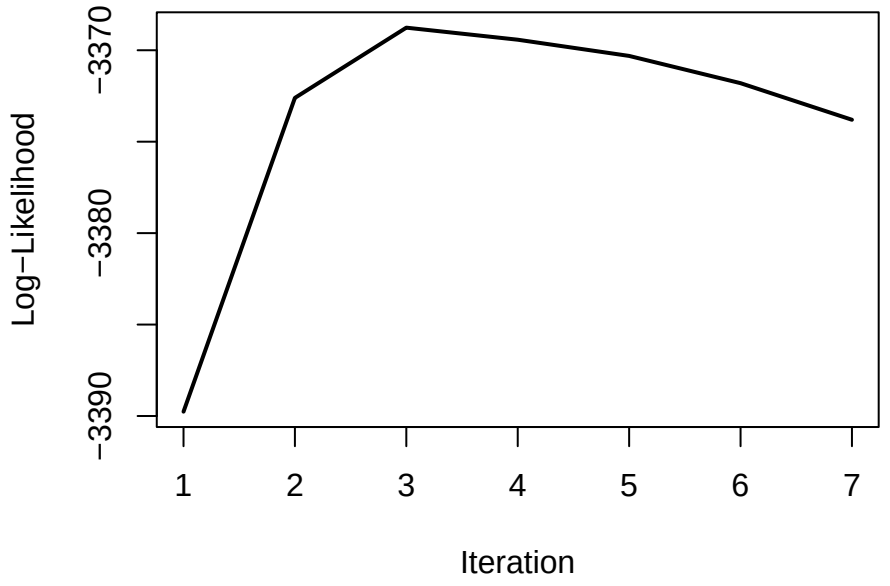
```
XMAPjoint = Viterbi(Y, VT$theta, p, VT$P)
plot(Y, col = XMAPjoint, pch = 16, cex = 0.5)
```




```
table(X, XMAPjoint)
```

```
##    XMAPjoint
## X      1   2   3
##   1 336  21   0
##   2  17 298  24
##   3   0  18 286
```

```
plot(VT$loglik[-1], type = "l", xlab = "Iteration", ylab = "Log-Likelihood",
    lwd = 2)
```



13 Application of the Gibbs Sampler to HMMs

Suppose that the rows of the transition probability matrix are a priori independent with conditionally conjugate Dirichlet(α) prior distributions and the following probability density functions:

$$f(p_k) = \frac{\Gamma(K\alpha)}{[\Gamma(\alpha)]^K} \prod_{\ell=1}^{K} p_{k,\ell}^{\alpha-1} \propto \prod_{\ell=1}^{K} p_{k,\ell}^{\alpha-1}.$$

Then, the conditional posterior distribution of the transition probability matrix P is given by:

$$\begin{aligned} f(P \mid x) &\propto \prod_{k=1}^{K} f(p_k) \prod_{i=2}^{n} \mathbb{P}(X_i = x_i \mid X_{i-1} = x_{i-1}, P) \\ &\propto \prod_{k=1}^{K} \prod_{\ell=1}^{K} \left[p_{k,\ell}^{\alpha-1} \prod_{i=2}^{n} p_{k,\ell}^{\mathbb{1}_{\{x_{i-1}=k, x_i=\ell\}}} \right] = \prod_{k=1}^{K} \prod_{\ell=1}^{K} p_{k,\ell}^{M_{k,\ell}+\alpha-1}, \end{aligned}$$

i.e. the rows $p_k \mid x \sim \text{Dirichlet}\left(M_{k,1}+\alpha, M_{k,2}+\alpha, \ldots, M_{k,K}+\alpha\right)$ of the transition probability matrix are a posteriori independent, where $M_{k,\ell} = \sum_{i=2}^{n} \mathbb{1}_{\{x_{i-1}=k, x_i=\ell\}}$ for $k, \ell = 1, 2, \ldots, K$. As a special case, we can consider Jeffreys' prior for the rows of the transition probability matrix, which results for $\alpha = 0.5$.

Next, we consider the conditionally conjugate Gamma(β, λ) prior distribution for the parameter θ_k with the following probability density function:

$$f(\theta_k) = \frac{\lambda^\beta}{\Gamma(\beta)} \theta_k^{\beta-1} e^{-\lambda\theta_k} \propto \theta_k^{\beta-1} e^{-\lambda\theta_k}.$$

Then, the conditional posterior distribution of θ_k is given by:

$$f(\theta_k \mid x, y) \propto f(\theta_k) \prod_{i=1}^{n} f(y_i \mid x_i, \theta_k) \propto \theta_k^{\beta-1} e^{-\lambda\theta_k} \prod_{i=1}^{n} \left(e^{-\theta_k} \frac{\theta_k^{y_i}}{y_i!} \right)^{\mathbb{1}_{\{x_i=k\}}} \propto \theta_k^{S_k+\beta-1} e^{-(N_k+\lambda)\theta_k},$$

i.e. $\theta_k \mid x, y \sim \text{Gamma}(S_k + \beta, N_k + \lambda)$, where $S_k = \sum_{i=1}^{n} y_i \mathbb{1}_{\{x_i=k\}}$ for $k = 1, 2, \ldots, K$. As a special case, we can consider the improper Jeffreys' prior for θ_k, which is given by $f(\theta_k) \propto \theta_k^{-0.5}$ and results for $\beta = 0.5$, $\lambda = 0$.

Finally, we know that the conditional posterior distribution of the hidden Markov chain is given by:

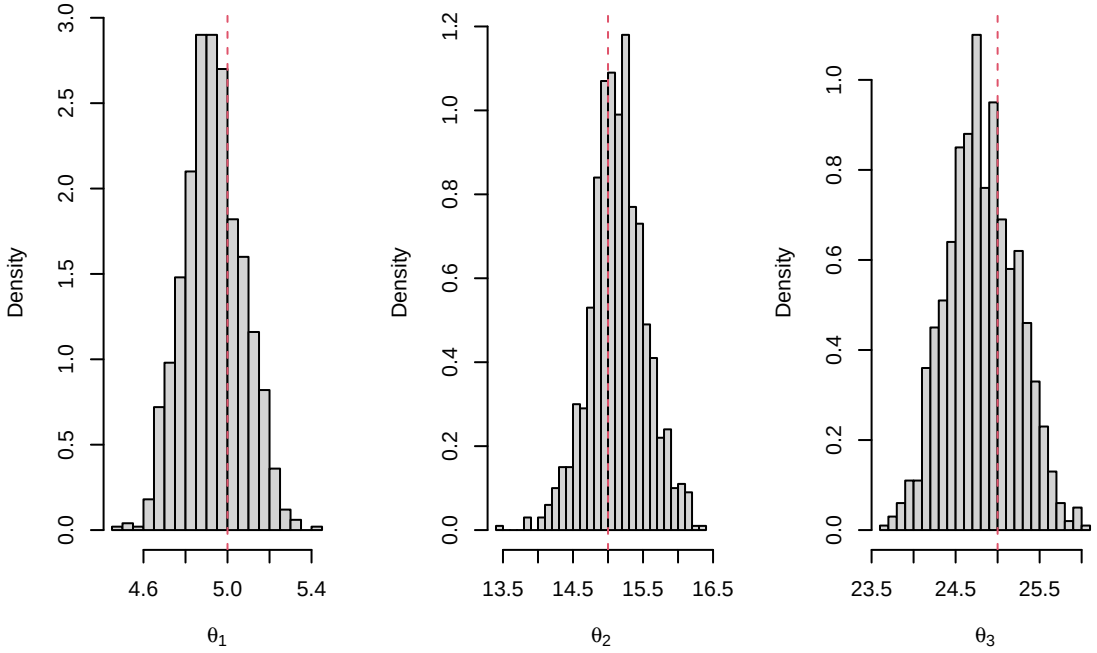
$$\mathbb{P}(X = x \mid y, \vartheta) = \phi_{n|n}(x_n) \prod_{i=1}^{n-1} B_i(x_i, x_{i+1}).$$

Therefore, we first implement the Forward Filtering algorithm to calculate the filtering distributions $\phi_{i|i}(\cdot)$ for $i = 1, 2, \ldots, n$. Then, we simulate the final state X_n of the hidden Markov chain from the final filtering distribution $\phi_{n|n}(\cdot)$. Lastly, we simulate state X_i from the backward transition distribution $B_i(\cdot, x_{i+1}) = \phi_{i|i}(\cdot) p_{\cdot, x_{i+1}}$ for $i = n-1, n-2, \ldots, 1$.

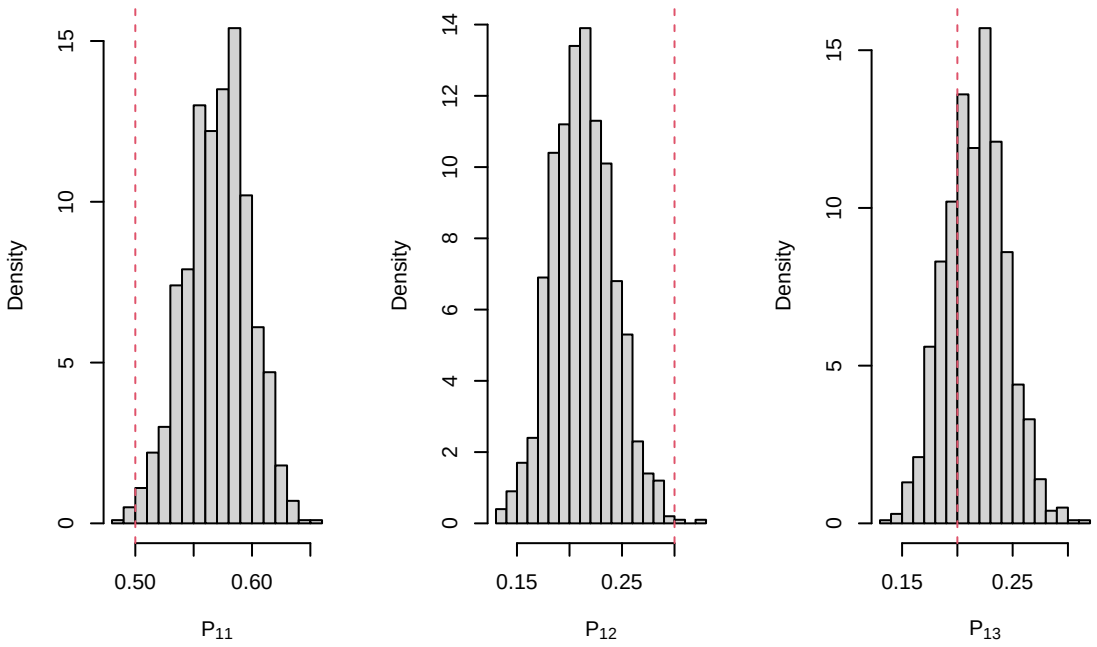
```
backward = function(filter, P) {
    n = dim(filter)[1]
    K = dim(P)[1]
    X = numeric(n)
    X[n] = sample(K, 1, prob = filter[n, ])
    for (i in (n - 1):1) {
```

```
        logB = log(filter[i, ]) + log(P[, X[i + 1]])
        B = exp(logB - max(logB))
        X[i] = sample(K, 1, prob = B)
    }
    return(X)
}

MCMCpoisHMM = function(Y, theta0, p, P0, alpha, beta, lambda, niter, nburn) {
    n = length(Y)
    K = length(theta)
    theta = matrix(0, niter, K)
    P = array(0, c(K, K, niter))
    X = matrix(0, niter, n)
    theta[1, ] = theta0
    P[, , 1] = P0
    X[1, ] = backward(forward(Y, theta[1, ], p, P[, , 1])$filter, P[, , 1])
    for (j in 2:niter) {
        x = factor(X[j - 1, ], levels = 1:K)
        M = aggregate(numeric(n - 1), data.frame(x[-n], x[-1]), length, drop = FALSE)[,
            3]
        P[, , j] = rgamma(K^2, M + alpha)
        P[, , j] = P[, , j]/rowSums(P[, , j])
        N = table(x)
        S = aggregate(Y ~ x, FUN = sum, drop = FALSE)[, 2]
        S[is.na(S)] = 0
        theta[j, ] = rgamma(K, S + beta, N + lambda)
        X[j, ] = backward(forward(Y, theta[j, ], p, P[, , j])$filter, P[, ,
            j])
        I = order(theta[j, ])
        theta[j, ] = theta[j, I]
        P[, , j] = P[I, I, j]
        X[j, ] = I[X[j, ]]
    }
    return(list(theta = theta[-(1:nburn), ], P = P[, , -(1:nburn)], X = X[-(1:nburn),
        ]))
}

posterior = MCMCpoisHMM(Y, mean(Y) + sd(Y) * ((1 - K)/2):((K - 1)/2), p, matrix(1,
    K, K)/K, 0.5, 0.5, 0, 2000, 1000)
par(mfrow = c(1, 3))
hist(posterior$theta[, 1], "FD", freq = FALSE, main = NA, xlab = expression(theta[1]))
abline(v = theta[1], col = 2, lty = 2)
hist(posterior$theta[, 2], "FD", freq = FALSE, main = NA, xlab = expression(theta[2]))
```

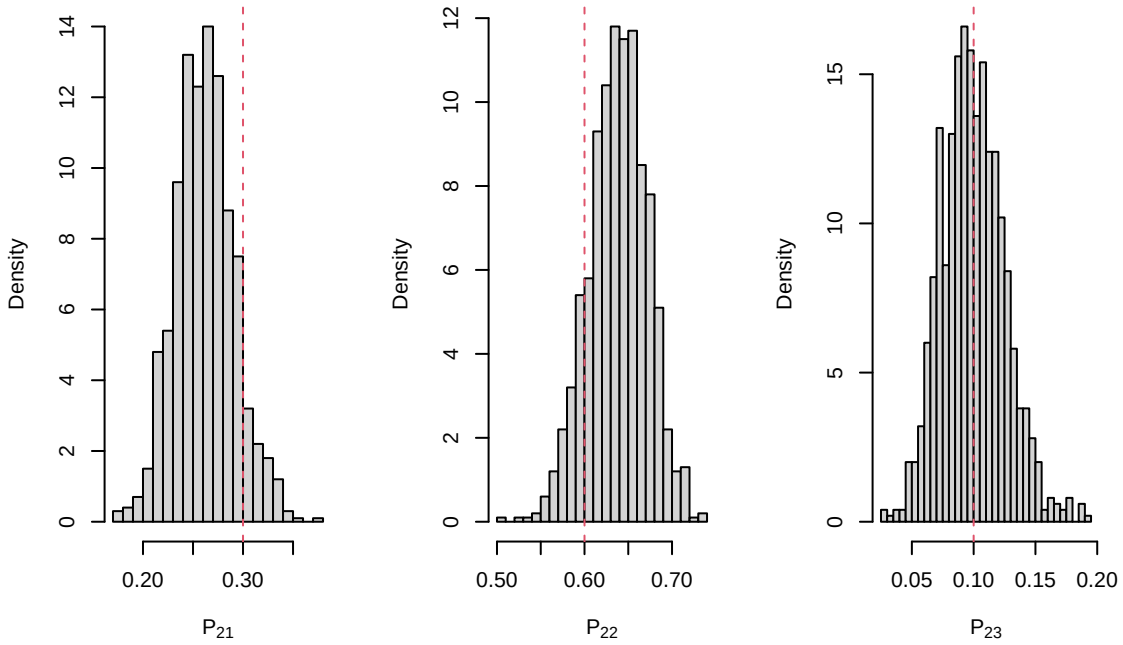
```
abline(v = theta[2], col = 2, lty = 2)
hist(posterior$theta[, 3], "FD", freq = FALSE, main = NA, xlab = expression(theta[3]))
abline(v = theta[3], col = 2, lty = 2)
```



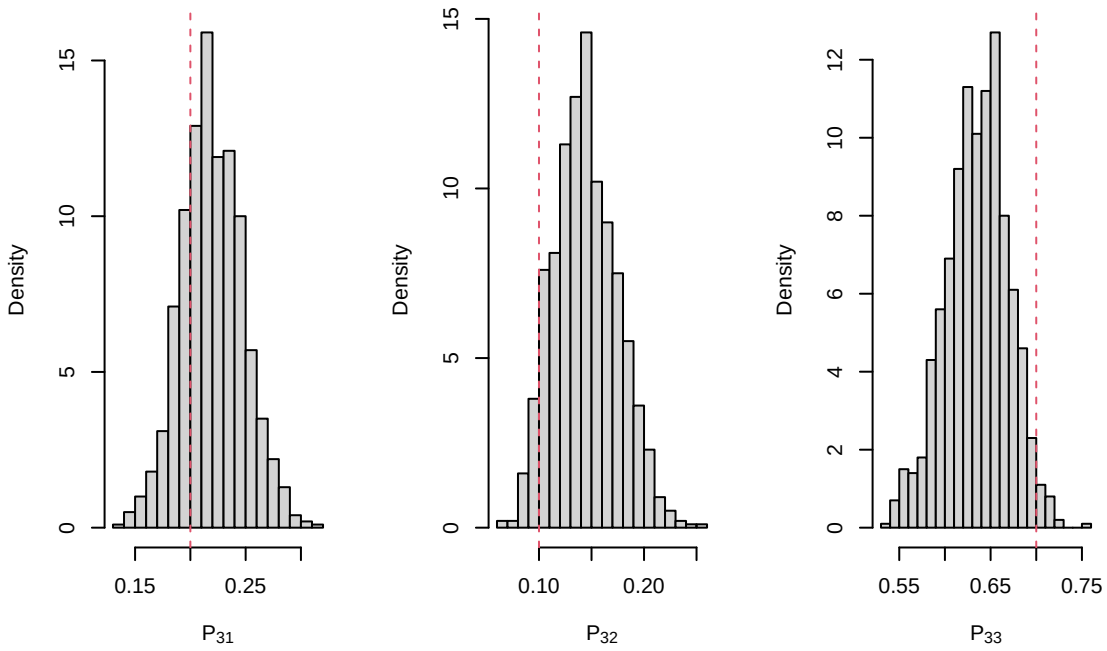

```
hist(posterior$P[1, 1, ], "FD", freq = FALSE, main = NA, xlab = expression(P[11]))
abline(v = P[1, 1], col = 2, lty = 2)
hist(posterior$P[1, 2, ], "FD", freq = FALSE, main = NA, xlab = expression(P[12]))
abline(v = P[1, 2], col = 2, lty = 2)
hist(posterior$P[1, 3, ], "FD", freq = FALSE, main = NA, xlab = expression(P[13]))
abline(v = P[1, 3], col = 2, lty = 2)
```



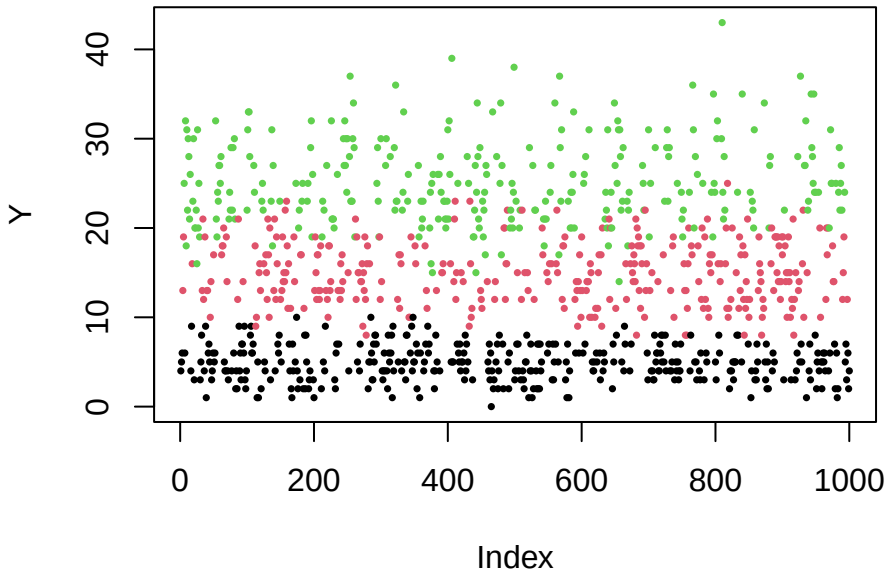
```
hist(posterior$P[2, 1, ], "FD", freq = FALSE, main = NA, xlab = expression(P[21]))
abline(v = P[2, 1], col = 2, lty = 2)
hist(posterior$P[2, 2, ], "FD", freq = FALSE, main = NA, xlab = expression(P[22]))
abline(v = P[2, 2], col = 2, lty = 2)
hist(posterior$P[2, 3, ], "FD", freq = FALSE, main = NA, xlab = expression(P[23]))
abline(v = P[2, 3], col = 2, lty = 2)
```




```
hist(posterior$P[3, 1, ], "FD", freq = FALSE, main = NA, xlab = expression(P[31]))
abline(v = P[3, 1], col = 2, lty = 2)
hist(posterior$P[3, 2, ], "FD", freq = FALSE, main = NA, xlab = expression(P[32]))
abline(v = P[3, 2], col = 2, lty = 2)
hist(posterior$P[3, 3, ], "FD", freq = FALSE, main = NA, xlab = expression(P[33]))
abline(v = P[3, 3], col = 2, lty = 2)
```




```
XMAP = apply(posterior$X, 2, function(x) {
    which.max(table(factor(x, levels = 1:K)))
})
plot(Y, col = XMAP, pch = 16, cex = 0.5)
```




```
table(X, XMAP)
```

```
##    XMAP
## X     1   2   3
##   1 347  10   0
##   2  17 305  17
##   3   0  21 283
```