

A Multiple Changepoint model for Count data

A. Taylan Cemgil,
Department of Computer Engineering, Boğaziçi University
34342 Bebek, Istanbul, Turkey
taylan.cemgil@boun.edu.tr

December 4, 2009

Abstract

This is a tutorial on exact inference in multiple changepoint models. We describe a model for a time series of count data.

1 Multiple Changepoint model

A switching state space model is a hierarchical hidden Markov model of form

$$\begin{aligned}\lambda_0 &\sim p(\lambda_0) \\ r_1 &\sim p(r_1) \\ r_t &\sim p(r_t|r_{t-1}) \\ \lambda_t &\sim p(\lambda_t|r_t, \lambda_{t-1}) \\ x_t &\sim p(x_t|r_t, \lambda_t)\end{aligned}$$

Here r_t is a Markov chain on a discrete state space D_r . Conditioned on r , λ_t is also a Markov chain. The state space of λ_t is denoted by D_λ . We say that λ_t is controlled or modulated by r_t .

If D_λ is finite, this model reduces to an ordinary finite state space HMM on $D = D_\lambda \times D_r$ and inference can be carried out by the forward backward algorithm. However, if D_λ is continuous, an exact forward backward algorithm can not be implemented in general. This is due to the fact that the prediction density $p(r_t, \lambda_t|x_t)$ needs to be computed by integrating over λ_{t-1} and summing over r_{t-1} . The summation over r_{t-1} renders the prediction density a mixture model where the number of mixture components grows exponentially with t . Exact inference is prohibitive and one has to resort to one of the several existing approximation algorithms [cite].

A multiple changepoint model is a switching state space model with a special structure. Here, the latent chain r_t is interpreted as a *regime label*. For a particular configuration of $c \in D_r$, the transition density becomes independent of λ_{t-1} , i.e.,

$$p(\lambda_t|r_t = c, \lambda_{t-1}) = f(\lambda_t)$$

In this case, a changepoint occurs and λ_t initialises itself independently from λ_{t-1} . In some circumstances, this forgetting property can be exploited to derive a polynomial time exact algorithm.

In this tutorial, we will deal with the following model: At each time t , we observe x_t . We assume that x_t is Poisson with unknown intensity λ . The intensity is constant, but at certain unknown times t , it jumps to a new value. The indicator variables r_τ denote whether time τ is a changepoint or not. Our task is finding the posterior probability of a change and the associated intensity levels for each region between two consecutive changepoints. In Figure 1, we illustrate a typical realisation from such a scenario. The data is generated by the following model.

$$\begin{aligned}\lambda_0 &\sim \mathcal{G}(\lambda_0; a_0, b_0) \\ r_t &\sim \mathcal{BE}(r_t; \pi_1) & \pi_0 = 1 - \pi_1 \\ \lambda_t|r_t, \lambda_{t-1} &\sim [r_t = 0] \delta(\lambda_t - \lambda_{t-1}) + [r_t = 1] \mathcal{G}(\lambda_t; \nu, B) \\ x_t|\lambda_t &\sim \mathcal{PO}(x_t; \lambda_t)\end{aligned}$$

Here, $r_t \in D_r = \{0, 1\}$ and $\lambda_t \in \mathbb{R}^+$. The symbols \mathcal{G} , \mathcal{BE} and \mathcal{PO} denote the gamma, Bernoulli and the Poisson distribution respectively

$$\begin{aligned}\mathcal{G}(\lambda; a, b) &= \exp((a-1)\log\lambda - b\lambda - \log\Gamma(a) + a\log b) \\ \mathcal{BE}(r; \pi) &= \exp(r\log\pi + (1-r)\log(1-\pi)) \\ \mathcal{PO}(x; \lambda) &= \exp(x\log\lambda - \lambda - \log\Gamma(x+1))\end{aligned}$$

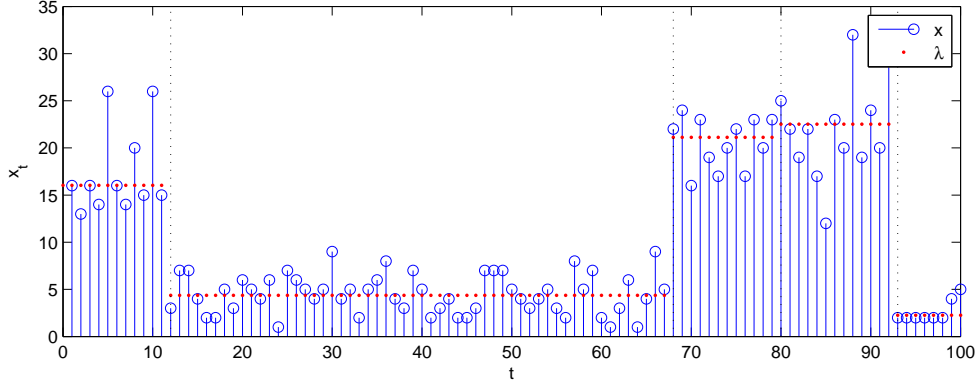


Figure 1: A realisation from the multiple changepoint model. The vertical dots denote the location of changepoint where $r_t = 1$.

1.1 Forward Backward algorithm

The forward backward algorithm is a well known algorithm for computing the marginals of form $p(r_t, \lambda_t | x_{1:T})$.

We define the following forward messages

$$\begin{aligned}\alpha_{0|0}(r_0, \lambda_0) &= p(\lambda_0) \\ & \quad t = 1 \dots M \\ \alpha_{t|t-1}(r_t, \lambda_t) &= p(r_t, \lambda_t, x_{1:t-1}) \\ \alpha_{t|t}(r_t, \lambda_t) &= p(r_t, \lambda_t, x_{1:t})\end{aligned}$$

and the backward messages

$$\begin{aligned}\beta_{M|M} &= p(x_M | r_M, \lambda_M) \\ & \quad t = M - 1 \dots 1 \\ \beta_{t|t+1}(r_t, \lambda_t) &= p(x_{t+1:M} | r_t, \lambda_t) \\ \beta_{t|t}(r_t, \lambda_t) &= p(x_{t:M} | r_t, \lambda_t)\end{aligned}$$

The messages can be computed by the following recursion

$$\begin{aligned}\alpha_{t|t-1}(r_t, \lambda_t) &= \sum_{r_{t-1}} \int d\lambda_{t-1} p(r_t, \lambda_t | r_{t-1}, \lambda_{t-1}) \alpha_{t-1|t-1}(r_{t-1}, \lambda_{t-1}) \\ \alpha_{t|t}(r_t, \lambda_t) &= p(x_t | r_t, \lambda_t) \alpha_{t|t-1}(r_t, \lambda_t)\end{aligned}$$

$$\begin{aligned}\beta_{t|t+1}(r_t, \lambda_t) &= p(x_{t+1:M} | r_t, \lambda_t) \\ &= \sum_{r_{t+1}} \int d\lambda_{t+1} p(r_{t+1}, \lambda_{t+1} | r_t, \lambda_t) \beta_{t+1|t+1}(r_{t+1}, \lambda_{t+1}) \\ \beta_{t|t}(r_t, \lambda_t) &= p(r_t, \lambda_t, x_{t:M}) \\ &= p(x_t | r_t, \lambda_t) \beta_{t|t+1}(r_t, \lambda_t)\end{aligned}$$

1.2 Smoothing

The posterior marginals can be simply obtained by multiplying the forward and backward messages:

$$\begin{aligned}p(r_t, \lambda_t | x_{1:M}) &\propto p(x_{1:M}, r_t, \lambda_t) \\ &= p(x_{1:t-1}, r_t, \lambda_t) p(x_{t:M} | r_t, \lambda_t, x_{1:t-1}) \\ &= p(x_{1:t-1}, r_t, \lambda_t) p(x_{t:M} | r_t, \lambda_t) \\ &= \alpha_{t|t-1} \beta_{t|t}\end{aligned}$$

2 Implementation of the message passing algorithm

The α and β messages are potentials over λ and r . As r is discrete and λ is continuous, we need to work with mixtures of Gamma distributions.

2.1 Gamma potentials

We will represent a Gamma potential

$$\phi(\lambda) = e^c \mathcal{G}(\lambda; a, b)$$

via the triple

$$(a, b, c)$$

Note that, in this representation, the normalisation constant is given as

$$c = \log \int d\lambda \phi(\lambda)$$

Similarly, we can represent a mixture

$$\phi(\lambda) = \sum_{i=1}^M w_i \mathcal{G}(\lambda; a_i, b_i)$$

as $\{(a_1, b_1, \log w_1), (a_2, b_2, \log w_2), \dots, (a_M, b_M, \log w_M)\}$. This will be simply a $M \times 3$ array of parameters.

2.2 Products of Gamma potentials

Note that the update equations correspond to multiplication of a message with the observation model $p(x_t | \lambda_t) = \mathcal{PO}(x_t; \lambda_t)$. A property of the Poisson distribution is that

$$\begin{aligned} \mathcal{PO}(x; \lambda) &= +x \log \lambda - \lambda - \log \Gamma(x + 1) \\ &= +(x + 1 - 1) \log \lambda - \lambda - \log \Gamma(x + 1) = \mathcal{G}(\lambda; x + 1, 1) \end{aligned}$$

Hence, the update equation requires multiplication of two gamma potentials. A nice property is that the product is also a gamma potential, as derived in the appendix A:

$$(a_1, b_1, c_1) \times (a_2, b_2, c_2) = (a_1 + a_2 - 1, b_1 + b_2, c_1 + c_2 + g(a_1, b_1, a_2, b_2))$$

where

$$g(a_1, b_1, a_2, b_2) \equiv \log \Gamma(a_1 + a_2 - 1) - \log \Gamma(a_1) - \log \Gamma(a_2) + \log(b_1 + b_2) + a_1 \log(b_1 / (b_1 + b_2)) + a_2 \log(b_2 / (b_1 + b_2))$$

The identity element is the (improper) Gamma distribution with canonical parameters $(1, 0, 0)$.

2.3 Forward Pass

We define the parameters

$$\begin{aligned} l_0 &= \log \pi_0 = \log p(r_t = 0) \\ l_1 &= \log \pi_1 = \log p(r_t = 1) \end{aligned}$$

The messages α and β will be represented as mixtures where each component $k = 0, \dots, K - 1$ will be denoted by α^k or β^k . We also use the notation

$$\alpha_{t|t-1;c} = \alpha_{t|t-1}(\lambda_t, r_t = c)$$

We will illustrate a few steps of the recursion. To start the recursion, we define

$$\begin{aligned} \alpha_{0|0}(\lambda_0) &= p(\lambda_0) \\ \alpha_{0|0} &= (a_0, b_0, 0) \\ (a_{0|0}^0, b_{0|0}^0, c_{0|0}^0) &= (a_0, b_0, 0) \end{aligned}$$

One step prediction is given as

$$\begin{aligned}
\alpha_{1|0}(\lambda_1, r_1) &= \int d\lambda_0 p(\lambda_1, r_1 | \lambda_0) \alpha_{0|0}(\lambda_0) \\
\alpha_{1|0;1} &= \alpha_{1|0}(\lambda_1, r_1 = 1) = (\nu, B, l_1) \\
\alpha_{1|0;0} &= \alpha_{1|0}(\lambda_1, r_1 = 0) = (a_0, b_0, l_0) \\
(a_{1|0}^0, b_{1|0}^0, c_{1|0}^0) &= (\nu, B, l_1) \\
(a_{1|0}^1, b_{1|0}^1, c_{1|0}^1) &= (a_{0|0}^0, b_{0|0}^0, l_0 + c_{0|0}^0)
\end{aligned}$$

The update step can be calculated by multiplying each component of the gamma mixture with the observation model.

$$\alpha_{1|1}(\lambda_1, r_1) = p(x_1, \lambda_1, r_1) = \alpha_{1|0}(\lambda_1, r_1) p(x_1 | \lambda_1)$$

$$\begin{aligned}
k &= 0, 1 \\
(a_{1|1}^k, b_{1|1}^k, c_{1|1}^k) &= (a_{1|0}^k, b_{1|0}^k, c_{1|0}^k) \times (x_1 + 1, 1, 0) \\
&= (a_{1|0}^k + x_1, b_{1|0}^k + 1, c_{1|0}^k + g(a_{1|0}^k, b_{1|0}^k, x_1 + 1, 1))
\end{aligned}$$

The predictive distribution

$$\begin{aligned}
\alpha_{2|1}(\lambda_2, r_2) &= p(x_1, \lambda_2, r_2) = \sum_{r_1} \int d\lambda_1 p(\lambda_2, r_2 | \lambda_1) \alpha_{1|1}(\lambda_1, r_1) \\
&= \sum_{r_1} \int d\lambda_1 ([r_2 = 0] \delta(\lambda_2 - \lambda_1) p(r_2 = 0) + [r_2 = 1] \mathcal{G}(\lambda_2; \nu, B) p(r_2 = 1)) \alpha_{1|1}(\lambda_1, r_1)
\end{aligned}$$

$$\begin{aligned}
\alpha_{2|1;1} &= [r_2 = 1] \mathcal{G}(\lambda_2; \nu, B) p(r_2 = 1) \sum_{r_1} \int d\lambda_1 \alpha_{1|1}(\lambda_1, r_1) \\
&= [r_2 = 1] \mathcal{G}(\lambda_2; \nu, B) \exp(l_1) \sum_{\tau=0}^1 \exp(c_{1|1}^\tau)
\end{aligned}$$

This integral evaluates to the sum of normalising coefficients of each potential, hence

$$\begin{aligned}
(a_{2|1}^0, b_{2|1}^0, c_{2|1}^0) &= (\nu, B, l_1 + \log \sum_{\tau=0}^1 \exp(c_{1|1}^\tau)) \\
\alpha_{2|1;0} &= \sum_{r_1} \int d\lambda_1 [r_2 = 0] \delta(\lambda_2 - \lambda_1) p(r_2 = 0) \alpha_{1|1}(\lambda_1, r_1) \\
&= [r_2 = 0] \exp(l_0) \sum_{r_1} \alpha_{1|1}(\lambda_2, r_1) \\
(a_{2|1}^1, b_{2|1}^1, c_{2|1}^1) &= (a_{1|1}^0, b_{1|1}^0, l_0 + c_{1|1}^0) \\
(a_{2|1}^2, b_{2|1}^2, c_{2|1}^2) &= (a_{1|1}^1, b_{1|1}^1, l_0 + c_{1|1}^1)
\end{aligned}$$

The prediction step at time t is

$$\alpha_{t|t-1}(\lambda_t, r_t) = p(x_{1:t-1}, \lambda_t, r_t) = \int d\lambda_{t-1} \sum_{r_{t-1}} p(\lambda_t, r_t | \lambda_{t-1}) \alpha_{t-1|t-1}(\lambda_{t-1}, r_{t-1})$$

$$\begin{aligned}
(a_{t|t-1}^0, b_{t|t-1}^0, c_{t|t-1}^0) &= (\nu, B, l_1 + \log \sum_{\tau=0}^{t-1} \exp(c_{t-1|t-1}^\tau)) \\
&\quad \tau = 1 \dots t \\
(a_{t|t-1}^\tau, b_{t|t-1}^\tau, c_{t|t-1}^\tau) &= (a_{t-1|t-1}^{\tau-1}, b_{t-1|t-1}^{\tau-1}, l_0 + c_{t-1|t-1}^{\tau-1})
\end{aligned}$$

The update step at time t is

$$\alpha_{t|t}(\lambda_t, r_t) = p(x_{1:t}, \lambda_t, r_t) = \alpha_{t|t-1}(\lambda_t) p(x_t | \lambda_t)$$

$$\begin{aligned} \tau &= 0 \dots t \\ (\hat{a}_{t|t}^\tau, \hat{b}_{t|t}^\tau, \hat{c}_{t|t}^\tau) &= (\hat{a}_{t|t-1}^\tau, \hat{b}_{t|t-1}^\tau, \hat{c}_{t|t-1}^\tau)(x_t + 1, 1, 0) \end{aligned}$$

2.4 Backward Pass

The backward pass is initialised as follows

$$\begin{aligned} \beta_{M|M+1}(\lambda_M, r_M) &= 1 \\ (\hat{a}_{M|M+1}^0, \hat{b}_{M|M+1}^0, \hat{c}_{M|M+1}^0) &= (1, 0, 0) \end{aligned}$$

$$\begin{aligned} \beta_{M|M}(\lambda_M, r_M) &= p(x_M | \lambda_M, r_M) \beta_{M|M+1}(\lambda_M, r_M) \\ (\hat{a}_{M|M}^0, \hat{b}_{M|M}^0, \hat{c}_{M|M}^0) &= (1, 0, 0) \times (x_M + 1, 1, 0) = (x_M + 1, 1, 0) \end{aligned}$$

We define $c[(a, b, c)] = c$

$$\begin{aligned} \beta_{M-1|M}(\lambda_{M-1}, r_{M-1}) &= \int d\lambda_M \sum_{r_M} p(\lambda_M, r_M | \lambda_{M-1}) \beta_{M|M} \\ \beta_{M-1|M}(\lambda_{M-1}, r_{M-1}) &= \int d\lambda_M p(r_M = 1) \mathcal{G}(\lambda_3; \nu, B) \beta_{M|M} + \int d\lambda_M p(r_M = 0) \delta(\lambda_M - \lambda_{M-1}) \beta_{M|M} \\ (\hat{a}_{M-1|M}^0, \hat{b}_{M-1|M}^0, \hat{c}_{M-1|M}^0) &= (1, 0, l_1 + c((\nu, B, 0) \times (\hat{a}_{M|M}^0, \hat{b}_{M|M}^0, \hat{c}_{M|M}^0))) \\ (\hat{a}_{M-1|M}^1, \hat{b}_{M-1|M}^1, \hat{c}_{M-1|M}^1) &= (\hat{a}_{M|M}^0, \hat{b}_{M|M}^0, l_0 + \hat{c}_{M|M}^0) \end{aligned}$$

The update step can be implemented as follows:

$$\begin{aligned} \beta_{t|t} &= \beta_{t|t+1} p(x_t | \lambda_t) = p(x_{t:M} | \lambda_t, r_t) \\ \tau &= 0 \dots M - t \\ (\hat{a}_{t|t}^\tau, \hat{b}_{t|t}^\tau, \hat{c}_{t|t}^\tau) &= (\hat{a}_{t|t+1}^\tau, \hat{b}_{t|t+1}^\tau, \hat{c}_{t|t+1}^\tau)(x_t + 1, 1, 0) \\ (\hat{a}_{t|t}^0, \hat{b}_{t|t}^0, \hat{c}_{t|t}^0) &= (x_t + 1, 1, l_1 + \log \sum_{\tau=0}^{M-(t+1)} \exp(c((\nu, B, 0)(\hat{a}_{t+1|t+1}^\tau, \hat{b}_{t+1|t+1}^\tau, \hat{c}_{t+1|t+1}^\tau)))) \\ \tau &= 1 \dots M - t \\ (\hat{a}_{t|t}^\tau, \hat{b}_{t|t}^\tau, \hat{c}_{t|t}^\tau) &= (\hat{a}_{t+1|t+1}^{\tau-1}, \hat{b}_{t+1|t+1}^{\tau-1}, l_0 + \hat{c}_{t+1|t+1}^{\tau-1})(x_t + 1, 1, 0) \end{aligned}$$

The backward propagation step is:

$$\begin{aligned} \beta_{t-1|t} &= \int d\lambda_t \sum_{r_t} p(\lambda_t, r_t | \lambda_{t-1}) \beta_{t|t} \\ &= p(x_{t:M} | \lambda_{t-1}, r_{t-1}) \\ (\hat{a}_{t-1|t}^0, \hat{b}_{t-1|t}^0, \hat{c}_{t-1|t}^0) &= (1, 0, l_1 + \log \sum_{\tau=0}^{M-t} \exp(c((\nu, B, 0)(\hat{a}_{t|t}^\tau, \hat{b}_{t|t}^\tau, \hat{c}_{t|t}^\tau)))) \\ \tau &= 1 \dots M - t \\ (\hat{a}_{t-1|t}^\tau, \hat{b}_{t-1|t}^\tau, \hat{c}_{t-1|t}^\tau) &= (\hat{a}_{t|t}^{\tau-1}, \hat{b}_{t|t}^{\tau-1}, l_0 + \hat{c}_{t|t}^{\tau-1}) \end{aligned}$$

2.5 Results

An example run is shown in the figure 2. In 3, we show what happens if data is generated from a slightly different model.

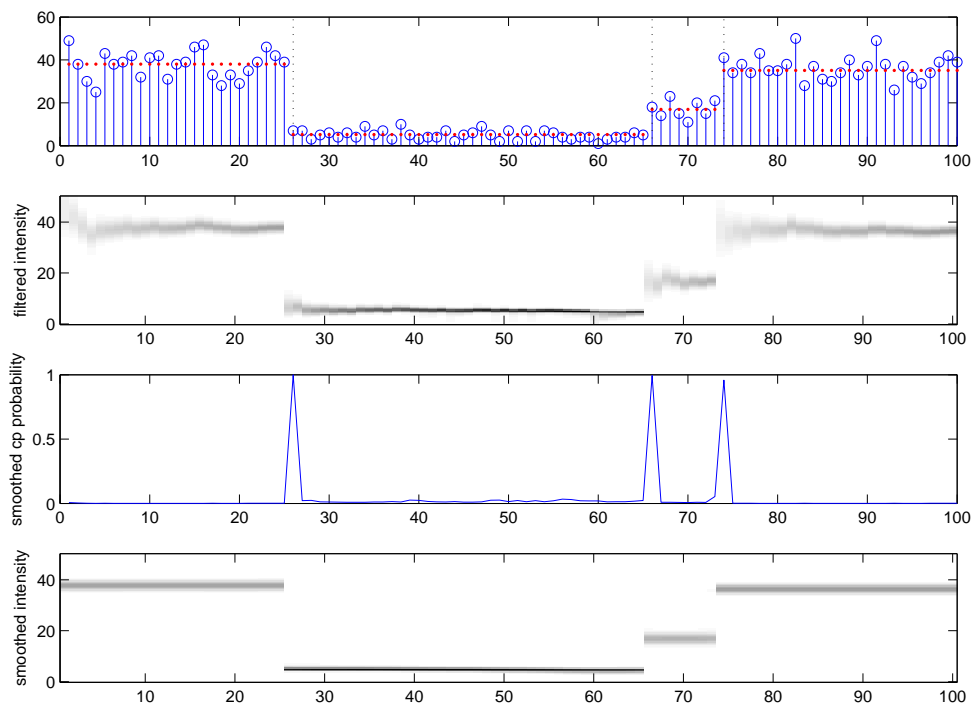


Figure 2: Top to bottom. Synthetic data generated from the multiple changepoint model. Filtered density $p(\lambda_t|x_{1:t})$, Smoothed $p(r_t = 1|x_{1:M})$, smoothed $p(\lambda_t|x_{1:M})$.

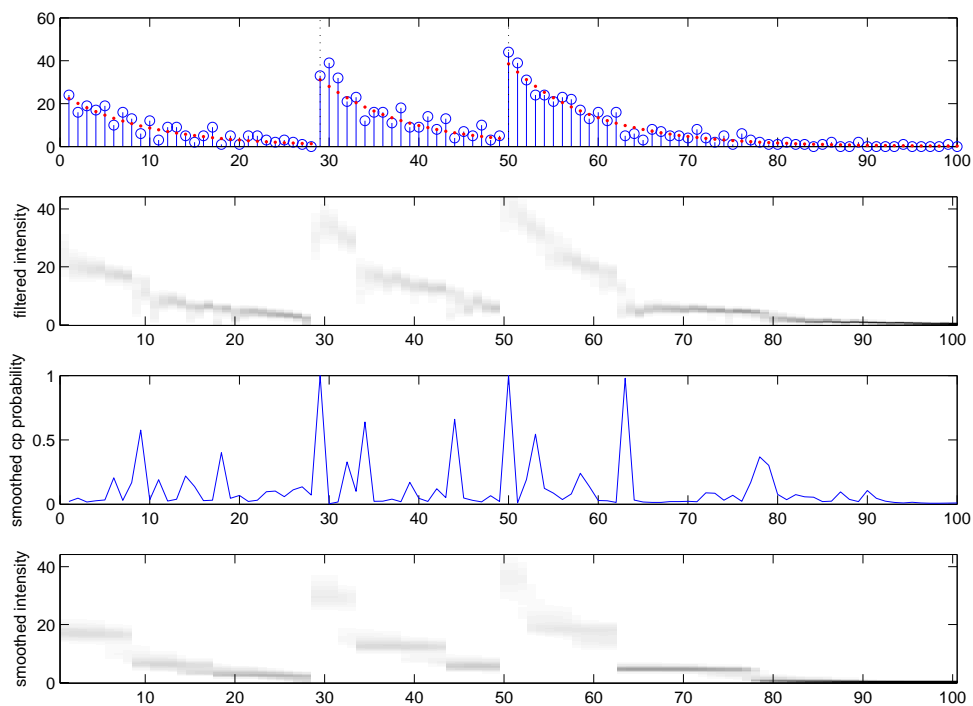


Figure 3: Top to bottom. Synthetic data generated from a different multiple changepoint model. Filtered density $p(\lambda_t|x_{1:t})$, Smoothed $p(r_t = 1|x_{1:M})$, smoothed $p(\lambda_t|x_{1:M})$. Note that the algorithm finds a reasonable step approximation to the changing intensities.

A Product of two Gamma densities

We verify that when

$$\begin{aligned} p_1(x) &\sim \mathcal{G}(x; a_1, b_1) \\ p_2(x) &\sim \mathcal{G}(x; a_2, b_2) \end{aligned}$$

the product is also gamma potential:

$$\begin{aligned} \log p_1(x)p_2(x) &= +(a_1 - 1) \log x - b_1 x - \log \Gamma(a_1) + a_1 \log b_1 \\ &\quad +(a_2 - 1) \log x - b_2 x - \log \Gamma(a_2) + a_2 \log b_2 \\ &= (a_1 + a_2 - 1 - 1) \log x - (b_1 + b_2)x - \log \Gamma(a_1) - \log \Gamma(a_2) + a_1 \log b_1 + a_2 \log b_2 \\ &= \log \mathcal{G}(x; a_1 + a_1 - 1, b_1 + b_2) \\ &\quad + \log \Gamma(a_1 + a_2 - 1) - (a_1 + a_2 - 1) \log(b_1 + b_2) - \log \Gamma(a_1) - \log \Gamma(a_2) + a_1 \log b_1 + a_2 \log b_2 \\ &= \log \mathcal{G}(x; a_1 + a_2 - 1, b_1 + b_2) \\ &\quad + \log \Gamma(a_1 + a_2 - 1) - \log \Gamma(a_1) - \log \Gamma(a_2) + \log(b_1 + b_2) + a_1 \log(b_1/(b_1 + b_2)) + a_2 \log(b_2/(b_1 + b_2)) \end{aligned}$$

Matching the coefficients, we obtain the formula in ??.

B Implementation

B.1 Data generation

```
% CMPE58N_MCP_POISS Script to generate data from the model
%

% Change History :
% Date Time Prog Note
% 01-Apr-2009 9:40 AM ATC Created under MATLAB 7.7.0

% ATC = Ali Taylan Cemgil,
% Department of Computer Engineering, Bogazici University
% e-mail : taylan.cemgil@boun.edu.tr

% Set the Hyperparameters
M = 100;
data.M = M;
data.nu = 0.9; data.B = 0.1;
data.a0 = 5; data.b0 = 0.2;
data.log_p1 = log(0.05); data.log_p0 = log(1 - exp(data.log_p1));

data.r = double(rand(1, M) < exp(data.log_p1));
data.lambda0 = gamrnd(data.a0, 1/data.b0);
data.lambda = zeros(1, M);
for t=1:M,
    if data.r(t)==1,
        data.lambda(t) = gamrnd(data.nu, 1/data.B);
    else
        if t>1,
            data.lambda(t) = data.lambda(t-1);
        else
            data.lambda(t) = data.lambda0;
        end;
    end;
end;
data.x = poissrnd(data.lambda);

% Plot data
stem(data.x)
```

```

hold on
plot(0:M, [data.lambda0 data.lambda], 'r.')
ln= grid_line(find(data.r)); set(ln, 'lines', ':')
hold off

```

```

legend('x', '\lambda'); xlabel('t'); ylabel('x_t')

```

B.2 Inference

```

% CMPE58N_MCP_POISS_INFERENCE Exact inference for the Poisson changepoint model
%

```

```

% Change History :
% Date Time Prog Note
% 26-Nov-2009 11:32 PM ATC Created under MATLAB 7.7.0

```

```

% ATC = Ali Taylan Cemgil,
% Department of Computer Engineering, Bogazici University
% e-mail : taylan.cemgil@boun.edu.tr

```

```

M = data.M;
% Potential
% 1st index : mixture component
% 2nd       : time slice
% 3rd       : params of the gamma potential
eng.ff = zeros(M+1, M, 3);
eng.fp = zeros(M+1, M, 3);

```

```

% Backward potentials
eng.bf = zeros(M+1, M, 3);

```

```

for t=1:M
    if t==1,
        tau = 0;
        eng.fp(tau +1, t, :) = reshape([data.nu, data.B, data.log_p1], [1 1 3]);
        tau = 1;
        eng.fp(tau +1, t, :) = reshape([data.a0, data.b0, data.log_p0], [1 1 3]);
    else
        c = log_sum_exp(eng.ff((0:t-1) +1, t-1, 3), 1);
        tau = 0;
        eng.fp(0 +1, t, :) = reshape([data.nu, data.B, data.log_p1 + c], [1 1 3]);
        tau = 1:t;
        eng.fp(tau +1, t, :) = eng.ff(tau-1 +1, t-1, :);
        eng.fp(tau +1, t, 3) = eng.fp(tau +1, t, 3) + data.log_p0;
    end;

    % Filter
    eng.ff((0:t) +1, t, :) = cmpe58n_mcp_update( eng.fp((0:t) +1, t, :), data.x(t));
end;

```

```

% Backward pass
for t=M:-1:1,
    if t==M,
        tau = 0;
        eng.bf(tau +1, t, :) = reshape([data.x(t)+1, 1, 0], [1 1 3]);
    else
        tau = 0;
        temp = cmpe58n_mult_gampot(eng.bf((0:(M-(t+1))) +1, t+1, :), reshape([data.nu, data.B, 0], [1 1 3]));
        c = log_sum_exp(temp(:,1,3));
        eng.bf(tau +1, t, :) = reshape([data.x(t)+1, 1, data.log_p1 + c], [1 1 3]);
    end;
end;

```

```

    tau = 1:(M-t);
    eng.bf(tau +1, t, :) = cmpe58n_mult_gampot(eng.bf(tau-1 +1, t+1, :), reshape([data.x(t)+1, 1, data.log_p0],
end
end

```

B.3 Helper routines

```

function [lg] = cmpe58n_eval_mogamma(g, lambda)
% CMPE58N_EVAL_MOGAMMA Evaluates a mixture of gamma densities
%
% Inputs :
% g : N x 1 x 3 array of gamma potential parameters
%      lambda : grid to evaluate the posterior
%
% Outputs :
% lg : log density
%
% Change History :
% Date Time Prog Note
% 24-Nov-2009 4:33 PM ATC Created under MATLAB 7.7.0

% ATC = Ali Taylan Cemgil,
% Department of Computer Engineering, Bogazici University
% e-mail : taylan.cemgil@boun.edu.tr

J = size(g, 1);
lg = zeros(size(lambda));
Z = log_sum_exp(g(:, 1, 3),1);
for j=1:J,
    a = g(j, 1, 1);
    b = g(j, 1, 2);

    lg = lg + exp(g(j,1,3) - Z + (a - 1).*log(lambda) - b.*lambda - gammaln(a) + a.*log(b));
end;

function [gp] = cmpe58n_mult_gampot(g1, g2)
% CMPE58N_MULT_GAMPOT Multiplies gamma potentials
%
% Inputs :
% g1, g2 gamma potentials :
%
% Outputs :
% gp : Coefficients of exp(c_1)Gamma(x; a_1, b_1)exp(c_2)Gamma(x; a_2, b_2)
%
% Change History :
% Date Time Prog Note
% 24-Nov-2009 2:19 PM ATC Created under MATLAB 7.7.0

% ATC = Ali Taylan Cemgil,
% Department of Computer Engineering, Bogazici University
% e-mail : taylan.cemgil@boun.edu.tr

M1 = size(g1,1);
M2 = size(g2,1);
gp = zeros(M1, M2, 3);

%a = a_1 + a_2 - 1;
gp(:, :, 1) = repmat(g1(:,1,1), [1 M2]) + repmat(g2(:,1,1)', [M1 1]) - 1;
%b = b_1 + b_2;
gp(:, :, 2) = repmat(g1(:,1,2), [1 M2]) + repmat(g2(:,1,2)', [M1 1]);
%c = c_1 + c_2 + gammaln(a_1 + a_2 - 1) - (a_1 + a_2 - 1).*log(b_1 + b_2) - gammaln(a_1) - gammaln(a_2) + a_1.*log(b_1) + a_2.*log(b_2);

```

```
gp(:,:,3) = repmat(g1(:,1,3), [1 M2]) + repmat(g2(:,1,3)', [M1 1]) + gammaln(gp(:,:,1)) - gp(:,:,1).*log(gp(:,:,1));
```

```
function [gu] = cmpe58n_mcp_update(g, x)
% CMPE58N_MCP_UPDATE Updates several gamma potentials with a single observation
%
% [gu] = cmpe58n_mcp_update(g, x)
%
% Inputs :
% g : A collection of gamma potentials
% x : Observation
%
% Outputs :
% gu : Updated potentials
%
% Usage Example : [] = cmpe58n_mcp_update();
%
% Note :
% See also
%
% Uses :
%
% Change History :
% Date Time Prog Note
% 24-Nov-2009 2:46 PM ATC Created under MATLAB 7.7.0
%
% ATC = Ali Taylan Cemgil,
% Department of Computer Engineering, Bogazici University
% e-mail : taylan.cemgil@boun.edu.tr
%
M = size(g, 1);
gu = zeros(size(g));
%
a = g(:, :, 1);
b = g(:, :, 2);
c = g(:, :, 3) + gammaln(a + x) - gammaln(a) - gammaln(x+1) + a.*log(b) - (a + x).*log(b+1) ;
gu = cat(3, a + x, b + 1, c);
```

B.4 Visualisation

```
% CMPE58N_MCP_POISS_VISUALISE Compute smoothed estimates and plot
```

```
% Change History :
% Date Time Prog Note
% 26-Nov-2009 11:33 PM ATC Created under MATLAB 7.7.0
%
% ATC = Ali Taylan Cemgil,
% Department of Computer Engineering, Bogazici University
% e-mail : taylan.cemgil@boun.edu.tr
%
lam = linspace(0.01,max(data.x),100);
%
eng.pr = zeros(2, M);
eng.prs = zeros(2, M);
%
LAM = zeros(length(lam), M);
LAMS = zeros(length(lam), M);
%
Evaluate filtered estimates of indicators
for j=1:M
    eng.pr(1, j) = eng.ff(0 +1, j, 3);
    eng.pr(2, j) = log_sum_exp(eng.ff((1:j) +1, j, 3), 1);
```

```

    LAM(:, j) = cmpe58n_eval_mogamma(eng.ff((0:j) +1, j, :), lam);
end;

Evaluate smoothed densities
for j=M:-1:1,
    gamma = cmpe58n_mult_gampot(eng.fp((0:j) +1, j, :), eng.bf( (0:(M-j)) +1, j, : ) );
    eng.prs(1,j) = log_sum_exp(gamma(1, :, 3),2);
    c = gamma(2:end, :, 3);
    eng.prs(2,j) = log_sum_exp(c(:), 1);
    LAMS(:, j) = cmpe58n_eval_mogamma(reshape(gamma, [size(gamma,1)*size(gamma,2) 1 3]), lam);
end

subplot(411)
stem(data.x)
hold on
plot((data.lambda), 'r.')
ln= grid_line(find(data.r)); set(ln, 'lines', ':')
hold off

subplot(412)
imagesc(1:M, lam, LAM); set(gca, 'ydir', 'n')
ylabel('filtered intensity')

subplot(413);
pr = normalize_exp(eng.pr, 1); plot(pr(1,:))
prs = normalize_exp(eng.prs, 1); plot(prs(1,:))
ylabel('smoothed cp probability')

subplot(414)
imagesc( 1:M, (lam), LAMS); set(gca, 'ydir', 'n')
ylabel('smoothed intensity')

colormap(flipud(gray))

```