

BNPlib: A Nonparametric C++ Library

(part 3)

Bruno Guindani
Elena Zazzetti

February 19th, 2020



POLITECNICO
MILANO 1863

<https://github.com/poliprojects/BNPlib>

Model

DP and DPM models

Having observed the iid sample $\{y_i\}_i$, $i = 1, \dots, n$:

- Dirichlet process model (discrete):

$$y_i | G \stackrel{\text{iid}}{\sim} G$$
$$G \sim DP(MG_0)$$

- Dirichlet process mixture (**DPM**) model (continuous):

$$y_i | G \stackrel{\text{iid}}{\sim} f_G(\cdot) = \int_{\Theta} f(\cdot | \boldsymbol{\vartheta}) G(d\boldsymbol{\vartheta})$$
$$G \sim DP(MG_0)$$

Equivalent formulations (1)

- (DPM) is equivalent to:

$$y_i | \boldsymbol{\vartheta}_i \stackrel{\text{iid}}{\sim} f(\cdot | \boldsymbol{\vartheta}_i), \quad i = 1, \dots, n$$

$$\boldsymbol{\vartheta}_i | G \stackrel{\text{iid}}{\sim} G, \quad i = 1, \dots, n$$

$$G \sim DP(MG_0)$$

- State $\forall i$: $\boldsymbol{\vartheta}_i$ latent variables (discrete)

Equivalent formulations (2)

- (DPM) is also equivalent to:

$$y_i | c_i, \phi_1, \dots, \phi_k \stackrel{\text{iid}}{\sim} f(\cdot | \phi_{c_i}), \quad i = 1, \dots, n$$

$$c_i | \mathbf{p} \stackrel{\text{iid}}{\sim} \sum_{j=1}^K p_j \delta_j(\cdot), \quad i = 1, \dots, n$$

$$\phi_c \stackrel{\text{iid}}{\sim} G_0, \quad c = 1, \dots, k$$

$$\mathbf{p} \sim \text{Dir}(M/K, \dots, M/K)$$

$$K \rightarrow +\infty$$

- State $\forall i$: c_i **allocations** to clusters
- State $\forall i$: ϕ_{c_i} **unique values** for each cluster
- Only the finitely many ϕ_c used are kept track of

Case study

- (DPM) with a Normal Normal-InverseGamma (**NNIG**) hierarchy:

$$y_i | \boldsymbol{\vartheta}_i \stackrel{\text{ind}}{\sim} f(\cdot | \boldsymbol{\vartheta}_i), \quad i = 1, \dots, n$$

$$\boldsymbol{\vartheta}_i | G \stackrel{\text{iid}}{\sim} G, \quad i = 1, \dots, n$$

$$G \sim DP(MG_0)$$

$$f(y | \boldsymbol{\vartheta}) = N(y | \mu, \sigma^2)$$

$$G_0(\boldsymbol{\vartheta} | \mu_0, \lambda_0, \alpha_0, \beta_0) = N\left(\mu | \mu_0, \frac{\sigma^2}{\lambda_0}\right) \times \text{Inv-Gamma}(\sigma^2 | \alpha_0, \beta_0)$$

- Latent variables: $\boldsymbol{\vartheta} = (\mu, \sigma)$
- State $\forall i$: c_i, ϕ_{c_i}

Algorithms

General structure

```
template<template <class> class Hierarchy,  
        class Hypers, class Mixture> class Algorithm  
  
    void step(){  
        sample_allocations();  
        sample_unique_values();  
    }  
  
    void run(){  
        initialize();  
        unsigned int iter = 0;  
        while(iter < maxiter){  
            step();  
            if(iter >= burnin){  
                save_iteration(iter);  
            }  
            iter++;  
        }  
    }
```


Auxiliary classes

- Specific common interface
- Mixture \rightarrow SimpleMixture
- Hypers \rightarrow HypersFixedNNIG
- Hierarchy<Hypers> \rightarrow HierarchyNNIG<Hypers>

- Has a vector of m `aux_unique_values`
- `initialize()`
- `sample_allocations()`: for all observations $i = 1, \dots, n$
 - ▶ compute $\text{card}[c] = n_{-i,c}$ for all clusters $c = 1, \dots, k$
 - ▶ if c_i is a singleton, move ϕ_{c_i} to `aux_unique_values[0]`
 - ▶ draw all (other) `aux_unique_values` iid from G_0
 - ▶ draw a new value c for c_i according to:

$$\mathbb{P}(c_i = c | \mathbf{c}_{-i}, y_i, \phi_1, \dots, \phi_h) \propto \begin{cases} \frac{n_{-i,c}}{n-1+M} f(y_i | \phi_c), & \text{for } 1 \leq c \leq k^- \\ \frac{M/m}{n-1+M} f(y_i | \phi_c), & \text{for } k^- + 1 < c \leq h \end{cases}$$

with k^- unique values excluding c_i and $h = k^- + m$

- ▶ update `card` and `allocations` (4 cases)
- `sample_unique_values()`: for all clusters $c = 1, \dots, k$
 - ▶ build `curr_data` that contains all observations in cluster c
 - ▶ draw ϕ_c from its posterior distribution given `curr_data`

- For conjugate models only, e.g. (DPM)+(NNIG)
- `initialize()`
- `sample_allocations()`: for all observations $i = 1, \dots, n$
 - ▶ compute $\text{card}[c] = n_{-i,c}$ for all clusters $c = 1, \dots, k$
 - ▶ draw a new value c for c_i according to:

$$\text{If } c = c_j \text{ for some } j: \mathbb{P}(c_i = c | \mathbf{c}_{-i}, y_i, \phi) \propto \frac{n_{-i,c}}{n-1+M} f(y_i | \phi_c)$$

$$\mathbb{P}(c_i \neq c_j \text{ for all } j | \mathbf{c}_{-i}, y_i, \phi) \propto \frac{M}{n-1+M} \int_{\Theta} f(y_i | \boldsymbol{\vartheta}) G_0(d\boldsymbol{\vartheta})$$

- ▶ if the latter, draw a new ϕ_c from its posterior given y_i
 - ▶ update card and allocations (4 cases)
- `sample_unique_values()`: for all clusters $c = 1, \dots, k$
 - ▶ build `curr_data` that contains all observations in cluster c
 - ▶ draw ϕ_c from its posterior distribution given `curr_data`

Applications

Cluster estimation

```
unsigned int cluster_estimate();
```

$$\hat{k} = \arg \min_k \left\| D^{(k)} - \bar{D} \right\|_F^2 = \arg \min_k \sum_{i,j} \left(D_{ij}^{(k)} - \bar{D}_{ij} \right)^2$$

- $D^{(k)}$: dissimilarity matrix at iteration k
- $\bar{D} = \frac{1}{K} \sum_k D^{(k)}$: mean over K iterations

Density estimation

```
void eval_density(const std::vector<double> grid);
```

$$\hat{f}^{(k)}(x) = \sum_j \frac{n_j^{(k)}}{M+n} f(x|\phi_j^{(k)}) + \frac{M}{M+n} m(x)$$

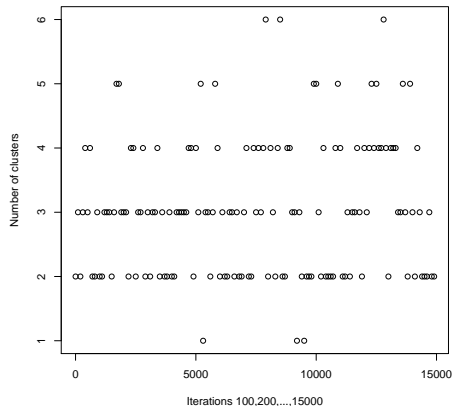
$$\hat{m}(x) = \frac{1}{m} \sum_{h=0}^{m-1} f(x|\phi_h)$$

$$\implies \hat{f}(x) = \frac{1}{K} \sum_k \hat{f}^{(k)}(x)$$

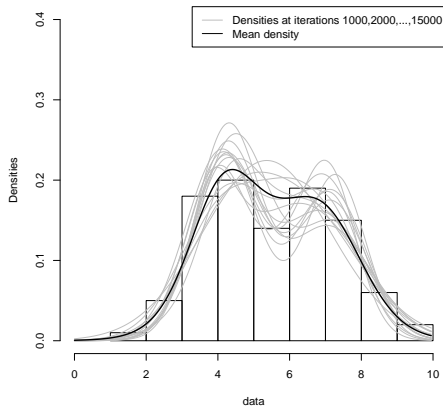
Results

Oscillations

Number of clusters at single iterations

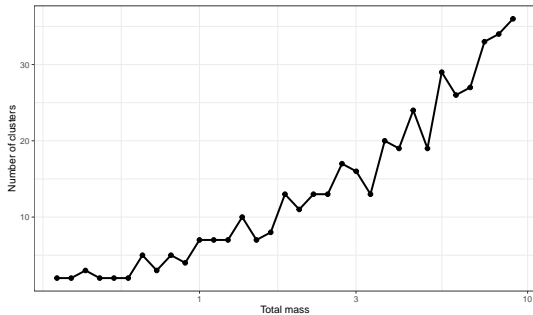


Local density estimates at single iterations

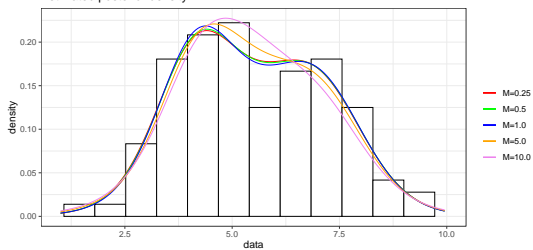


Total mass

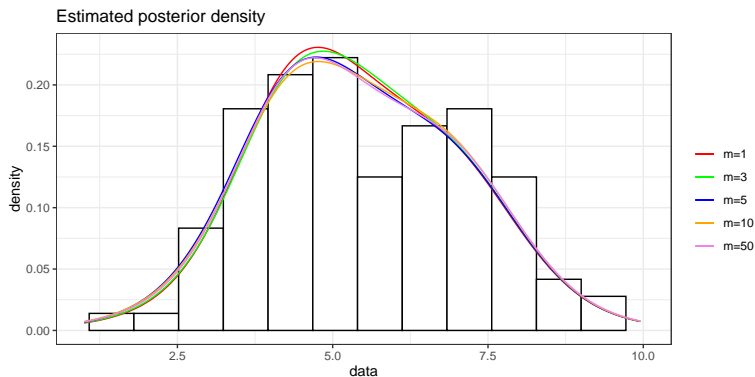
Number of clusters as a function of the total mass



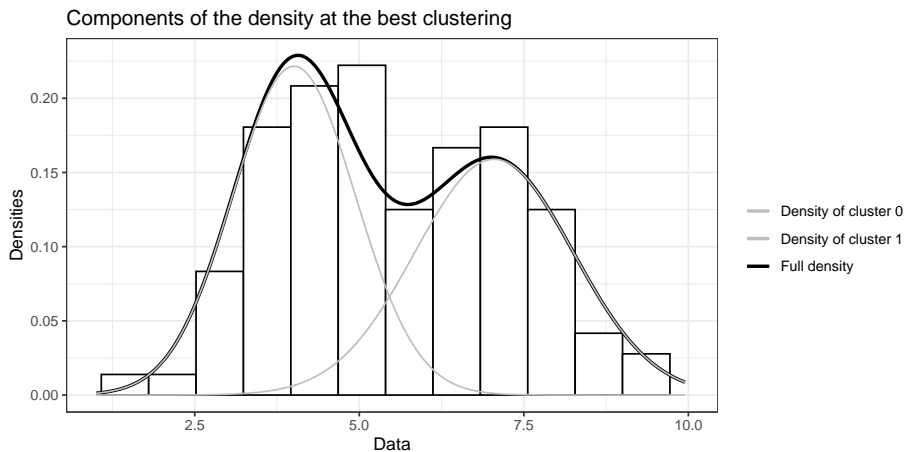
Estimated posterior density



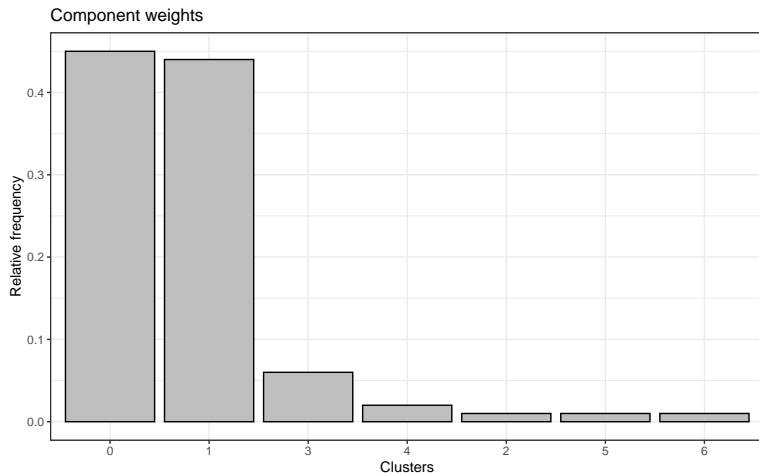
Auxiliary parameters



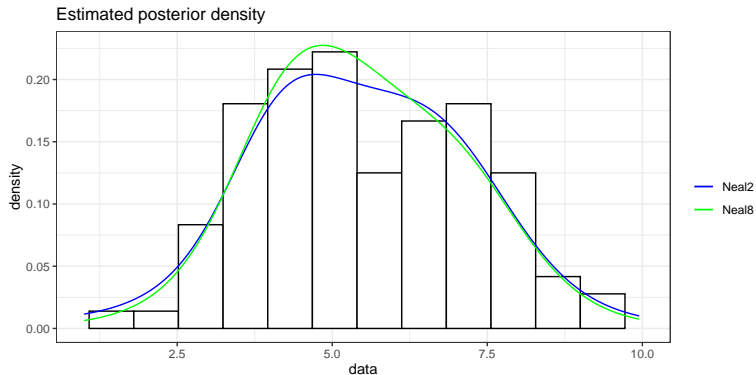
Density components












Clustering



Neal2 vs Neal8



Bibliography

-  Muller, Quintana, *Bayesian Nonparametric Data Analysis*
-  Neal (2000), *Markov Chain Sampling Methods for Dirichlet Process Mixture Models*
-  Ishwaran, James (2001), *Gibbs Sampling Methods for Stick-Breaking Priors*
-  Murphy (2007), *Conjugate Bayesian analysis of the Gaussian distribution*
-  Protocol Buffers: <https://developers.google.com/protocol-buffers/docs/cpptutorial>
-  Stan: <http://mc-stan.org/math>
-  Eigen: <https://eigen.tuxfamily.org/dox>
-  GitHub codes of Mario Beraha and Riccardo Corradin for similar projects
-  Course material for Bayesian Statistics: <https://beep.metid.polimi.it/web/2019-20-bayesian-statistics-alessandra-guglielmi-/>