`BNPlib` for density estimation:
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 $\{y_i\}_i$, $i = 1, \ldots, n$:

- Dirichlet process model (discrete):

$$\begin{aligned} y_i | G &\overset{\text{iid}}{\sim} G \\ G &\sim DP(MG_0) \end{aligned}$$

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

$$\begin{aligned} y_i | G &\overset{\text{iid}}{\sim} f_G(\cdot) = \int_\Theta f(\cdot|\boldsymbol{\vartheta})\, G(\mathrm{d}\boldsymbol{\vartheta}) \\ G &\sim DP(MG_0) \end{aligned}$$

Equivalent formulations (1)

- (DPM) is equivalent to:

$$
\begin{aligned}
y_i|\vartheta_i &\overset{\text{ind}}{\sim} f(\cdot|\vartheta_i), \quad i = 1, \dots, n \\
\vartheta_i|G &\overset{\text{iid}}{\sim} G, \quad i = 1, \dots, n \\
G &\sim DP(MG_0)
\end{aligned}
$$

- State $\forall i$: ϑ_i latent variables (discrete)

Equivalent formulations (2)

- (DPM) is also equivalent to:

$$
\begin{aligned}
y_i | \boldsymbol{\phi}, c_i &\overset{\text{ind}}{\sim} f(\cdot|\phi_{c_i}), \quad i = 1, \ldots, n \\
c_i | \mathbf{p} &\overset{\text{iid}}{\sim} \sum_{j=1}^{K} p_j \delta_j(\cdot), \quad i = 1, \ldots, n \\
\phi_c &\overset{\text{iid}}{\sim} G_0, \quad c = 1, \ldots, k \\
\mathbf{p} &\sim \mathrm{Dir}(M/K, \ldots, M/K) \\
K &\to +\infty
\end{aligned}
$$

- 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 Normal Normal-InverseGamma (**NNIG**) prior:

$$\begin{aligned} y_i|\vartheta_i &\overset{\text{ind}}{\sim} f(\cdot|\vartheta_i), \quad i = 1, \dots, n \\ \vartheta_i|G &\overset{\text{iid}}{\sim} G, \quad i = 1, \dots, n \\ G &\sim DP(MG_0) \end{aligned}$$

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

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

- Latent variables: $\boldsymbol{\vartheta} = (\mu, \sigma)$
- State: $\boldsymbol{\phi}, \boldsymbol{c}$

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

- Need a specific common interface
- `Mixture` $\rightarrow$ `SimpleMixture`
- `Hypers` $\rightarrow$ `HypersFixedNNIG`
- `Hierarchy<Hypers>` $\rightarrow$ `HierarchyNNIG<Hypers>`

Neal8

- Has a vector of m `aux_unique_values`
- `initialize()`
- `sample_allocations()`: for all observations $i = 1, \dots, n$
  - compute `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:

$$\mathbb{P}(c_i = c | \boldsymbol{c}_{-i}, y_i, \boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_h) \propto \begin{cases} \dfrac{n_{-i,c}}{n-1+M} f(y_i | \boldsymbol{\phi}_c), & \text{for } 1 \leq c \leq k^- \\ \dfrac{M/m}{n-1+M} f(y_i | \boldsymbol{\phi}_c), & \text{for } k^- + 1 < c \leq h \end{cases}$$

    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

Neal2

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

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

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

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

Applications

Cluster estimation

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

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

Density estimation

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

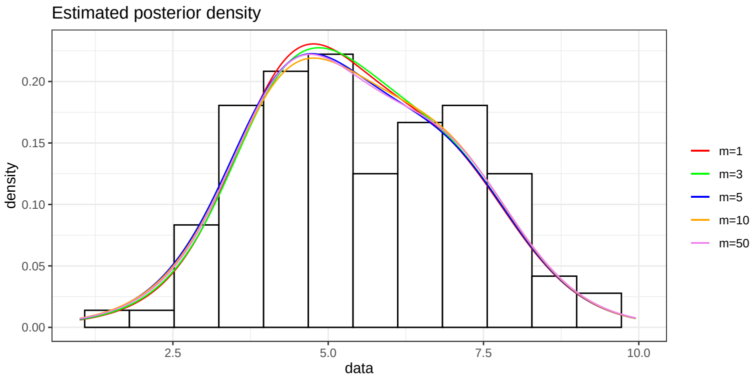
$$\hat{f}^{(k)}(x) \;=\; \sum_j \frac{n_j^{(k)}}{M+n} f\left(x|\phi_j^{(k)}\right) \;+\; \frac{M}{M+n} m(x)$$

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

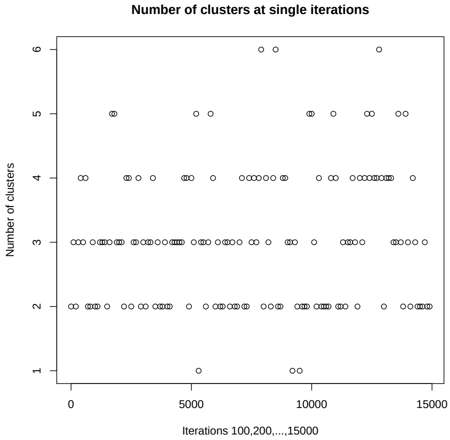
$$\Longrightarrow \hat{f}(x) = \frac{1}{K} \sum_k \hat{f}^{(k)}(x)$$

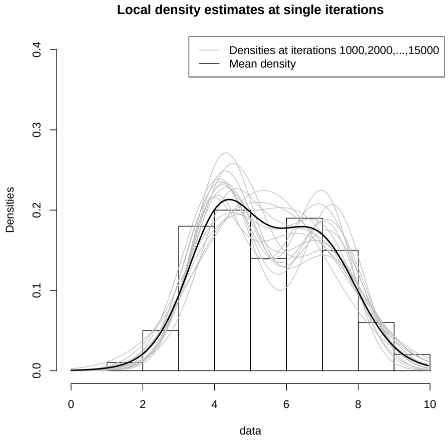
Results

Auxiliary parameters

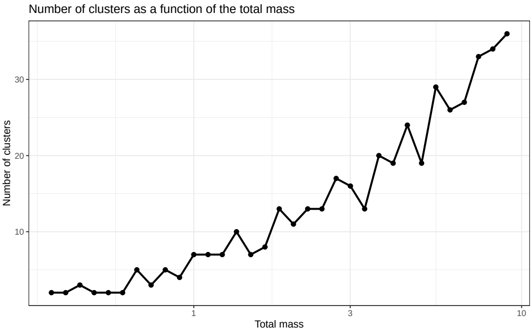


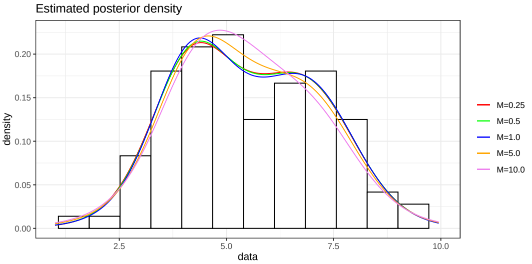
Oscillations



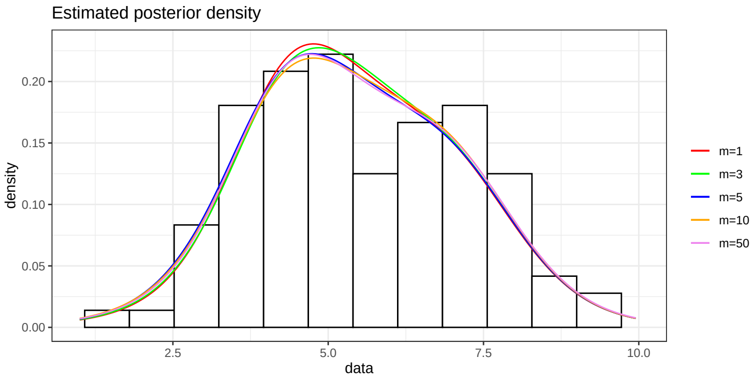


Total mass

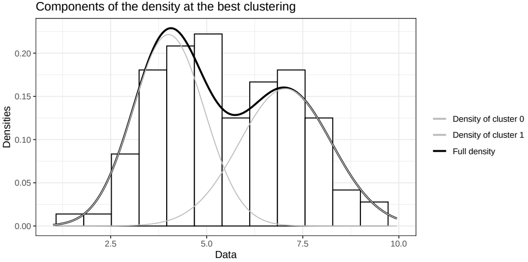

Number of clusters as a function of the total mass
Number of clusters
30
20
10
1
3
10
Total mass

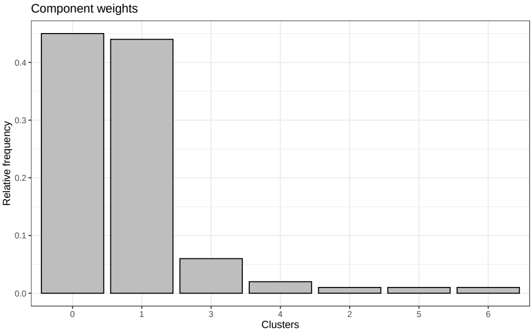

Estimated posterior density
density
0.20
0.15
0.10
0.05
0.00
2.5
5.0
7.5
10.0
data
M=0.25
M=0.5
M=1.0
M=5.0
M=10.0

Auxiliary parameters

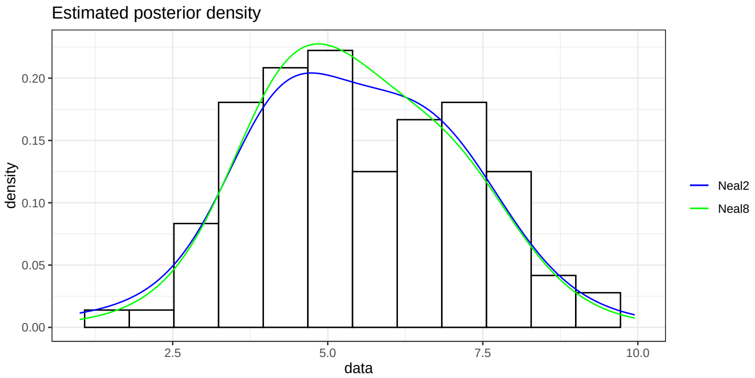


Density components





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-/`