BNPlib for density estimation:

A nonparametric C++ library (part 2)

Bruno Guindani
Elena Zazzetti

January 8th, 2020

POLITECNICO
MILANO 1863

https://github.com/poliprojects/BNPlib

Model

$$
\begin{aligned}
(Y_i|\vartheta_i) &\sim F(\cdot, \vartheta_i) \\
(\vartheta_i|G) &\sim G \\
G &\sim DP(M, G_0)
\end{aligned}
\quad \overset{K\to\infty}{\Longleftrightarrow} \quad
\begin{aligned}
(Y_i|\phi, c_i) &\sim F(\cdot, \phi_{c_i}) \\
(c_i|\mathbf{p}) &\sim \sum_{k=1}^{K} p_k \delta_k(\cdot) \\
\phi_c &\sim G_0 \\
\mathbf{p} &\sim \mathrm{Dir}(M/K, \ldots, M/K)
\end{aligned}
$$

(hierarchical model)

(K-discrete model)

with $\boldsymbol{\vartheta} \leftrightsquigarrow (\boldsymbol{\phi}, \mathbf{c})$

Algorithms

- Neal2, Neal8, blocked Gibbs
- **Gibbs sampling** procedures
- General structure:
  - sample **allocations** c from some conditional distribution
  - sample **unique values** ϕ from some conditional distribution
  - (sample **weighGibbs samplingts** p of the unique values deltas)

Implementation

Libraries:

- Stan
- Eigen
- Armadillo

Structure of Algorithm

Structure: Algorithm<Hierarchy, Mixture>

Algorithm<Hierarchy, Mixture>
Mixture mixture vector<data_type> data vector<unsigned int> allocations vector<Hierarchy> unique_values

Hierarchy<Hypers>
state_type state shared_ptr <Hypers> hypers void draw() void sample_given_data()

Specializations

Hierarchy : NNIGHierarchy

- draw()
  - stan::math::inv_gamma_rng
  - stan::math::normal_rng
- sample_given_data()
  - NormalGammaUpdate
  - stan::math::inv_gamma_rng
  - stan::math::normal_rng

Mixture : SimpleMixture $\rightarrow$ TotalMass

Hypers : HypersFixed $\rightarrow \mu_0 \ \lambda_0 \ \alpha_0 \ \beta_0$

Neal8

Algorithm Base → Neal8 Derived

Neal8<Hierarchy, Mixture>
Mixture mixture vector<data_type> data vector<unsigned int> allocations vector<Hierarchy> unique_values vector<Hierarchy> aux_unique_values

The algorithms in C++

```
299
300     void run(){
301         initalize();
302         unsigned int iter = 0;
303         while(iter < maxiter){
304             step();
305             if(iter >= burnin)
306                 save_iteration(iter);
307         }
308     }
```

- Example: Normal-NormalInvGamma hierarchy, no hyperpriors
- `initalize()`: random allocation
- `step()`
  - `sample_allocations()`: vector `card` of cardinalities of clusters
  - 4 cases handled separately: singleton vs !singleton, aux vs old
  - `sample_unique_values()`: vector `clust_idxs` to record which data are in each cluster
- Actual cluster structures?

Impending extensions

- **Hyperpriors**: objects of class `Hypers` store pointers to objects of class `HypersFixed`
- **Non-conjugacy**: via Stan's HMC sampler
- **R interface**: via **protocol buffers**

Protocol Buffers

- API developed by Google
- Data is saved in XML-like structures, called **messages**, that are defined in `.proto` files
- Each message corrresponds to a class in C++
- The `protoc` compiler produces the C++ files that make up the API
- RProtoBuf
- Compromise between efficiency and human-readibility

A general library?

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*
- https://developers.google.com/protocol-buffers/docs/cpptutorial