Programs for Bayesian inference of Gaussian mixture models with noninformative priors

Colin J. Stoneking*

May 26, 2014

This is the documentation for a set of implementations of the Bayesian model described in (Stoneking, 2014). In brief, this is a modified Gaussian mixture model (GMM) which enables noninformative Jeffreys priors to be placed on the parameters of the mixture components. This avoids certain difficulties which otherwise occur in prior specification for a GMM. At the moment, the implementations are only for data points in one dimension (i.e. mixtures of univariate Gaussians). They are provided as MATLAB and R functions. These functions use Markov Chain Monte Carlo to generate approximate samples from the posterior distribution of the component means, the component standard deviations, and the proportions with which these components contribute to the observed data, conditional on the data vector $\mathbf{x}$.

Both collapsed Gibbs sampling (CG) and Metropolis-Hastings (MH)-based implementations are provided here. Generally, the MH-based function (bayesian_GMM_MH) will be more suitable, as the sampler has better convergence properties and is more computationally efficient. However, it requires proposal distributions to be specified, and possibly to be fine-tuned to some extent. By contrast, the CG-based function (bayesian_GMM_CG) does not require these specifications, so it may be more convenient to use in some cases. The functions return the following:

*Seminar for Statistics, ETH Zürich. Email: cjstoneking@gmail.com

mean_samples	S × K matrix, where each column is a sequence of samples from the posterior distribution of the mean of a single component. S is the number of samples, and is given by post_burn_in / thin (see arguments) K is the number of components to use in the model (see arguments)
sd_samples	S × K matrix, where each column is a sequence of samples from the posterior distribution of the standard deviation of a single component
prop_samples	S × K matrix, where each column is a sequence of samples from the posterior distribution of the proportion of data points in the sample generated by this component
acceptances	Only for MH algorithm: the number of times the proposal is accepted

Note that usually, a necessary requirement for convergence of the Markov chain to have occurred is the presence of multiple symmetric modes in the sample density of the component means, due to label-switching (Celeux *et al.*, 2000). However, observing symmetric modes does not guarantee that convergence has occurred.

Both the CG and MH algorithms have the following arguments (which must be provided unless otherwise stated):

x	Vector of data points (first argument, unnamed)
K	Number of Gaussian components to use
delta	Determines the parameter vector $\boldsymbol{\delta}$ of the prior $\pi^\star(\mathbf{G})$: $\boldsymbol{\delta}$ is a vector of length K whose entries are all equal to delta (optional, default is 1)
burn_in	Number of Markov chain iterations to run for a burn-in period
post_burn_in	Number of Markov chain iterations to run after burn-in, while storing samples
thin	Interval at which to store samples from the Markov chain
progress_bar	Whether to draw a progress bar (boolean, optional, default is *true*)

Some additional arguments must be provided to the MH algorithm:

start_mean	Vector of starting values for the component means
start_sd	Vector of starting values for the component standard deviations
min_mean	Minimum value the component means are allowed to take
max_mean	Maximum value the component means are allowed to take
min_sd	Minimum value the component standard deviations (SDs) are allowed to take
scale_mean	SD of the Gaussian proposal distribution for each component mean
scale_sd	SD of the Gaussian proposal distribution for each component SD

All arguments except x (the first argument) are given via name-value pairs, so their order does not matter. Both R and MATLAB implementations are provided. The MATLAB functions are in the files **bayesian_GMM_CG.m**, **bayesian_GMM_MH.m**, **regular_partition.m** and **randdirichlet.m**. All R functions are defined in the file **Bayesian_GMM.R**.

MATLAB example:

```
[mean_samples, sd_samples, prop_samples] = bayesian_GMM_CG(x, 'K', 2, 'thin', 100,...
                                           'burn_in', 10^5, 'post_burn_in', 10^6);
```

or, using GMM_MH:

```
[mean_samples, sd_samples, prop_samples, acceptances] =...
               bayesian_GMM_MH(x, 'K', 2, 'min_mean', -5,...
               'max_mean', 5, 'min_sd', 0.01,...
               'start_mean', [-1, 1], 'scale_mean', 1, 'start_sd', [-1 1],...
               'scale_sd', 0.1, 'thin', 100, 'burn_in', 10^5, 'post_burn_in', 10^5);
```

R example:

```
results <- bayesian_GMM_CG(x, K=2, thin=100, burn_in=10^5, post_burn_in=10^6)
mean_samples <- results$mean_samples
```

or, using GMM_MH:

```
results <- bayesian_GMM_MH(x, K=2, min_mean=-5, max_mean=5, min_sd=0.01,
   start_mean=c(-1,1), scale_mean=1, start_sd=c(1,1), scale_sd=0.1,
   thin=100, burn_in=10^5, post_burn_in=10^5)
mean_samples <- results$mean_samples
```

References

Celeux, G., Hurn, M., & Robert, C. P. 2000. Computational and inferential difficulties with mixture posterior distributions. *Journal of the American Statistical Association*, **95**(451), 957–970.

Stoneking, C. J. 2014. Bayesian inference of Gaussian mixture models with noninformative priors. *arXiv preprint arXiv:1405.4895*.