DPP: Reference documentation

version 0.1.2

Luis M. Avila, Mike R. May and Jeffrey Ross-Ibarra

17th November 2017

Contents

1 DPP: Introduction

DPP infers a potentially infinite number of normal distributions from a numeric vector. It uses an MCMC (Markov Chains Monte Carlo) algorithm with a Dirichlet process prior to estimate posterior distributions for the parameters of each normal distribution (μ and σ^2) and for the number of normals (k) in the mixture of normals.

The user provides a numeric vector of values for which the algorithm will infer a categorization of the values of the vector as originated from a number of normal distributions. Other user supplied parameters are the prior distributions for the mean (defined as normally distributed with user provided mean_prior_mean and mean_prior_sd) and standard deviation (defined as gamma distributed with user provided sd_prior_shape and sd_prior_rate) for the normal distributons to be inferred.

The default values provided and used in the documentation and help files for the package have worked well inferring the number of normal distributions in simulated data for 1 to 5 normals.

Perhaps the first parameter to start playing around with would be the maximum number of generations that the MCMC would run for. The default value used in the documentation is 10,000 generations which runs in a few minutes and makes good inferences for the number of normals and their parameters (μ and σ^2) for mixtures when the means of the normals are well separated and a vector of fewer than 1000 values is used. For examples with more categories with closer means and more values in the vector, more generations would be needed to achieve convergence.

DPPmcmc is capable of stopping automatically when an Effective Sample Size (ESS) threshold is achieved (default 500), as an indication that convergence of the MCMC has been achieved and no more generations are necessary for proper estimation of the posterior distributions of the parameters.

DPPmcmc can be extended to use other distributions in addition of the normal distribution. As an example of this we have included the class GammaModel to allow inferring Gamma distributions.

Three classes are made available by this package:

- **NormalModel**
- **GammaModel**
- **dppMCMC_C**

2 DPP: Classes and methods

2.1 Class: NormalModel

Objects of the NormalModel class are initialized with prior parameters to be used by the MCMC algorithm in dppMCMC_C class.

```
normal.model<-new(NormalModel,
                  mean_prior_mean=0.5,
                  mean_prior_sd=0.1,
                  sd_prior_shape=3,
                  sd_prior_rate=20,
                  estimate_concentration_parameter=TRUE,
                  concentration_parameter_alpha=10,
                  proposal_disturbance_sd=0.1)
```

Listing 1: Initializing an object of class NormalModel

The means for the normal distributions to be inferred from the data will be sampled from $N(\mu = mean\_prior\_mean, \sigma^2 = mean\_prior\_sd^2)$ and the variance for those means will be sampled from a $\Gamma(\alpha = sd\_prior\_shape, \beta = sd\_prior\_rate)$ provided as priors when instantiating NormalModel.

Additional parameters to be provide to the NormalModel object are:

estimate_concentration_parameter={TRUE,FALSE}

- if **FALSE** the same concentration parameter will used for all generations of the MCMC.
- if **TRUE** the concentration parameter will be estimated for each generation of the MCMC.

The concentration parameter is used for the initial allocation proposal, as a parameter of the SimulateChineseRestaurant method used for the prior allocation and for the allocation proposal function in sucesive generations of the MCMC. By allocation we refer here to the membership of each element of the user supplied **data** numeric vector belonging to each of the infered normal distributions. **data** is provided when an object of class dppMCMC_C is initialized.

concentration_parameter_alpha is used in the estimation of the concentration_parameter.

2.2 Class: NormalModel, visualizing parameters

The **getParameters** method shows the values of the different parameters initialized in a NormalModel object.

```
normal.model$getParameters()
$mean_prior_mean
[1] 0.5

$mean_prior_sd
[1] 0.1

$sd_prior_shape
[1] 3

$sd_prior_rate
```

```
[1] 20

$estimate_concentration_parameter
[1] TRUE

$concentration_parameter_alpha
[1] 10

$proposal_disturbance_sd
[1] 0.1
```

Listing 2: Parameters of a NormalModel object

2.3 Class: GammaModel

Objects of the GammaModel class are initialized with prior parameters to be used by the MCMC algorithm in dppMCMC_C class.

```
gamma.model<-new(GammaModel,
            shape_prior_mean=4,
            shape_prior_sd=1,
            rate_prior_mean=1.5,
            rate_prior_sd=0.54,
            estimate_concentration_parameter=TRUE,
            concentration_parameter_alpha=10,
            proposal_disturbance_sd=0.1)
```

Listing 3: Initializing an object of class GammaModel

The shapes for the gamma distributions to be inferred from the data will be sampled from $N(\mu = shape\_prior\_mean, \sigma^2 = shape\_prior\_sd^2)$ and the rate for those gamma distributions will be sampled from a $N(\mu = rate\_prior\_mean, \sigma^2 = rate\_prior\_sd^2)$ provided as priors when instantiating GammaModel.

Additional parameters to be provide to the GammaModel object are:

estimate_concentration_parameter={TRUE,FALSE}

- if **FALSE** the same concentration parameter will used for all generations of the MCMC.
- if **TRUE** the concentration parameter will be estimated for each generation of the MCMC.

2.4 Class: GammaModel, visualizing parameters

The **getParameters** method shows the values of the differnet parameters initialized in a GammaModel object.

```
gamma.model$getParameters()
[1] 4
```

```
$shape_prior_sd
[1] 1

$rate_prior_mean
[1] 1.5

$rate_prior_sd
[1] 0.54

$estimate_concentration_parameter
[1] TRUE

$concentration_parameter_alpha
[1] 10

$proposal_disturbance_sd
[1] 0.1
```

Listing 4: Parameters of a GammaModel object

2.5 Class: dppMCMC_C, initialization

Class dppMCMC_C provides the main functionality of the DPPmcmc package.

The dppMCMC_C class is initialized with parameters:

data: your data, a numeric vector

output: a prefix that will be used for the ouput files

normal_model: a NormalModel object with prior parameters for the normals

num_auxiliary_tables: the number of auxiliary tables

expected_k: a prior for the of expected number of means

power: it is used for the estimation of the likelihood usually taking values 0 or 1. If 1 (default) the likelihood of the parameters given the data is estimated at each generation, if set to 0 the likelihood estimate is set to 1.

verbose: TRUE or FALSE, when TRUE several log files will be generated (default: TRUE). When false only the main _mcmc.log file is created and updated in each generation.

```
my_dpp_analysis <- dppMCMC_C(data=y,
                         output = "output_prefix_",
                         model=normal.model,
                         num_auxiliary_tables=4,
                         expected_k=1.5,
                         power=1,
```

```
                                    verbose=TRUE)
```

Listing 5: Initializing an object of class dppMCMC_C

2.6 Class: dppMCMC_C, running the MCMC

The run method of dppMCMC_C receives the following parameters:

generations: the number of generations the MCMC will run for when auto_stop is set to false.

auto_stop: when FALSE the MCMC will run for **generations** generations, when TRUE the MCMC will run until the Effective Sample Size(ESS) of the parameters reach **min_ess** or the MCMC has run for **max_gen** generations. The **effectiveSize** function from package "**coda**" is used for ESS estimation.

max_gen: when **auto_stop** is true this will be the maximum number of generations the MCMC will run even if convergence is not achiveded (the **min_ess** is reached)

min_ess: when this **min_ess** is reached the MCMC will stop running as convergence for the estimation of the parameters will be assumed.

sample_freq: specifies the frequency at which the MCMC will write to the log files (e.g. set to 10 to write to the log files every 10 generations).

```
my_dpp_analysis$run(generations=1000,auto_stop=TRUE,
    max_gen = 10000,min_ess = 500,sample_freq=10)
```

Listing 6: runnning the MCMC using class dppMCMC_C

2.7 Class: dppMCMC_C, convenience methods

Users of this package would want to look at the logs created in order to diagnose the parameter estimation performed by the MCMC. For quick summary reports that are stored in memory in the dppMCMC_C object, the following convenience methods are provided:

getNumCategoryTrace and **getNumCategoryProbabilities** receive a **burnin_cutoff parameter** (with 0.25 as default value) that should range between 0 and 1. **getNumCategoryTrace** returns the trace of the estimated number of categories (k, number of normal distributions) inferred for the last generations of the MCMC after a percentage of results specified by burnin_cutoff is removed. **getNumCategoryProbabilities** returns the same information as probabilities of obtaining k=1,2,...,n.

```
#we get rid of the first 25% of the output (burn-in)
hist(my_dpp_analysis$getNumCategoryTrace(burnin_cutoff
    =0.25))
```

```
my_dpp_analysis$getNumCategoryProbabilities(burnin_
    cutoff=0.25)
```

Listing 7: displaying result posterior distribution from the dppMCMC_C object

2.8 Class: dppMCMC_C, the dpp_mcmc_object

Additional methods are available through the dpp_mcmc_object of dppMCMC_C. Of particular interest is the allocation vector with category/cluster/normal distribution assignments for each of the elements of the numeric vector provided as "data" when the dppMCMC_C is initialized. For example, if a total of three normal distributions are inferred from the data (k=3), a vector containing values 1, 2 and 3 will be returned by dpp_mcmc_object$getAllocationVector() indicating

```
my_dpp_analysis$dpp_mcmc_object$getAllocationVector()
```

Listing 8: accesing the dpp_mcmc_object: allocation vector

In a similar way, the parameters estimated for the last run of the MCMC are available in dpp_mcmc_object$getParamVector()

```
my_dpp_analysis$dpp_mcmc_object$getParamVector()
$means
[1] 1.2791923 0.1930994 0.7037923

$sds
[1] 0.09843063 0.04862315 0.04882795
```

Listing 9: accesing the dpp_mcmc_object : estimated parameters

3 Log files generated

The filename prefix specified in the creation of dppMCMC_C objects with the **output** parameter is used as a prefix for the naming the log files generated by running the MCMC. The following files are created:

<filenaming prefix>_mcmc.log: This trace file logs the likelihood of the allocation of the elements of the **data** numeric vector as originated from or belonging to a k number of normal distributions. The current proposed number of normal distributions k, the current concentration parameter and the current minimum effective sample size (min_ESS) are also logged for each generation of the MCMC. Note that the minimum effective sample size will remain 0 for all generations if auto_stop is set to FALSE. min_ESS will not be estimated as the MCMC will run for a fixed numer of generations.

<filenaming prefix>_param_1.log: The proposed normal means for each generation of the MCMC.

<filenaming prefix>_param_2.log: The proposed normal standard deviations for each generation of the MCMC.

<filenaming prefix>_allocation.log: The proposed allocation for each element of **data** as belonging to a finite number of normal distributions.

When verbose=FALSE on class creation only _mcmc.log will be created and updated.

4 Interpretation of results

As the MCMC relies on random sampling for proposing new parameters (number of normals k and the μ and σ^2 for each normal) that maximize the likelihood given the data, different runs can take different amounts of time to converge and obtain different posterior distributions. It is advisable to perform multiple runs over the same data to increase our confidence on the posterior distributions obtained before we make inferences based on those posterior distributions.

5 Troubleshooting

5.1 dppMCMC_C fails on object instantiation

In some systems a problem has been reported when creating the dppMCMC_C object.

This operation should take a fraction of a second but in some cases it does not finish initializing the object. It appears to be a problem with the R object passed as "data" parameter to the dppMCMC_C underlying Rcpp classes. The following "hack" solves the problem. An element is added to the "data" numeric vector and then removed so that the data is not affected but in practice allows for instantitation of dppMCMC_C.

```
#y will be passed as "data" parameter to dppMCMC_C
y <- c(y,rnorm(1,0,0.1))
y<-y[1:(length(y)-1)]
```

Listing 10: A hack that helps in some systems

5.2 I get the error message "could not find function "new"" when running dppMCMC_C with Rscript

The function **new** used to instantiate Rcpp objects in dppMCMC_C is part of the "methods" package that is loaded by default in the R shell but not by Rscript. To fix that add **library(methods)** to your R script before initializing dppMCMC_C objects.