Package ‘BayesBinMix’

January 15, 2017

Type Package

Title Bayesian Estimation of Mixtures of Multivariate Bernoulli
Distributions

Version 1.4

Date 2017-01-15

Author Panagiotis Papastamoulis

Maintainer Panagiotis Papastamoulis `<papapast@yahoo.gr>`

Description Fully Bayesian inference for estimating the number of clusters and related parameters to heterogeneous binary data.

Imports label.switching, foreach, doParallel, coda

License GPL-2

NeedsCompilation no

R topics documented:

`BayesBinMix-package` *Bayesian Estimation of Mixtures of Multivariate Bernoulli Distributions*

Description

Fully Bayesian inference for estimating the number of clusters and related parameters to heterogeneous binary data.

Details

This package can be used in order to cluster multivariate binary data (NAs are allowed). The main function of the package is `coupledMetropolis`.

The input is an $n \times d$ binary array where n and d denote the number of observations and dimension of the data. The underlying model is a mixture of independent multivariate Bernoulli distributions with an unknown number of components:

$$x_i \sim \sum_{k=1}^{K} \pi_k \prod_{j=1}^{d} f(x_{ij}; \theta_{kj}),$$

with $x_i = (x_{i1}, \ldots, x_{id})$; $d > 1$, independent for $i = 1, \ldots, n$. The term $f(x_{ij}; \theta_{kj})$ denotes the probability density function of the Bernoulli distribution with parameter $\theta_{kj} \in (0, 1)$. The number of clusters K is a random variable with support $\{1, \ldots, K_{\text{max}}\}$, where K_{max} is an upper bound for the number of clusters. The model uses the following prior assumptions:

$$K \sim \text{discrete}\{1, \ldots, K_{\text{max}}\}$$

$$(\pi_1, \ldots, \pi_K) | K \sim \text{Dirichlet}(\gamma, \ldots, \gamma)$$

$$\theta_{kj} | K \sim \text{Beta}(\alpha, \beta); \quad \text{independent for} \quad k = 1, \ldots, K; j = 1, \ldots, d.$$

The discrete distribution on the number of clusters it can be a truncated Poisson(1) or Uniform distribution. The model uses data augmentation by also considering the (latent) allocation variable z_i, which a priori assigns observation i to cluster $k = 1, \ldots, K$ with probability $P(z_i = k | K, \pi_1, \ldots, \pi_K) = \pi_k$ independently for $i = 1, \ldots, n$.

In order to infer the parameters of the model, a Markov chain Monte Carlo (MCMC) approach is adopted. Given K, the component-specific parameters π_k and θ_{kj} are integrated out and a collapsed allocation sampler which also updates the number of clusters (Nobile and Fearnside, 2007) is implemented. In the case that the observed data contains missing values, the algorithm simulates their values from the corresponding full conditional distribution. In order to improve the mixing of the simulated chain, a Metropolis-coupled MCMC sampler (Altekar et al., 2004) is incorporated. In particular, various heated chains are run in parallel and swaps are proposed between pairs of chains. The number of chains should be equal to the number of available cores. Each chain runs in parallel using the packages `foreach` and `doParallel`.

After inferring the most probable number of clusters, the simulated parameters which correspond to this specific value of K are post-processed in order to undo the label switching problem. For this purpose the `label.switching` package (Papastamoulis, 2016; see also Papastamoulis and Iliopoulos 2010, 2013 and Papastamoulis, 2014) is used.

Author(s)

Panagiotis Papastamoulis

Maintainer: Panagiotis Papastamoulis

References

Altekar G, Dwarkadas S, Huelsenbeck JP, Ronquist F. (2004): Parallel Metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference. Bioinformatics 20(3): 407-415.

Nobile A and Fearnside A (2007): Bayesian finite mixtures with an unknown number of components: The allocation sampler. Statistics and Computing, 17(2): 147-162.

Papastamoulis P. and Iliopoulos G. (2010). An artificial allocations based solution to the label switching problem in Bayesian analysis of mixtures of distributions. Journal of Computational and Graphical Statistics, 19: 313-331.

Papastamoulis P. and Iliopoulos G. (2013). On the convergence rate of Random Permutation Sampler and ECR algorithm in missing data models. Methodology and Computing in Applied Probability, 15(2): 293-304.

Papastamoulis P. (2014). Handling the label switching problem in latent class models via the ECR algorithm. Communications in Statistics, Simulation and Computation, 43(4): 913-927.

Papastamoulis P (2016): label.switching: An R package for dealing with the label switching problem in MCMC outputs. Journal of Statistical Software, 69(1): 1-24.

See Also

`coupledMetropolis`

`allocationSamplerBinMix` *The allocation sampler algorithm.*

Description

This function implements the collapsed allocation sampler of Nobile and Fearnside (2007) at the context of mixtures of multivariate Bernoulli distributions.

Usage

```
allocationSamplerBinMix(Kmax, alpha, beta, gamma, m, burn, data,
thinning, z.true, ClusterPrior, ejectionAlpha, Kstart, outputDir,
metropolisMoves, reorderModels, heat, zStart, LS, rsX, originalX, printProgress)
```

Arguments

Kmax Maximum number of clusters (integer, at least equal to two).

alpha First shape parameter of the Beta prior distribution (strictly positive). Defaults to 1.

beta Second shape parameter of the Beta prior distribution (strictly positive). Defaults to 1.

gamma Kmax-dimensional vector (positive) corresponding to the parameters of the Dirichlet prior of the mixture weights. Default value: rep(1,Kmax).

m Number of MCMC iterations.

burn The number of initial MCMC iterations that will be discarded as burn-in period.

data Binary data array (NAs not allowed here).

thinning Integer that defines a thinning of the reported MCMC sample. Under the default setting, every 5th MCMC iteration is saved.

z.true An optional vector of cluster assignments considered as the ground-truth clustering of the observations. Useful for simulations.

ClusterPrior Character string specifying the prior distribution of the number of clusters on the set $\{1, \ldots, K_{max}\}$. Available options: poisson or uniform. It defaults to the truncated Poisson distribution.

ejectionAlpha Probability of ejecting an empty component. Defaults to 0.2.

Kstart Initial value for the number of clusters. Defaults to 1.

outputDir The name of the produced output folder.

metropolisMoves

A vector of character strings with possible values M1, M2, M3, M4. Each entry specifies Metropolis-Hastings type moves on the latent allocation variables.

reorderModels Character string specifying whether to post-process the MCMC sample of each distinct generated value of K. The default setting is onlyMAP and in this case only the part of the MCMC sample that corresponds to the most probable number of clusters is reordered.

heat The temperature of the simulated chain, that is, a scalar in the set $(0, 1]$.

zStart n-dimensional integer vector of latent allocations to initialize the sampler.

LS Boolean indicating whether to post-process the MCMC sample using the label switching algorithms.

rsX Optional vector containing the row-sums of the observed data (NAs are allowed). It is required only in the case of missing values.

originalX Optional array containing the observed data (containing NAs). It is required only in the case of missing values.

printProgress Logical, indicating whether to print the progress of the sampler or not. Default: FALSE.

Details

The output is reordered according to the following label-switching solving algorithms: ECR, ECR-ITERATIVE-1 and STEPHENS. In most cases the results of these different algorithms are identical.

Note

This function is recursively called inside the `coupledMetropolis` function. There is no need to call it separately.

Author(s)

Panagiotis Papastamoulis

References

Nobile A and Fearnside A (2007): Bayesian finite mixtures with an unknown number of components: The allocation sampler. Statistics and Computing, 17(2): 147-162.

Papastamoulis P. and Iliopoulos G. (2010). An artificial allocations based solution to the label switching problem in Bayesian analysis of mixtures of distributions. Journal of Computational and Graphical Statistics, 19: 313-331.

Papastamoulis P. and Iliopoulos G. (2013). On the convergence rate of Random Permutation Sampler and ECR algorithm in missing data models. Methodology and Computing in Applied Probability, 15(2): 293-304.

Papastamoulis P. (2014). Handling the label switching problem in latent class models via the ECR algorithm. Communications in Statistics, Simulation and Computation, 43(4): 913-927.

Papastamoulis P (2016): label.switching: An R package for dealing with the label switching problem in MCMC outputs. Journal of Statistical Software, 69(1): 1-24.

See Also

`coupledMetropolis`

`collapsedGibbsBinMix` *collapsed Gibbs sampler*

Description

This function applied collapsed Gibbs sampling assuming that the number of mixture components is known.

Usage

```
collapsedGibbsBinMix(alpha, beta, gamma, K, m, burn,
data, thinning, z.true, outputDir)
```

Arguments

`alpha`	First shape parameter of the Beta prior distribution (strictly positive). Defaults to 1.
`beta`	Second shape parameter of the Beta prior distribution (strictly positive). Defaults to 1.
`gamma`	`K`-dimensional vector (positive) corresponding to the parameters of the Dirichlet prior of the mixture weights. Default value: `rep(1,K)`.
`K`	Number of clusters.
`m`	Number of MCMC iterations.
`burn`	The number of initial MCMC iterations that will be discarded as burn-in period.
`data`	Binary data array.
`thinning`	Integer that defines a thinning of the reported MCMC sample. Under the default setting, every 5th MCMC iteration is saved.
`z.true`	An optional vector of cluster assignments considered as the ground-truth clustering of the observations. Useful for simulations.
`outputDir`	The name of the produced output folder.

Note

Not really used.

Author(s)

Panagiotis Papastamoulis

`complete.loglikelihood` *complete log-likelihood*

Description

Returns the complete log-likelihood of the mixture.

Usage

```
complete.loglikelihood(x, z, pars)
```

Arguments

`x`	Binary data.
`z`	Latent allocations vector.
`pars`	Parameters of the mixture.

Value

Complete log-likelihood value.

Author(s)

Panagiotis Papastamoulis

`coupledMetropolis` *Metropolis-coupled Markov chain Monte Carlo sampler*

Description

Main function of the package. The algorithm consists of the allocation sampler combined with a MC3 scheme.

Usage

```
coupledMetropolis(Kmax, nChains, heats, binaryData, outPrefix,
ClusterPrior, m, alpha, beta, gamma, z.true, ejectionAlpha, burn)
```

Arguments

`Kmax` Maximum number of clusters (integer, at least equal to two).

`nChains` Number of parallel (heated) chains. Ideally, it should be equal to the number of available threads.

`heats` `nChains`-dimensional vector specifying the temperature of each chain: the 1st entry should always be equal to 1 and the rest of them lie on the set: $(0, 1]$.

`binaryData` The observed binary data (array). Missing values are allowed as long as the corresponding entries are denoted as `NA`.

`outPrefix` The name of the produced output folder. An error is thrown if the directory exists.

`ClusterPrior` Character string specifying the prior distribution of the number of clusters on the set $\{1, \ldots, K_{max}\}$. Available options: `poisson` or `uniform`. It defaults to the truncated Poisson distribution.

`m` The number of MCMC cycles. At the end of each cycle a swap between a pair of heated chains is attempted. Each cycle consists of 10 iterations.

`alpha` First shape parameter of the Beta prior distribution (strictly positive). Defaults to 1.

`beta` Second shape parameter of the Beta prior distribution (strictly positive). Defaults to 1.

`gamma` `Kmax`-dimensional vector (positive) corresponding to the parameters of the Dirichlet prior of the mixture weights. Default value: `rep(1,Kmax)`.

`z.true` An optional vector of cluster assignments considered as the ground-truth clustering of the observations. Useful for simulations.

`ejectionAlpha` Probability of ejecting an empty component. Defaults to 0.2.

`burn` Optional integer denoting the number of MCMC cycles that will be discarded as burn-in period.

Details

In the case that the most probable number of clusters is larger than 1, the output is post-processed using the label.switching package. In addition to the objects returned to the user (see `value` below), the complete output of the sampler is written to the directory `outPrefix`. It consists of the following files:

- K.allChains.txt `m`×`nChains` matrix containing the simulated values of the number of clusters (K) per chain.
- K.txt the `m` simulated values of the number of clusters (K) of the cold chain (posterior distribution).
- p.varK.txt the simulated values of the mixture weights (not identifiable).
- rawMCMC.mapK.KVALUE.txt the raw MCMC output which corresponds to the most probable model (not identifiable).
- reorderedMCMC-ECR-ITERATIVE1.mapK.KVALUE.txt the reordered MCMC output which corresponds to the most probable model, reordered according to the `ECR-ITERATIVE-1` algorithm.
- reorderedMCMC-ECR.mapK.KVALUE.txt the reordered MCMC output which corresponds to the most probable model, reordered according to the `ECR` algorithm.
- reorderedMCMC-STEPHENS.mapK.KVALUE.txt the reordered MCMC output which corresponds to the most probable model, reordered according to the `STEPHENS` algorithm.
- reorderedSingleBestClusterings.mapK.KVALUE.txt the most probable allocation of each observation after reordering the MCMC sample which corresponds to the most probable number of clusters.
- theta.varK.txt the simulated values of Bernoulli parameters (not identifiable).
- z-ECR-ITERATIVE1.mapK.KVALUE.txt the reordered simulated latent allocations which corresponds to the most probable model, reordered according to the `ECR-ITERATIVE-1` algorithm.
- z-ECR.mapK.KVALUE.txt the reordered simulated latent allocations which corresponds to the most probable model, reordered according to the `ECR` algorithm.
- z-KL.mapK.KVALUE.txt the reordered simulated latent allocations which corresponds to the most probable model, reordered according to the `STEPHENS` algorithm.
- z.varK.txt the simulated latent allocations (not identifiable).
- classificationProbabilities.mapK.KVALUE.csv the reordered classification probabilities per observation after reordering the most probable number of clusters with the `ECR` algorithm.
- xEstimated.txt Observed data with missing values estimated by their posterior mean estimate. This file is produced only in the case that the observed data contains missing values.

`KVALUE` will be equal to the inferred number of clusters. Note that the label switching part is omitted in case that the most probable number of clusters is equal to 1.

Value

The basic output of the sampler is returned to the following R objects:

`K.mcmc` object of class `mcmc` (see `coda` package) containing the simulated values (after burn-in) of the number of clusters for the cold chain.

`parameters.ecr.mcmc`
object of class `mcmc` (see `coda` package) containing the simulated values (after burn-in) of θ_{kj} (probability of success per cluster k and feature j) and π_k (weight of cluster k) for $k = 1, \ldots, K_{\mathrm{map}}$; $j = 1, \ldots, d$, where K_{map} denotes the most probable number of clusters. The output is reordered according to `ECR` algorithm.

`allocations.ecr.mcmc`
object of class `mcmc` (see `coda` package) containing the simulated values (after burn-in) of z_{kj} (allocation variables) for $k = 1, \ldots, K_{\mathrm{map}}$, $j = 1, \ldots, d$. The output is reordered according to `ECR` algorithm.

`classificationProbabilities.ecr`
data frame of the reordered classification probabilities per observation after reordering the most probable number of clusters with the `ECR` algorithm.

`clusterMembershipPerMethod`
data frame of the most probable allocation of each observation after reordering the MCMC sample which corresponds to the most probable number of clusters according to `ECR`, `STEPHENS` and `ECR-ITERATIVE-1` methods.

`K.allChains` `m`×`nChains` matrix containing the simulated values of the number of clusters (K) per chain.

`chainInfo` Number of cycles, burn-in period and acceptance rate of swap moves.

Author(s)

Panagiotis Papastamoulis

References

Altekar G, Dwarkadas S, Huelsenbeck JP, Ronquist F. (2004): Parallel Metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference. Bioinformatics 20(3): 407-415.

Nobile A and Fearnside A (2007): Bayesian finite mixtures with an unknown number of components: The allocation sampler. Statistics and Computing, 17(2): 147-162.

Papastamoulis P. and Iliopoulos G. (2010). An artificial allocations based solution to the label switching problem in Bayesian analysis of mixtures of distributions. Journal of Computational and Graphical Statistics, 19: 313-331.

Papastamoulis P. and Iliopoulos G. (2013). On the convergence rate of Random Permutation Sampler and ECR algorithm in missing data models. Methodology and Computing in Applied Probability, 15(2): 293-304.

Papastamoulis P. (2014). Handling the label switching problem in latent class models via the ECR algorithm. Communications in Statistics, Simulation and Computation, 43(4): 913-927.

Papastamoulis P (2016): label.switching: An R package for dealing with the label switching problem in MCMC outputs. Journal of Statistical Software, 69(1): 1-24.

Examples

```
#generate dataset from a mixture of 2 ten-dimensional Bernoulli distributions.
set.seed(1)
d <- 10 # number of columns
n <- 50 # number of rows (sample size)
K <- 2   # true number of clusters
p.true <- myDirichlet(rep(10,K)) # true weight of each cluster
z.true <- numeric(n) # true cluster membership
z.true <- sample(K,n,replace=TRUE,prob = p.true)
#true probability of positive responses per cluster:
theta.true <- array(data = NA, dim = c(K,d))
for(j in 1:d){
    theta.true[,j] <- rbeta(K, shape1 = 1, shape2 = 1)
}
x <- array(data=NA,dim = c(n,d)) # data: n X d array
for(k in 1:K){
        myIndex <- which(z.true == k)
        for (j in 1:d){
                x[myIndex,j] <- rbinom(n = length(myIndex),
size = 1, prob = theta.true[k,j])
        }
}
# number of heated paralled chains
nChains <- 2
heats <- seq(1,0.8,length = nChains)
## Not run:
cm <- coupledMetropolis(Kmax = 10,nChains = nChains,heats =  heats,
binaryData = x, outPrefix = BayesBinMixExample,
ClusterPrior = poisson, m = 1100, burn = 100)
# print summary using:
print(cm)

## End(Not run)
# it is also advised to use z.true = z.true in order to directly compare with
# the true values. In general it is advised to use at least 4 chains with
# heats <- seq(1,0.3,length = nChains)
```

`dealWithLabelSwitching` *Label switching algorithms*

Description

This is a wrapper for the `label.switching` package. It is used to post-process the generated MCMC sample in order to undo the label switching problem. This function is called internally to the `coupledMetropolis` command.

Usage

```
dealWithLabelSwitching(outDir, reorderModels, binaryData, z.true)
```

Arguments

`outDir` The directory where the output of `coupledMetropolis` was previously produced.

`reorderModels` Boolean value indicating whether to reorder the MCMC corresponding to each distinct generated value of number of clusters or not.

`binaryData` The input data.

`z.true` An optional vector of cluster assignments considered as the ground-truth clustering of the observations. Useful for simulations.

Details

See the `label.switching` package.

Author(s)

Panagiotis Papastamoulis

References

Papastamoulis P. and Iliopoulos G. (2010). An artificial allocations based solution to the label switching problem in Bayesian analysis of mixtures of distributions. Journal of Computational and Graphical Statistics, 19: 313-331.

Papastamoulis P. and Iliopoulos G. (2013). On the convergence rate of Random Permutation Sampler and ECR algorithm in missing data models. Methodology and Computing in Applied Probability, 15(2): 293-304.

Papastamoulis P. (2014). Handling the label switching problem in latent class models via the ECR algorithm. Communications in Statistics, Simulation and Computation, 43(4): 913-927.

Papastamoulis P (2016): label.switching: An R package for dealing with the label switching problem in MCMC outputs. Journal of Statistical Software, 69(1): 1-24.

`dealWithLabelSwitchingMissing` *Label switching algorithms for the case of missing data*

Description

This is a wrapper for the `label.switching` package. It is used to post-process the generated MCMC sample in order to undo the label switching problem. This function is called internally to the `coupledMetropolis` command.

Usage

```
dealWithLabelSwitchingMissing(outDir, reorderModels, binaryData, z.true)
```

Arguments

outDir	The directory where the output of coupledMetropolis was previously produced.
reorderModels	Boolean value indicating whether to reorder the MCMC corresponding to each distinct generated value of number of clusters or not.
binaryData	The input data.
z.true	An optional vector of cluster assignments considered as the ground-truth clustering of the observations. Useful for simulations.

Details

See the label.switching package.

Author(s)

Panagiotis Papastamoulis

References

Papastamoulis P. and Iliopoulos G. (2010). An artificial allocations based solution to the label switching problem in Bayesian analysis of mixtures of distributions. Journal of Computational and Graphical Statistics, 19: 313-331.

Papastamoulis P. and Iliopoulos G. (2013). On the convergence rate of Random Permutation Sampler and ECR algorithm in missing data models. Methodology and Computing in Applied Probability, 15(2): 293-304.

Papastamoulis P. (2014). Handling the label switching problem in latent class models via the ECR algorithm. Communications in Statistics, Simulation and Computation, 43(4): 913-927.

Papastamoulis P (2016): label.switching: An R package for dealing with the label switching problem in MCMC outputs. Journal of Statistical Software, 69(1): 1-24.

gibbsBinMix *Standard Gibbs sampler*

Description

This function implements full Gibbs sampling to simulate an MCMC sample from the posterior distribution assuming known number of mixture components.

Usage

```
gibbsBinMix(alpha, beta, gamma, K, m, burn, data,
thinning, z.true, outputDir)
```

Arguments

`alpha` First shape parameter of the Beta prior distribution (strictly positive). Defaults to 1.

`beta` Second shape parameter of the Beta prior distribution (strictly positive). Defaults to 1.

`gamma` `Kmax`-dimensional vector (positive) corresponding to the parameters of the Dirichlet prior of the mixture weights. Default value: `rep(1,Kmax)`.

`K` Number of clusters.

`m` Number of MCMC iterations.

`burn` Burn-in period.

`data` Binary data.

`thinning` Thinning of the simulated chain.

`z.true` An optional vector of cluster assignments considered as the ground-truth clustering of the observations. Useful for simulations.

`outputDir` Output directory.

Details

Not really used.

Author(s)

Panagiotis Papastamoulis

`myDirichlet` *Simulate from Dirichlet distribution*

Description

This function simulates random vectors from a Dirichlet distribution.

Usage

```
myDirichlet(alpha)
```

Arguments

`alpha` Vector of positive numbers denoting the parameters of the Dirichlet distribution.

Author(s)

Panagiotis Papastamoulis

`print.bbm.object` *Print function*

Description

This function prints a summary of objects returned by the `coupledMetropolis` function.

Usage

```
## S3 method for class bbm.object
print(x, printSubset, ...)
```

Arguments

`x`	An object of class `bbm.object`, which is returned by the `coupledMetropolis` function.
`printSubset`	Logical indicating whether to print the header or the whole matrix of estimates. Default value: TRUE.
`...`	ignored.

Details

The function prints the estimated distribution of the number of clusters, the estimated number of observations assigned to each cluster after post-processing the output with three label switching algorithms, as well as the header of the posterior mean estimates of θ_{kj} (probability of success for cluster k and feature j) (conditionally on the most probable number of clusters).

Author(s)

Panagiotis Papastamoulis

`toSolve` *An equation to solve*

Description

Approximately solve the equation (25) of Nobile and Fearnside (2007).

Usage

```
toSolve(a, n, p0)
```

Arguments

`a`	Positive number.
`n`	Positive integer.
`p0`	Probability.

Author(s)

Panagiotis Papastamoulis

Index