Package 'copre'

October 12, 2022

Type Package

Title Tools for Nonparametric Martingale Posterior Sampling

Version 0.1.0

Description Performs Bayesian nonparametric density estimation using Martingale posterior distributions and including the Copula Resampling (CopRe) algorithm. Also included are a Gibbs sampler for the marginal Mixture of Dirichlet Process (MDP) model and an extension to include full uncertainty quantification via a new Polya completion algorithm for the MDP. The CopRe and Polya samplers generate random nonparametric distributions as output, leading to complete nonparametric inference on posterior summaries. Routines for calculating arbitrary functionals from the sampled distributions are included as well as an important algorithm for finding the number and location of modes, which can then be used to estimate the clusters in the data using, for example, k-means. Implements work developed in Moya B., Walker S. G. (2022) <doi:10.48550/arxiv.2206.08418>, Fong, E., Holmes, C., Walker, S. G. (2021) <doi:10.48550/arxiv.2103.15671>, and Escobar M. D., West, M. (1995) <doi:10.1080/01621459.1995.10476550>.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.2.1

LinkingTo Rcpp, RcppArmadillo, BH

Imports Rcpp, pracma, ggplot2, abind, dirichletprocess

Depends R (>= 2.10)

NeedsCompilation yes

Author Blake Moya [cre, aut],
The University of Texas at Austin [cph, fnd]

Maintainer Blake Moya <blakemoya@utexas.edu>

Repository CRAN

Date/Publication 2022-08-16 10:00:04 UTC

R topics documented:

copre-package *CopRe Tools for Nonparametric Martingale Posterior Sampling*

Description

Performs Bayesian nonparametric density estimation using Martingale posterior distributions and including the Copula Resampling (CopRe) algorithm. Also included are a Gibbs sampler for the marginal Mixture of Dirichlet Process (MDP) model and an extension to include full uncertainty quantification via a new Polya completion algorithm for the MDP. The CopRe and Polya samplers generate random nonparametric distributions as output, leading to complete nonparametric inference on posterior summaries. Routines for calculating arbitrary functionals from the sampled distributions are included as well as an important algorithm for finding the number and location of modes, which can then be used to estimate the clusters in the data using, for example, k-means.

Author(s)

Blake Moya <blakemoya@utexas.edu>

References

- Fong, E., Holmes, C., Walker, S. G. (2021). Martingale Posterior Distributions. arXiv. DOI: doi: 10.48550/arxiv.2103.15671
- Moya B., Walker S. G. (2022). Uncertainty Quantification and the Marginal MDP Model. arXiv. DOI: doi: 10.48550/arxiv.2206.08418
- Escobar M. D., West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association. DOI: doi: 10.1080/01621459.1995.10476550

copre *Copula Resampling*

Description

A function that samples predictive distributions for univariate continuous data using the bivariate gaussian copula.

Usage

```
copre(
  data,
  N,
  k,
  rho = 0.91,
  grd_res = 1000,
  nthreads = parallel::detectCores(),
  gpu = FALSE,
  gpu_path = NULL,
  gpu_odir = NULL,
  gpu_seed = 1234
)
```

Arguments

`data` The from which to sample predictive distributions.

`N` The number of unobserved data points to resample for each chain.

`k` The number of predictive distributions to sample.

`rho` A scalar concentration parameter.

`grd_res` The number of points on which to evaluate the predictive distribution.

`nthreads` The number of threads to call for parallel execution.

`gpu` A logical value indicating whether or not to use the CUDA implementation of the algorithm.

`gpu_path` The path to the CUDA implementation source code.

`gpu_odir` A directory to output the compiled CUDA code.

`gpu_seed` A seed for the CUDA random variates.

Value

A `copre_result` object, whose underlying structure is a list which contains the following components:

References

Fong, E., Holmes, C., Walker, S. G. (2021). Martingale Posterior Distributions. arXiv. DOI: doi: 10.48550/arxiv.2103.15671

Examples

```
res_cop <- copre(rnorm(50), 10, 10, nthreads = 1)
```

---

functional *Obtain Functionals from a CopRe Result*

Description

Obtain Functionals from a CopRe Result

Usage

```
functional(obj, f, mean = FALSE)
```

Arguments

obj	A copre_result object.
f	A list of functions.
mean	A logical value indicating whether or not to obtain the functional from the pointwise mean of the sampled distributions or from each individually.

Value

The integral over the copre_result grid of the functions in the list multiplied by the density of each sample distribution in obj.

---

modes *Mode Extractor*

Description

Extracts the modes from a copre_result or mdp_result object.

Usage

```
modes(obj, mean = TRUE, grd = NULL, anti = FALSE)

## S3 method for class 'mdpolya_result'
modes(obj, mean = TRUE, grd = NULL, anti = FALSE)

## S3 method for class 'grideval_result'
modes(obj, mean = TRUE, grd = NULL, anti = FALSE)

n_modes(obj, mean = TRUE, grd = NULL, anti = FALSE)
```

Arguments

obj A copre_result or mdp_result object

mean A logical value indicating whether to count the modes of the mean density of each of the individual sampled density

grd For mdpolya_result, a grid on which to evaluate the object.

anti A logical value indicating whether to extract true modes or anti-modes.

Value

A matrix of modes values in the support of the copre_result density

Methods (by class)

- modes(mdpolya_result): Mode-counting method for mdpolya_result objects.
- modes(grideval_result): Mode-counting method for grideval_result objects.

Functions

- n_modes(): Counts the modes from a copre_result or mdp_result object.

moments *Moment calculation generic function*

Description

Moment calculation generic function

Usage

```
moments(obj, mom, cntrl = TRUE, grd = NULL)

## S3 method for class 'mdpolya_result'
moments(obj, mom, cntrl = TRUE, grd = NULL)

## S3 method for class 'grideval_result'
moments(obj, mom, cntrl = TRUE, grd = NULL)
```

Arguments

obj The object for which a moment will be calculated.

mom A numeric scalar indicating the moment to calculate.

cntrl A logical value indicating whether the moment should be central or not. Defaults to TRUE.

grd A numeric vector of grid values on which the density function samples in ‘obj‘ should be calculated for trapezoidal integration.

Value

A vector of moment values for each sampled distribution in `obj`.

Methods (by class)

- `moments(mdpolya_result)`: Moment calculation method for `mdpolya_result` objects.
- `moments(grideval_result)`: Moment calculation method for `grideval_result` objects.

`plot.copre_result` *CopRe Result Plotter*

Description

CopRe Result Plotter

Usage

```
## S3 method for class 'copre_result'
plot(x, ..., func = "density", confint = NULL)
```

Arguments

`x`	A `copre_result` object.
`...`	Additional arguments discarded from `plot`.
`func`	Either ’distribution’, ’density’, or ’gradient’.
`confint`	A decimal value indicating the confidence interval width (e.g. 0.95 for a 95 no confidence intervals will be drawn.

Value

A `ggplot` object.

`plot.grideval_result` *Plotting method for* `grideval_result` *objects*

Description

Plotting method for `grideval_result` objects

Usage

```
## S3 method for class 'grideval_result'
plot(x, ..., confint = NULL)
```

Arguments

`x` A `grideval_result` object.

`...` Additional arguments discarded from `plot`.

`confint` A decimal value indicating the confidence interval width (e.g. 0.95 for a 95 percent confidence interval). Defaults to `NULL`, in which case no confidence intervals will be drawn.

Value

A `ggplot` object.

`plot.mdpolya_result` *Plotting method for* `mdpolya_result` *objects*

Description

Plotting method for `mdpolya_result` objects

Usage

```
## S3 method for class 'mdpolya_result'
plot(x, ..., grd = NULL, func = "density", confint = NULL, nthreads = 1)
```

Arguments

`x` An `mdpolya_result` object, substituting `obj`.

`...` Additional arguments discarded from `plot`.

`grd` For `mdpolya_result` objects, a numeric vector of `m` grid points.

`func` Either 'distribution', 'density', or 'gradient'.

`confint` A decimal value indicating the confidence interval width (e.g. 0.95 for a 95 no confidence intervals will be drawn.

`nthreads` The number of parallel threads to launch with OpenMP.

Value

A `ggplot` object.

polya *Polya Completion for Marginal MDP Samples*

Description

Polya Completion for Marginal MDP Samples

Usage

```
polya(res_mdp, epsilon = 0.01, upsilon = 0.01, nthreads = 1)

## S3 method for class 'mdp'
polya(res_mdp, epsilon = 0.01, upsilon = 0.01, nthreads = 1)

## S3 method for class 'dirichletprocess'
polya(res_mdp, epsilon = 0.01, upsilon = 0.01, nthreads = 1)
```

Arguments

`res_mdp`	Samples from a marginal MDP model.
`epsilon`	The desired maximum weight associated with the final remainder component.
`upsilon`	The portion of samples which do not meet the desired epsilon.
`nthreads`	UNSTABLE: The number of parallel threads to launch with OpenMP, not recommended due to induced instability.

Value

If `res_mdp` was an `mdpolya_result` object, returns another `mdpolya_result`object with `phi`, `eta` and `args` entries as in [mdp()]. If `res_mdp` was a `dirichletprocess` object, returns another `dirichletprocess` object with new components and altered weights.

Methods (by class)

- `polya(mdp)`: Polya extension to a `mdpolya_result` object.
- `polya(dirichletprocess)`: Polya extension to a `dirichletprocess` object.

References

Moya B., Walker S. G. (2022). Uncertainty Quantification and the Marginal MDP Model. arXiv. DOI: doi: 10.48550/arxiv.2206.08418

Examples

```
res_mdp <- mdp(rnorm(50), 10)
res_pol <- polya(res_mdp, nthreads = 1)
```

[[.mdpolya_result *Marginal MDP Sampler*

Description

Marginal MDP Sampler

Usage

```
## S3 method for class 'mdpolya_result'
obj[[i]]

mdp(
  data,
  k,
  alpha = 1,
  mu = 21,
  tau = 25,
  s = 4,
  S = 2,
  c = 2,
  C = 4,
  a = 21,
  A = 21,
  w = 1,
  W = 100,
  fix_a = FALSE,
  fix_m = FALSE,
  fix_t = FALSE,
  burn = 1000,
  thin = 150
)
```

Arguments

`obj`	An `mdpolya_result` object.
`i`	A numeric vector of sample indices.
`data`	A numeric vector of `n` observation values.
`k`	The number of sampling iterations to record, will be truncated if a non-integer.
`alpha`	The concentration parameter for the Dirichlet Process prior.
`mu`	The mean parameter for the Normal-Inverse-Gamma prior.
`tau`	The variance parameter for the Normal-Inverse-Gamma prior.
`s`	The shape parameter for the Normal-Inverse-Gamma prior.
`S`	The scale parameter for the Normal-Inverse-Gamma prior.

`c` The shape parameter for the Gamma prior on alpha.

`C` The scale parameter for the Gamma prior on alpha.

`a` The mean parameter for the Normal prior on mu.

`A` The variance parameter for the Normal prior on mu.

`w` The shape parameter for the Inverse-Gamma prior on tau.

`W` The scale parameter for the Inverse-Gamma prior on tau.

`fix_a` A logical value indicating whether or not to fix alpha at its initial value.

`fix_m` A logical value indicating whether or not to fix mu at its initial value.

`fix_t` A logical value indicating whether or not to fix tau at its initial value.

`burn` The number of initial sampling iterations to discard, will be truncated if a non-integer.

`thin` The number of sampling iterations to discard between records, will be truncated if a non-integer.

Value

An `mdpolya_result` object. A list with four entries: * `theta`: An array of dimension `[k, n, 3]` encoding the component label, mean, and standard deviation for each data point for each iteration. This represents the samples from the Polya posterior distribution of the marginal MDP model. * `eta`: A matrix of dimension `[k, 5]` encoding the hyperparameter values for each iteration. * `args`: A list of input arguments. * `phi`: A list of matrices encoding the unique values from `theta` and associated weights for each iteration.

Functions

- `[[`: Subset method for `mdpolya_result` objects

References

- Moya B., Walker S. G. (2022). Uncertainty Quantification and the Marginal MDP Model. arXiv. DOI: doi: 10.48550/arxiv.2206.08418
- Escobar M. D., West, M. (1995) Bayesian Density Estimation and Inference Using Mixtures. Journal of the American Statistical Association. DOI: doi: 10.1080/01621459.1995.10476550

See Also

[polya()]

Examples

```
res_mdp <- mdp(rnorm(50), 10)
```

$.grideval_result *Grid evaluation of* copre_result *and* mdpolya_result *objects*

Description

Grid evaluation of copre_result and mdpolya_result objects

Usage

```
## S3 method for class 'grideval_result'
obj$name

## S3 method for class 'grideval_result'
obj[[i]]

grideval(obj, grd = NULL, func = "density", nthreads = 1)

## S3 method for class 'copre_result'
grideval(obj, grd = NULL, func = "density", nthreads = 1)

## S3 method for class 'mdpolya_result'
grideval(obj, grd = NULL, func = "density", nthreads = 1)
```

Arguments

obj	A copre_result or mdpolya_result object.
name	The name of the attribute to access (i.e. func, grid, or args).
i	A numeric vector of sample indices.
grd	For mdpolya_result objects, a numeric vector of m grid points.
func	Either ’distribution’, ’density’, or ’gradient’.
nthreads	The number of parallel threads to launch with OpenMP.

Value

A grideval_result object, which is a matrix with dimension [k, m] of evaluated sample functions, with the following attributes: * func: The evaluated function. * grid: The grid points on which each of the k rows was evaluated. * args: A copy of the args entry from obj.

Methods (by class)

- grideval(copre_result): Grid evaluation method for copre_result objects.
- grideval(mdpolya_result): Grid evaluation method for mdpolya_result objects.

Functions

- $: Attribute access method for grideval_result objects
- [[: Subset method for grideval_result objects

Index