Package: PosteriorBootstrap (via r-universe)

August 13, 2024

Title Non-Parametric Sampling with Parallel Monte Carlo

Version 0.1.3

Description An implementation of a non-parametric statistical model using a parallelised Monte Carlo sampling scheme. The method implemented in this package allows non-parametric inference to be regularized for small sample sizes, while also being more accurate than approximations such as variational Bayes. The concentration parameter is an effective sample size parameter, determining the faith we have in the model versus the data. When the concentration is low, the samples are close to the exact Bayesian logistic regression method; when the concentration is high, the samples are close to the simplified variational Bayes logistic regression. The method is described in full in the paper Lyddon, Walker, and Holmes (2018), ``Nonparametric learning from Bayesian models with randomized objective functions" <arXiv:1806.11544>.

Language en-GB

License MIT + file LICENSE

Encoding UTF-8

Imports e1071 (>= 1.7.1), MASS (>= 7.3.51.1), utils (>= 3.4.3)

Suggests BH (>= 1.81.0), covr (>= 3.3.0), dplyr (>= 0.7.4), ggplot2 (>= 3.1.1), gridExtra (>= 2.3), knitr (>= 1.21), lintr (>= 1.0.3), RcppEigen (>= 0.3.3), RcppParallel (>= 5.1.7), rmarkdown (>= 1.11), roxygen2 (>= 6.1.1), rstan (>= 2.18.2), testthat (>= 2.0.1), tibble (>= 2.1.1)

VignetteBuilder knitr, rmarkdown

URL https://github.com/alan-turing-institute/PosteriorBootstrap/

BugReports https://github.com/alan-turing-institute/PosteriorBootstrap/issues

RoxygenNote 7.2.3

Repository https://alan-turing-institute.r-universe.dev

RemoteUrl https://github.com/alan-turing-institute/posteriorbootstrap

RemoteRef HEAD

RemoteSha c7c370b9b47375ccbc73d993600caf5a76b58833

Contents

`PosteriorBootstrap-package`

A package with a parallel approach for adaptive non-parametric learning

---

Description

The PosteriorBootstrap package provides two categories of functions. The first category returns or loads the system files that ship with the package: get_stan_file, get_german_credit_file, get_german_credit_dataset. The second category performs statistical sampling: draw_stick_breaks and draw_logit_samples (for adaptive non-parametric learning of the logistic regression model).

Details

Please see the vignette for sample usage and performance metrics.

Author(s)

Maintainer: James Robinson <james.em.robinson@gmail.com>

Authors:

- Simon Lyddon <simon.lyddon@stats.ox.ac.uk>
- Miguel Morin <info@turing.ac.uk>

Other contributors:

- The Alan Turing Institute <info@turing.ac.uk> [copyright holder]

See Also

Useful links:

- https://github.com/alan-turing-institute/PosteriorBootstrap/
- Report bugs at https://github.com/alan-turing-institute/PosteriorBootstrap/issues

`draw_logit_samples` *Draw adaptive non-parametric learning samples for logistic regression*

Description

`draw_logit_samples` returns samples of the parameter of interest in a logistic regression.

Usage

```
draw_logit_samples(
  x,
  y,
  concentration,
  n_bootstrap = 100,
  posterior_sample = NULL,
  gamma_mean = NULL,
  gamma_vcov = NULL,
  threshold = 1e-08,
  num_cores = 1,
  show_progress = FALSE
)
```

Arguments

`x`	The features of the data.
`y`	The outcomes of the data (either `0` or `1`).
`concentration`	The parameter `c` in the paper (page 3, formula 3),
`n_bootstrap`	The number of bootstrap samples required.
`posterior_sample`	The function can take samples from the posterior to generate non-parametric-learning samples, or it can take NULL and the posterior is assumed normal N(`gamma_mean`, `gamma_vcov`). If provided, the posterior sample must have a number of columns equal to the number of covariates and a number of rows equal or larger than the 'n_bootstrap' (as the algorithm draws a new sample based on a single draw of the posterior sample).
`gamma_mean`	In case `posterior_sample` is NULL, the mean for the centering model (equation 9, page 4).
`gamma_vcov`	In case `posterior_sample` is NULL, the variance-covariance of the centering model for gamma (equation 9, page 4).
`threshold`	The threshold of stick remaining below which the function stops looking for more stick-breaks. It correspondes to epsilon in the paper, at the bottom of page 5 and in algorithm 2 in page 12.
`num_cores`	Number of processor cores for the parallel run of the algorithm. See `mc.cores` in mclapply for details.
`show_progress`	Boolean whether to show the progress of the algorithm in a progress bar.

Details

This function implements the non-parametric-learning algorithm, which is algorithm 2 in page 12 in the paper. It uses a mixture of Dirichlet processes and stick-breaking to find the number of posterior samples and logistic regression to find the randomized parameter of interest. For examples, see the vignette.

Value

A matrix of bootstrap samples for the parameter of interest.

`draw_stick_breaks` *Draw stick-breaks depending on a concentration parameter*

Description

`draw_stick_breaks` returns a vector with the breaks of a stick of length 1.

Usage

```
draw_stick_breaks(
  concentration = 1,
  min_stick_breaks = 100,
  threshold = 1e-08,
  seed = NULL
)
```

Arguments

`concentration`	The parameter `c` in the paper (page 3, formula 3), which is an effective sample size.
`min_stick_breaks`	The minimal number of stick-breaks.
`threshold`	The threshold of stick remaining below which the function stops looking for more stick-breaks. It corresponds to epsilon in the paper, at the bottom of page 5 and in algorithm 2 in page 12.
`seed`	A seed to start the sampling.

Details

This function implements the stick-breaking process for non-parametric learning described in section 2 of the supplementary material. The name "stick-breaking" comes from a stick of unit length that we need to break into a number of items. This code implements algorithm 2 and the stick-breaking function calculates the parameter T in algorithm 1, which is the only difference between the two algorithms. The code uses the Beta distribution as that distribution is part of the definition of the stick-breaking process. The function draws from the beta distribution, e.g. `b_1`, `b_2`, `b_3`, ..., and computes the stick breaks as `b_1`, `(1-b_1)*b_2`, `(1-b_1)*(1-b_2)*b_3`, ... The length remaining in the stick at each step is `1-b_1`, `(1-b_1)* (1-b_2)`, `(1-b_1)*(1-b_2)*(1-b_3)`, ... so the latter converges to zero.

Value

A vector of stick-breaks summing to one.

Examples

```
draw_stick_breaks(1)
draw_stick_breaks(1, min_stick_breaks = 10)
draw_stick_breaks(1, min_stick_breaks = 10, threshold = 1e-8)
```

---

get_file *Get a file from extdata by name*

Description

Get a file from extdata by name

Usage

```
get_file(name)
```

Arguments

name	The filename that is requested

Value

The requested file

Examples

```
f <- get_file('bayes_logit.stan')
writeLines(readLines(f))
```

---

get_german_credit_dataset *Load and pre-process the dataset that ships with the package*

Description

Load and pre-process the dataset that ships with the package

Usage

```
get_german_credit_dataset(
  scale = TRUE,
  add_constant_term = TRUE,
  download_destination = NULL
)
```

Arguments

`scale`	Whether to scale the features to have mean 0 and variance 1.
`add_constant_term`	Whether to add a constant term as the first feature.
`download_destination`	Provide a filepath if you want to download the dataset from source. Note that although the original dataset has 20 features (some of them qualitative), the numeric dataset has 24 features.

Value

A list with fields `x` for features and `y` for outcomes.

Examples

```
german <- get_german_credit_dataset()
head(german$y)
head(german$x)
```

`get_german_credit_file` *Get the file with the German Statlog credit dataset*

Description

The file contains a local copy of the German Statlog credit dataset with 1,000 observations and 24 features. The data page is at: https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data) and the original files at: http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/ We use the file 'german.data-numeric', which has 24 covariates instead of the 20 in the original data (as some are qualitative).

Usage

```
get_german_credit_file()
```

Value

A file with the plain-text raw data for the German Statlog credit that ships with this package (extension `.dat`).

Examples

```
f <- get_german_credit_file()
writeLines(readLines(f, n=5))
```

`get_stan_file` *Get the Stan file with Bayesian Logistic Regression*

Description

Get the Stan file with Bayesian Logistic Regression

Usage

```
get_stan_file()
```

Value

An RStan file with the model for variational Bayes that ships with this package (extension `.stan`).

Examples

```
f <- get_stan_file()
writeLines(readLines(f))
```

Index