Package ‘loo’

December 10, 2015

Type Package

Title Efficient Leave-One-Out Cross-Validation and WAIC for Bayesian Models

Version 0.1.4

Date 2015-12-09

Maintainer Jonah Gabry <jsg2201@columbia.edu>

URL https://github.com/jgabry/loo

BugReports https://github.com/jgabry/loo/issues

Description Efficient approximate leave-one-out cross-validation (LOO) using Pareto smoothed importance sampling (PSIS), a new procedure for regularizing importance weights. As a byproduct of the calculations, we also obtain approximate standard errors for estimated predictive errors, and for the comparison of predictive errors between models. We also compute the widely applicable information criterion (WAIC).

License GPL (>= 3)

LazyData TRUE

Depends R (>= 3.1.2)

Imports graphics, matrixStats (>= 0.14.1), parallel, stats

Suggests knitr, testthat

VignetteBuilder knitr

RoxygenNote 5.0.1

NeedsCompilation no

Author Aki Vehtari [aut],
Andrew Gelman [aut],
Jonah Gabry [cre, aut],
Juho Piironen [ctb],
Ben Goodrich [ctb]

Repository CRAN

Date/Publication 2015-12-10 09:49:17

R topics documented:

loo-package 2
compare 5
extract_log_lik 6
loo 7
print.loo 10
psislw 11
waic 12

Index **14**

`loo-package` *Efficient LOO and WAIC for Bayesian models*

Description

This package implements the methods described in

Vehtari, A., Gelman, A., and Gabry, J. (2015). Efficient implementation of leave-one-out cross-validation and WAIC for evaluating fitted Bayesian models. arXiv preprint: `http://arxiv.org/abs/1507.04544`.

The package documentation is largely based on excerpts from the paper.

Authors

Aki Vethari, Andrew Gelman, Jonah Gabry.

Summary

Leave-one-out cross-validation (LOO) and the widely applicable information criterion (WAIC) are methods for estimating pointwise out-of-sample prediction accuracy from a fitted Bayesian model using the log-likelihood evaluated at the posterior simulations of the parameter values. LOO and WAIC have various advantages over simpler estimates of predictive error such as AIC and DIC but are less used in practice because they involve additional computational steps. This package implements the fast and stable computations for LOO and WAIC laid out in Vehtari, Gelman, and Gabry (2015). From existing posterior simulation draws, we compute LOO using Pareto Smoothed Importance Sampling (PSIS; Vehtari and Gelman, 2015), a new procedure for regularizing importance weights. As a byproduct of our calculations, we also obtain approximate standard errors for estimated predictive errors and for comparing of predictive errors between two models.

Details

After fitting a Bayesian model we often want to measure its predictive accuracy, for its own sake or for purposes of model comparison, selection, or averaging. Cross-validation and information criteria are two approaches for estimating out-of-sample predictive accuracy using within-sample fits.

Exact cross-validation requires re-fitting the model with different training sets. Approximate leave-one-out cross-validation (LOO) can be computed easily using importance sampling (Gelfand, Dey, and Chang, 1992, Gelfand, 1996) but the resulting estimate is noisy, as the variance of the importance weights can be large or even infinite (Peruggia, 1997, Epifani et al., 2008). Here we propose a novel approach that provides a more accurate and reliable estimate using importance weights that are smoothed by fitting a generalized Pareto distribution to the upper tail of the distribution of the importance weights.

WAIC (the widely applicable or Watanabe-Akaike information criterion; Watanabe, 2010) can be viewed as an improvement on the deviance information criterion (DIC) for Bayesian models. DIC has gained popularity in recent years in part through its implementation in the graphical modeling package BUGS (Spiegelhalter, Best, et al., 2002; Spiegelhalter, Thomas, et al., 1994, 2003), but it is known to have some problems, arising in part from it not being fully Bayesian in that it is based on a point estimate (van der Linde, 2005, Plummer, 2008). For example, DIC can produce negative estimates of the effective number of parameters in a model and it is not defined for singular models. WAIC is fully Bayesian and closely approximates Bayesian cross-validation. Unlike DIC, WAIC is invariant to parametrization and also works for singular models. WAIC is asymptotically equal to LOO, and can thus be used as an approximation to LOO. In the finite case, WAIC and LOO often give very similar estimates, but for influential observations WAIC underestimates the effect of leaving out one observation.

One advantage of AIC and DIC has been their computational simplicity. In this package we present fast and stable computations for LOO and WAIC that can be performed directly on posterior simulations, thus allowing these newer tools to enter routine statistical practice. As a byproduct of our calculations, we also obtain approximate standard errors for estimated predictive errors and for the comparison of predictive errors between two models.

PSIS-LOO

The distribution of the importance weights used in LOO may have a long right tail. We use the empirical Bayes estimate of Zhang and Stephens (2009) to fit a generalized Pareto distribution (gPd) to the tail (20% largest importance ratios). By examining the shape parameter k of the fitted gPd, we are able to obtain sample based estimates of the existance of the moments (Koopman et al, 2009). This extends the diagnostic approach of Peruggia (1997) and Epifani et al. (2008) to be used routinely with importance-sampling LOO for any model with a factorizing likelihood.

Epifani et al. (2008) show that when estimating the leave-one-out predictive density, the central limit theorem holds if the variance of the weight distribution is finite. These results can be extended using the generalized central limit theorem for stable distributions. Thus, even if the variance of the importance weight distribution is infinite, if the mean exists the estimate's accuracy improves when additional draws are obtained. When the tail of the weight distribution is long, a direct use of importance sampling is sensitive to one or few largest values. By fitting a gPd to the upper tail of the importance weights we smooth these values. The procedure goes as follows:

1. Fit the gPd to the 20% largest importance ratios r_s. The computation is done separately for each held-out data point i. In simulation experiments with thousands and tens of thousands of draws, we have found that the fit is not sensitive to the specific cutoff value (for a consistent estimation the proportion of the samples above the cutoff should get smaller when the number of draws increases).
2. Stabilize the importance ratios by replacing the M largest ratios by the expected values of the order statistics of the fitted gPd

$$G((z-0.5)/M), z = 1, ..., M,$$

where M is the number of simulation draws used to fit the Pareto (in this case, $M = 0.2 * S$) and G is the inverse-CDF of the gPd.

3. To guarantee finite variance of the estimate, truncate the smoothed ratios with

$$S^{3/4}\bar{w},$$

where $\bar{w}$ is the average of the smoothed weights.

The above steps must be performed for each data point i. The result is a vector of weights $w_i^s, s = 1, ..., S$, for each i, which in general should be better behaved than the raw importance ratios r_i^s from which they were constructed.

The results are then combined to compute the desired LOO estimates. The reliability of the estimates can be assessed using the estimates for the shape parameter k of the gPd.

- If $k < 1/2$ the variance of the raw importance ratios is finite, the central limit theorem holds, and the estimate converges quickly.
- If k is between 1/2 and 1 the variance of the raw importance ratios is infinite but the mean exists, the generalized central limit theorem for stable distributions holds, and the convergence of the estimate is slower. The variance of the PSIS estimate is finite but may be large.
- If $k > 1$ the variance and the mean of the raw ratios distribution do not exist. The variance of the PSIS estimate is finite but may be large.

If the estimated tail shape parameter $k \geq 1/2$, the user should be warned. Even if the PSIS estimate has a finite variance, the user should consider sampling directly from $p(\theta^s|y_{-i})$ for the problematic i, use k-fold cross-validation, or use a more robust model.

Importance sampling is likely to work less well if the marginal posterior $p(\theta^s|y)$ and LOO posterior $p(\theta^s|y_{-i})$ are much different, which is more likely to happen with a non-robust model and highly influential observations. A robust model may reduce the sensitivity to highly influential observations.

References

Vehtari, A., Gelman, A., and Gabry, J. (2015). Efficient implementation of leave-one-out cross-validation and WAIC for evaluating fitted Bayesian models. http://arxiv.org/abs/1507.04544/ (preprint)

Epifani, I., MacEachern, S. N., and Peruggia, M. (2008). Case-deletion importance sampling estimators: Central limit theorems and related results. *Electronic Journal of Statistics* **2**, 774-806.

Gelfand, A. E. (1996). Model determination using sampling-based methods. In *Markov Chain Monte Carlo in Practice*, ed. W. R. Gilks, S. Richardson, D. J. Spiegelhalter, 145-162. London: Chapman and Hall.

Gelfand, A. E., Dey, D. K., and Chang, H. (1992). Model determination using predictive distributions with implementation via sampling-based methods. In *Bayesian Statistics 4*, ed. J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, 147-167. Oxford University Press.

Gelman, A., Hwang, J., and Vehtari, A. (2014). Understanding predictive information criteria for Bayesian models. *Statistics and Computing* **24**, 997-1016.

Ionides, E. L. (2008). Truncated importance sampling. *Journal of Computational and Graphical Statistics* **17**, 295-311.

Koopman, S. J., Shephard, N., and Creal, D. (2009). Testing the assumptions behind importance sampling. *Journal of Econometrics* **149**, 2-11.

Peruggia, M. (1997). On the variability of case-deletion importance sampling weights in the Bayesian linear model. *Journal of the American Statistical Association* **92**, 199-207.

Stan Development Team (2014a). Stan: A C++ library for probability and sampling, version 2.6. `mc-stan.org`.

Stan Development Team (2014b). RStan, version 2.6. `mc-stan.org/rstan.html`.

Vehtari, A., and Gelman, A. (2015). Pareto smoothed importance sampling. arXiv:1507.02646.

Watanabe, S. (2010). Asymptotic equivalence of Bayes cross validation and widely application information criterion in singular learning theory. *Journal of Machine Learning Research* **11**, 3571-3594.

Zhang, J., and Stephens, M. A. (2009). A new and efficient estimation method for the generalized Pareto distribution. *Technometrics* **51**, 316-325.

`compare` *Model comparison*

Description

Compare fitted models on LOO or WAIC

Usage

```
compare(...)
```

Arguments

`...`	At least two objects returned by `loo` or `waic`.

Details

When comparing two fitted models, we can estimate the difference in their expected predictive accuracy by the difference in `elpd_waic` or `elpd_loo` (multiplied by -2, if desired, to be on the deviance scale). To compute the standard error of this difference we can use a paired estimate to take advantage of the fact that the same set of N data points was used to fit both models. We think these calculations will be most useful when N is large, because then non-normality of the distribution is not such an issue when estimating the uncertainty in these sums. These standard errors, for all their flaws, should give a better sense of uncertainty than what is obtained using the current standard approach of comparing differences of deviances to a Chi-squared distribution, a practice derived for Gaussian linear models or asymptotically, and which only applies to nested models in any case.

Value

A vector or matrix with class 'compare.loo'. If ... contains more than two objects then a matrix is returned. This matrix summarizes the objects and also reports model weights (the posterior probability that each model has the best expected out-of-sample predictive accuracy). If ... contains exactly two objects then the difference in expected predictive accuracy and the standard error of the difference are returned (see Details) in addition to model weights.

References

Vehtari, A., Gelman, A., and Gabry, J. (2015). Efficient implementation of leave-one-out cross-validation and WAIC for evaluating fitted Bayesian models. http://arxiv.org/abs/1507.04544/ (preprint)

See Also

print.compare.loo

Examples

```
## Not run:
loo1 <- loo(log_lik1)
loo2 <- loo(log_lik2)
print(compare(loo1, loo2), digits = 3)

waic1 <- waic(log_lik1)
waic2 <- waic(log_lik2)
compare(waic1, waic2)

## End(Not run)
```

---

extract_log_lik *Extract log-likelihood from a Stan model*

Description

Convenience function for extracting the pointwise log-likelihood from a fitted Stan model.

Usage

```
extract_log_lik(stanfit, parameter_name = "log_lik")
```

Arguments

stanfit	A stanfit object (**rstan** package).
parameter_name	A character string naming the parameter (or generated quantity) in the Stan model corresponding to the log-likelihood.

Details

Stan does not automatically compute and store the log-likelihood. It is up to the user to incorporate it into the Stan program if it is to be extracted after fitting the model. In a Stan model, the pointwise log likelihood can be coded as a vector in the transformed parameters block (and then summed up in the model block) or it can be coded entirely in the generated quantities block. We recommend using the generated quantities block so that the computations are carried out only once per iteration rather than once per HMC leapfrog step.

For example, the following is the `generated quantities` block for computing and saving the log-likelihood for a linear regression model with `N` data points, outcome `y`, predictor matrix `X`, coefficients `beta`, and standard deviation `sigma`:

```
vector[N] log_lik;
```

```
for (n in 1:N) log_lik[n] <- normal_log(y[n], X[n] * beta, sigma);
```

Value

An S by N matrix of (post-warmup) extracted draws, where S is the size of the posterior sample and N is the number of data points.

References

Stan Development Team (2015). Stan: A C++ library for probability and sampling, version 2.6. `mc-stan.org`.

Stan Development Team (2015). RStan, version 2.6. `mc-stan.org/rstan.html`.

`loo` *Leave-one-out cross-validation (LOO)*

Description

Efficient approximate leave-one-out cross-validation for Bayesian models.

Usage

```
loo(x, ...)

## S3 method for class 'matrix'
loo(x, ...)

## S3 method for class 'function'
loo(x, ..., args)
```

Arguments

`x`	A log-likelihood matrix or function. See the **Methods (by class)** section below for a detailed description.
`...`	Optional arguments to pass to `psislw`. Possible arguments and their defaults are: `wcp = 0.2` The proportion of importance weights to use for the generalized Pareto fit. The `100*wcp`% largest weights are used as the sample from which to estimate the parameters k and σ of the generalized Pareto distribution. `wtrunc = 3/4` For truncating very large weights to S^`wtrunc` (set to zero for no truncation). `cores = getOption("loo.cores", parallel::detectCores())` The number of cores to use for parallelization. This can be set for an entire R session by `options(loo.cores = NUMBER)`. The default is `detectCores`(). We recommend using the default values for the `psislw` arguments unless there are problems (e.g. `NA` or `NaN` results).
`args`	Only required if `x` is a function. A list containing the data required to specify the arguments to the function. See the **Methods (by class)** section below for how `args` should be specified.

Value

A named list with class `'loo'` and components:

`elpd_loo, se_elpd_loo` Expected log pointwise predictive density and standard error.

`p_loo, se_p_loo` Estimated effective number of parameters and standard error.

`looic, se_looic` The LOO information criterion (`-2*elpd_loo`, i.e., converted to deviance scale) and standard error.

`pointwise` A matrix containing the pointwise contributions of each of the above measures.

`pareto_k` A vector containing the estimates of the shape parameter k for the generaelized Pareto fit to the importance ratios for each leave-one-out distribution. See PSIS-LOO section in `loo-package` for details about interpreting k. (By default, the `print` method for `'loo'` objects will also provide warnings about problematic values of k.)

Methods (by class)

- `matrix`: An S by N matrix, where S is the size of the posterior sample (the number of simulations) and N is the number of data points. Typically (but not restricted to be) the object returned by `extract_log_lik`.
- `function`: A function f that takes arguments `i`, `data`, and `draws` and returns a vector containing the log-likelihood for the `i`th observation evaluated at each posterior draw.

  The `args` argument must also be specified and should be a named list with the following components:

  - `draws`: An object containing the posterior draws for any parameters needed to compute the pointwise log-likelihood.

- data: An object containing data (e.g. observed outcome and predictors) needed to compute the pointwise log-likelihood. data should be in an appropriate form so that *f*(i=i, data=data[i,,drop=FALSE], draws=draws) returns the S-vector of log-likelihoods for the ith observation.
- N: The number of observations.
- S: The size of the posterior sample.

See Also

loo-package, print.loo, compare

Examples

```
## Not run:
### Usage with stanfit objects
log_lik1 <- extract_log_lik(stanfit1) # see ?extract_log_lik
loo1 <- loo(log_lik1)
print(loo1, digits = 3)

log_lik2 <- extract_log_lik(stanfit2)
(loo2 <- loo(log_lik2))
compare(loo1, loo2)

## End(Not run)

### Using log-likelihood function instead of matrix
set.seed(024)

# Simulate data and draw from posterior
N <- 50; K <- 10; S <- 100; a0 <- 3; b0 <- 2
p <- rbeta(1, a0, b0)
y <- rbinom(N, size = K, prob = p)
a <- a0 + sum(y); b <- b0 + N * K - sum(y)
draws <- rbeta(S, a, b)
data <- data.frame(y,K)

llfun <- function(i, data, draws) {
  dbinom(data$y, size = data$K, prob = draws, log = TRUE)
}
loo_with_fn <- loo(llfun, args = nlist(data, draws, N, S), cores = 1)

# Check that we get same answer if using log-likelihood matrix
log_lik_mat <- sapply(1:N, function(i) llfun(i, data[i,, drop=FALSE], draws))
loo_with_mat <- loo(log_lik_mat, cores = 1)
all.equal(loo_with_mat, loo_with_fn)
```

print.loo *Print and plot methods*

Description

print and plot methods for objects of class 'loo' and print method for objects of class 'compare.loo'.

Usage

```
## S3 method for class 'loo'
print(x, ..., digits = 1, warn = TRUE, plot_k = FALSE)

## S3 method for class 'compare.loo'
print(x, ..., digits = 1)

## S3 method for class 'loo'
plot(x, ..., label_points = FALSE)
```

Arguments

x	A list with class 'loo' (as returned by the loo function or, for print only, the waic function). For print, x can also have class 'compare.loo' (as returned by compare).
...	For plot, arguments to pass to text if label_points = TRUE.
digits	An integer passed to round.
warn	Logical. If TRUE (the default), a warning message will be printed if any estimates of the Pareto shape parameter k are problematic. See the PSIS-LOO section in loo-package for details on the interpretation of k. Ignored if x was generated by waic.
plot_k	Logical. If TRUE the estimates of the Pareto shape parameter k are plotted. Ignored if x was generated by waic. To just plot k without printing use plot(x).
label_points	Logical. If TRUE the observation numbers corresponding to any values of k greater than 0.5 will be displayed in the plot. Any arguments specified in ... will be passed to text and can be used to control the appearance of the labels.

Value

The print methods return x invisibly. The plot method is called for its side effect and does not return anything. If x is the result of a call to loo, plot(x) produces a plot of the estimates of the Pareto shape parameter k. There is no plot method for objects generated by a call to waic.

psislw *Pareto smoothed importance sampling (PSIS)*

Description

Pareto smoothed importance sampling (PSIS)

Usage

```
psislw(lw, wcp = 0.2, wtrunc = 3/4, cores = getOption("loo.cores",
  parallel::detectCores()), llfun = NULL, llargs = NULL, ...)
```

Arguments

`lw`	A matrix or vector of log weights. For computing LOO, `lw = -log_lik` (see `extract_log_lik`) and is an S by N matrix where S is the number of simulations and N is the number of data points. (If `lw` is a vector it will be coerced to a one-column matrix.)
`wcp`	The proportion of importance weights to use for the generalized Pareto fit. The `100*wcp`% largest weights are used as the sample from which to estimate the parameters of the generalized Pareto distribution.
`wtrunc`	For truncating very large weights to S^`wtrunc`. Set to zero for no truncation.
`cores`	The number of cores to use for parallelization. This can be set for an entire R session by `options(loo.cores = NUMBER)`. The default is `detectCores`().
`llfun, llargs`	See `loo.function`.
`...`	Ignored when `psislw` is called directly. The `...` is only used internally when `psislw` is called by the `loo` function.

Details

See the 'PSIS-LOO' section in `loo-package`.

Value

A named list with components `lw_smooth` (modified log weights) and `pareto_k` (estimated generalized Pareto shape parameter(s) k).

Note

This function is primarily intended for internal use, but is exported so that users can call it directly if interested in PSIS for purposes other than `LOO-CV`.

References

Vehtari, A., Gelman, A., and Gabry, J. (2015). Efficient implementation of leave-one-out cross-validation and WAIC for evaluating fitted Bayesian models. `http://arxiv.org/abs/1507.04544/` (preprint)

waic *Widely applicable information criterion (WAIC)*

Description

Widely applicable information criterion (WAIC)

Usage

```
waic(x, ...)

## S3 method for class 'matrix'
waic(x, ...)

## S3 method for class 'function'
waic(x, ..., args)
```

Arguments

`x`	A log-likelihood matrix or function. See the **Methods (by class)** section below for a detailed description.
`...`	Other arguments. Currently ignored.
`args`	Only required if `x` is a function. A list containing the data required to specify the arguments to the function. See the **Methods (by class)** section below for how `args` should be specified.

Value

A named list (of class `'loo'`) with components:

`elpd_waic, se_elpd_waic`	expected log pointwise predictive density and standard error
`p_waic, se_p_waic`	estimated effective number of parameters and standard error
`waic, se_waic`	`-2 * elpd_waic` (i.e., converted to the deviance scale) and standard error
`pointwise`	the pointwise contributions of each of the above measures

Methods (by class)

- `matrix`: An S by N matrix, where S is the size of the posterior sample (the number of simulations) and N is the number of data points. Typically (but not restricted to be) the object returned by `extract_log_lik`.
- `function`: A function f that takes arguments `i`, `data`, and `draws` and returns a vector containing the log-likelihood for the `i`th observation evaluated at each posterior draw.

  The `args` argument must also be specified and should be a named list with the following components:

  – `draws`: An object containing the posterior draws for any parameters needed to compute the pointwise log-likelihood.

- `data`: An object containing data (e.g. observed outcome and predictors) needed to compute the pointwise log-likelihood. `data` should be in an appropriate form so that f`(i=i, data=data[i,,drop=FALSE], draws=draws)` returns the `S`-vector of log-likelihoods for the `i`th observation.
- `N`: The number of observations.
- `S`: The size of the posterior sample.

See Also

`compare`, `print.loo`, `loo-package`

Examples

```
## Not run:
log_lik1 <- extract_log_lik(stanfit1)
log_lik2 <- extract_log_lik(stanfit2)
(waic1 <- waic(log_lik1))
(waic2 <- waic(log_lik2))
print(compare(waic1, waic2), digits = 2)

## End(Not run)
```

Index

compare, 5, *9*, *10*, *13*

detectCores, *8*, *11*

extract_log_lik, 6, *8*, *11*, *12*

loo, *5*, 7, *10*, *11*
loo-package, 2
loo.function, *11*

plot.loo *(*print.loo*)*, 10
print, *8*
print.compare.loo, *6*
print.compare.loo *(*print.loo*)*, 10
print.loo, *9*, 10, *13*
psislw, *8*, 11

round, *10*

text, *10*

waic, *5*, *10*, 12