Package 'fangs'

July 17, 2023

Title Feature Allocation Neighborhood Greedy Search Algorithm

Version 0.2.13

Description A neighborhood-based, greedy search algorithm is performed to estimate a feature allocation by minimizing the expected loss based on posterior samples from the feature allocation distribution. The method is currently under peer review but an earlier draft is available in Dahl, Johnson, and Andros (2022+) <doi:10.48550/arXiv.2207.13824>.

License MIT + file LICENSE | Apache License 2.0

URL https://github.com/dbdahl/fangs

BugReports https://github.com/dbdahl/fangs/issues

Depends R (>= 4.2.0)

SystemRequirements Cargo (Rust's package manager), rustc (>= 1.65)

Encoding UTF-8

LazyData TRUE

RoxygenNote 7.2.3

NeedsCompilation yes

Author David B. Dahl [aut, cre] (<https://orcid.org/0000-0002-8173-1547>),
R. Jacob Andros [aut] (<https://orcid.org/0000-0002-1289-385X>),
Devin J. Johnson [aut] (<https://orcid.org/0000-0003-2619-6649>),
Alex Crichton [cph] (Rust crates: proc-macro2, cfg-if.),
Andrii Dmytrenko [cph] (Rust crate: lapjv.),
Brendan Zabarauskas [cph] (Rust crate: approx.),
David B. Dahl [cph] (Rust crates: roxido, roxido_macro.),
David Tolnay [cph] (Rust crates: syn, proc-macro2, quote, unicode-ident.),
DutchGhost [cph] (Rust crate: matrixmultiply.),
Enthought, Inc. [cph] (Rust crate: ndarray.),
Gilad Naaman [cph] (Rust crate: memoffset.),
Jim Turner [cph] (Rust crate: ndarray.),
Josh Stone [cph] (Rust crates: rayon, autocfg, rayon-core.),
matrixmultiply authors [cph] (Rust crate: matrixmultiply.),
Melissa O'Neill [cph] (Rust crate: rand_pcg.),
ndarray developers [cph] (Rust crate: ndarray.),

Niko Matsakis [cph] (Rust crates: rayon-core, rayon.),
Paul Dicker [cph] (Rust crate: rand_pcg.),
PCG Project contributors [cph] (Rust crate: rand_pcg.),
Ralf Jung [cph] (Rust crate: memoffset.),
rawpointer developers [cph] (Rust crate: rawpointer.),
R. Janis Goldschmidt [cph] (Rust crate: matrixmultiply.),
SciPy Developers [cph] (Rust crate: ndarray.),
scopeguard developers [cph] (Rust crate: scopeguard.),
Sean McArthur [cph] (Rust crate: num_cpus.),
Stefan Lankes [cph] (Rust crate: hermit-abi.),
The Cranelift Project Developers [cph] (Rust crate: wasi.),
The Crossbeam Project Developers [cph] (Rust crates: crossbeam-utils, crossbeam-epoch, crossbeam-deque, crossbeam-channel.),
The CryptoCorrosion Contributors [cph] (Rust crates: ppv-lite86, rand_chacha.),
The Go Authors [cph] (Rust crate: crossbeam-channel.),
The Rand Project Developers [cph] (Rust crates: rand_chacha, getrandom, rand_core, rand, rand_pcg.),
The Rust Project Developers [cph] (Rust crates: quote, rayon, crossbeam-channel, rand, num-traits, getrandom, libc, rand_chacha, num-complex, num-integer, rand_core, log.),
Ulrik Sverdrup ``bluss'' [cph] (Rust crate: scopeguard, matrixmultiply, either, rawpointer, itertools, ndarray.),
Unicode, Inc. [cph] (Rust crate: unicode-ident.)

Maintainer David B. Dahl <dahl@stat.byu.edu>

Repository CRAN

Date/Publication 2023-07-17 19:40:02 UTC

R topics documented:

`compute_expected_loss` *Estimate the expected FARO Loss for a Feature Allocation*

Description

A Monte Carlo estimate of the expected FARO loss is computed for a feature allocation given a set of posterior samples.

Usage

```
compute_expected_loss(samples, Z, a = 1, nCores = 0)
```

Arguments

samples	An object of class 'list' containing posterior samples from a feature allocation distribution. Each list element encodes one feature allocation as a binary matrix, with items in the rows and features in the columns.
Z	A feature allocation in binary matrix form, with items in the rows and features in the columns.
a	A numeric scalar for the cost parameter of generalized Hamming distance used in FARO loss. The other cost parameter, b, is equal to $2 - a$.
nCores	The number of CPU cores to use, i.e., the number of simultaneous calculations at any given time. A value of zero indicates to use all cores on the system.

Value

The estimated expected FARO loss as a scalar value.

References

D. B. Dahl, D. J. Johnson, R. J. Andros (2023+), Comparison and Bayesian Estimation of Feature Allocations, Journal of Computational and Graphical Statistics, doi:10.1080/10618600.2023.2204136.

Examples

```
data(samplesFA)
Z <- matrix(sample(c(0,1), 60, replace=TRUE), byrow=TRUE, nrow=20)
compute_expected_loss(samplesFA, Z)
```

---

compute_loss *Compute the FARO Loss Between Feature Allocations*

Description

The FARO loss is computed between two feature allocations, each represented in binary matrix form.

Usage

```
compute_loss(Z1, Z2, a = 1, augmented = FALSE)
```

Arguments

Z1	A feature allocation in binary matrix form, with items in the rows and features in the columns.
Z2	A feature allocation in binary matrix form, with items in the rows and features in the columns.
a	A numeric scalar for the cost parameter of generalized Hamming distance used in FARO loss. The other cost parameter, b, is equal to $2 - a$.
augmented	If TRUE, the column permutation (used by FARO loss to compare the feature allocations) is returned for each matrix.

Value

The FARO loss as a scalar value if augmented = FALSE, otherwise, a list of 3 elements including the loss and the two column permutations.

References

D. B. Dahl, D. J. Johnson, R. J. Andros (2023+), Comparison and Bayesian Estimation of Feature Allocations, Journal of Computational and Graphical Statistics, doi:10.1080/10618600.2023.2204136.

Examples

```
Z1 <- matrix(c(0,1,1,0,1,1,0,1,1,1,1,1), byrow=TRUE, nrow=6)
Z2 <- matrix(c(0,0,1,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0), byrow=TRUE, nrow=6)
compute_loss(Z1,Z2)
x <- compute_loss(Z1,Z2,a=1,TRUE)
sum(cbind(Z1,0) != Z2)
sum(cbind(Z1,0)[,x$permutation1] != Z2)
sum(cbind(Z1,0) != Z2[,x$permutation2])
```

---

fangs *Feature Allocation Neighborhood Greedy Search*

Description

An implementation of the feature allocation greedy search algorithm is provided.

Usage

```
fangs(
  samples,
  nInit = 16,
  nSweet = 4,
  nIterations = 0,
  maxSeconds = 60,
  a = 1,
```

```
  nCores = 0,
  algorithm = "stochastic",
  quiet = FALSE
)
```

Arguments

samples	An object of class 'list' containing posterior samples from a feature allocation distribution. Each list element encodes one feature allocation as a binary matrix, with items in the rows and features in the columns.
nInit	The number of initial feature allocations to obtain using the alignment method. For each initial feature, a baseline feature allocation is uniformly selected from the list provided in `samples`. Samples are aligned to the baseline, proportions are computed for each matrix element, and the initial feature allocation is obtained by thresholding according to $a/2$.
nSweet	The number of feature allocations among `nInit` which are chosen (by lowest expected loss) to be optimized in the sweetening phase.
nIterations	The number of iterations (i.e., proposed changes) to consider per initial estimate in the stochastic sweetening phase, although the actual number may be less due to the `maxSeconds` argument. The default value is `0`, which sets the number of iterations to the number of items times the number of columns.
maxSeconds	Stop the search and return the current best estimate once the elapsed time exceeds this value.
a	A numeric scalar for the cost parameter of generalized Hamming distance used in FARO loss. The other cost parameter, b, is equal to $2 - a$.
nCores	The number of CPU cores to use, i.e., the number of simultaneous calculations at any given time. A value of zero indicates to use all cores on the system.
algorithm	A string indicating the algorithm to use; equal to "stochastic", "deterministic", or "draws". The "stochastic" algorithm is recommended, although the "deterministic" algorithm may provide an improvement at the cost of time.
quiet	If `TRUE`, intermediate status reporting is suppressed. Otherwise details are provided, especially when `algorithm="stochastic"`.

Value

A list with the following elements:

- estimate - The feature allocation point estimate in binary matrix form.
- expectedLoss - The estimated expected FARO loss of the point estimate.
- iteration - The iteration number (out of `nIterations`) at which the point estimate was found while sweetening.
- nIterations - The number of sweetening iterations performed.
- secondsInitialization - The elapsed time in the initialization phrase.
- secondsSweetening - The elapsed time in the sweetening phrase.
- secondsTotal - The total elapsed time.

- whichSweet - The proposal number (out of `nSweet`) from which the point estimate was found.
- nInit - The original supplied value of `nInit`.
- nSweet - The original supplied value of `nSweet`.
- a - The original supplied value of `a`.

References

D. B. Dahl, D. J. Johnson, R. J. Andros (2023+), Comparison and Bayesian Estimation of Feature Allocations, Journal of Computational and Graphical Statistics, doi:10.1080/10618600.2023.2204136.

Examples

```
# To reduce load on CRAN testing servers, limit the number of iterations.
data(samplesFA)
fangs(samplesFA, nIterations=100, nCores=2)
```

`samplesFA` *Samples from the Attraction Indian Buffet Distribution*

Description

Samples are provided from a latent feature allocation model using the Attraction Indian Buffet Distribution (Warr et al., 2022) as a prior distribution. The purpose of the model was to use pairwise distance information to identify and predict the presence of Alzheimer's disease in patients.

Usage

```
data(samplesFA)
```

Format

An object of class 'list' containing 100 posterior samples from Warr et al. (2022)'s analysis. Each list elements encodes one feature allocation as a binary matrix, with items in the rows and features in the columns. These 100 feature allocation samples are a subset of the original 1000 samples obtained using MCMC in the original simulation study described by Warr et al. (2022).

References

R. L. Warr, D. B. Dahl, J. M. Meyer, A. Lui (2022), The Attraction Indian Buffet Distribution, Bayesian Analysis, 17 (3), 931-967, doi:10.1214/21BA1279.

Index