Package 'aibd'

March 28, 2020

Type Package

Title Attraction Indian Buffet Distribution

Version 0.1.8

Description An implementation of probability mass function and sampling algorithms is provided for the attraction Indian buffet distribution (AIBD), originally from Dahl (2016) <https://ww2.amstat.org/meetings/jsm/2016/onlineprogram/ActivityDetails.cfm?SessionID=21303

License GPL-3

Encoding UTF-8

LazyData true

Imports utils, rscala (>= 3.2.18), commonsMath (>= 1.2.5)

RoxygenNote 7.1.0

Suggests testthat

NeedsCompilation no

Author David B. Dahl [aut, cre],
Richard Warr [aut],
Jeremy Meyer [aut]

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

Repository CRAN

Date/Publication 2020-03-28 06:20:03 UTC

R topics documented:

`aibd` *Define an Attraction Indian Buffet Distribution (AIBD) for Feature Allocations*

Description

This function specifies an Attraction Indian Buffet Distribution (AIBD), which is a distribution over feature allocations.

Usage

```
aibd(
  mass,
  permutation,
  temperature,
  distance,
  decayFunction = c("exponential", "reciprocal", "identity")[1]
)
```

Arguments

`mass`	The mass (a.k.a., concentration) parameter of the AIBD.
`permutation`	A permutation, i.e., a vector of integers `1,2,...,n` whose length is `n` and whose elements are unique. Using the Indian buffet analogy, the permutation represents the order the customers enter the buffet.
`temperature`	A nonnegative scalar which determines how influential the distance matrix is in the feature allocation distribution. The AIBD reduces to the IBP when the temperature is zero and diverges from the IBP as the temperature increases.
`distance`	A distance matrix, i.e., a symmetric matrix whose `(i,j)` entry is small if items `i` and `j` are similar. An object of class `"dist"` is also permissible.
`decayFunction`	One of the following strings: `"exponential"` (making `similarity = exp(-temperature*distance)`), `"reciprocal"` (making `similarity = 1/distance^temperature`), or `"identity"` (in which case `distance` is interpreted as a similarity instead of a distance).

Value

An object representing an Attraction Indian Buffet Distribution (AIBD) for feature allocations.

Examples

```
states <- c("California","Wisconsin","Nebraska","New York")
data <- USArrests[states,]
dist <- dist(scale(data))
aibd(1, seq_along(states), 1.0, dist)
```

`ibp` *Define an Indian Buffet Process (IBP) Distribution for Feature Allocations*

Description

This function specifies an Indian Buffet Process (IBP), which is a distribution over feature allocations.

Usage

```
ibp(mass, x)
```

Arguments

`mass`	The mass (a.k.a., concentration) parameter.
`x`	A character vector giving the labels of the items to place in a feature allocation, or an integer giving the number of items.

Value

An object representing an Indian Buffet Process (IBP) feature allocation distribution.

Examples

```
ibp(1,5)
ibp(1,c("CA","WI","NE","NY","UT"))
```

`logLikelihoodLGLFM` *Log of the Likelihood from the Linear Gaussian Latent Feature Model*

Description

The log of the likelihood of a feature allocation from the linear Gaussian latent feature model (LGLFM) is computed. The standard deviation of the error term (`sdX`) may be supplied or the associated precision (`precisionX`) can be provided instead. Likewise, only one of `sdA` and `precisionA` should be supplied.

Usage

```
logLikelihoodLGLFM(
  featureAllocation,
  X,
  precisionX,
  precisionA,
  sdX,
  sdA,
  implementation = "scala"
)
```

Arguments

featureAllocation	An N-by-K binary feature allocation matrix.
X	An N-by-D matrix of observed data.
precisionX	The scalar precision of the data error variance. This must be specified if sdX is missing.
precisionA	The scalar precision of a latent feature. This must be specified if sdA is missing.
sdX	The scalar standard deviation of the data error variance. This must be specified if precisionX is missing.
sdA	The scalar precision of a latent feature. This must be specified if precisionA is missing.
implementation	The default of "scala" should be used. The "R" option is not a supported implementation.

Value

A numeric vector giving the log of the likelihood.

See Also

This function is an implementation of the log of Equation (26) in "The Indian Buffet Process: An Introduction and Review" by Griffiths and Ghahramani (2011) in the Journal of Machine Learning.

Examples

```
 # Regardless of size, the initial warmup can exceed CRAN's 5 seconds threshold
sigx <- 0.1
siga <- 1.0
dimA <- 1
nItems <- 8  # Should be a multiple of 4
Z <- matrix(c(1,0,1,1,0,1,0,0),byrow=TRUE,nrow=nItems,ncol=2)
A <- matrix(rnorm(ncol(Z)*dimA,sd=siga),nrow=ncol(Z),ncol=dimA)
e <- rnorm(nrow(Z)*ncol(A),0,sd=sigx)
X <- Z %*% A + e
logLikelihoodLGLFM(Z, X, sdX=sigx, sdA=siga)
```

`logPosteriorLGLFM` *Log of the Posterior Density from the Linear Gaussian Latent Feature Model*

Description

The log of the unnormalized posterior density of a feature allocation from the linear Gaussian latent feature model (LGLFM) is computed. The standard deviation of the error term (`sdX`) may be supplied or the associated precision (`precisionX`) can be provided instead. Likewise, only one of `sdA` and `precisionA` should be supplied.

Usage

```
logPosteriorLGLFM(
  featureAllocation,
  distribution,
  X,
  precisionX,
  precisionA,
  sdX,
  sdA,
  implementation = "scala"
)
```

Arguments

`featureAllocation`	An N-by-K binary feature allocation matrix.
`distribution`	A prior distribution of feature allocations, i.e., a result from `ibp` or `aibd`.
`X`	An N-by-D matrix of observed data.
`precisionX`	The scalar precision of the data error variance. This must be specified if `sdX` is missing.
`precisionA`	The scalar precision of a latent feature. This must be specified if `sdA` is missing.
`sdX`	The scalar standard deviation of the data error variance. This must be specified if `precisionX` is missing.
`sdA`	The scalar precision of a latent feature. This must be specified if `precisionA` is missing.
`implementation`	The default of `"scala"` should be used. The `"R"` option is not a supported implementation.

Value

A numeric vector giving the log of the unnormalized posterior density.

Examples

```
 # Regardless of size, the initial warmup can exceed CRAN's 5 seconds threshold
sigx <- 0.1
siga <- 1.0
dimA <- 1
nItems <- 8  # Should be a multiple of 4
Z <- matrix(c(1,0,1,1,0,1,0,0),byrow=TRUE,nrow=nItems,ncol=2)
A <- matrix(rnorm(ncol(Z)*dimA,sd=siga),nrow=ncol(Z),ncol=dimA)
e <- rnorm(nrow(Z)*ncol(A),0,sd=sigx)
X <- Z %*% A + e
logLikelihoodLGLFM(Z, X, sdX=sigx, sdA=siga)
logPosteriorLGLFM(Z, ibp(1,nItems), X, sdX=sigx, sdA=siga)
```

---

logProbabilityFeatureAllocation *Evaluation of a Log Probabilty Mass Function of a Feature Allocation Distribution*

Description

This function evaluates the log of the probability mass function of a feature allocation matrix or a list of feature allocations for the supplied distribution.

Usage

```
logProbabilityFeatureAllocation(
  featureAllocation,
  distribution,
  implementation = "scala"
)
```

Arguments

`featureAllocation`	An N-by-K binary feature allocation matrix, or a list of such matrices.
`distribution`	A feature allocation distribution as defined in the functions `aibd` or `ibp`.
`implementation`	The default of "scala" should be used. The "R" option is not a supported implementation.

Value

The log probability of the feature allocation under the supplied distribution.

Examples

```
 # Regardless of size, the initial warmup can exceed CRAN's 5 seconds threshold
d1 <- ibp(1,4)

states <- c("California","Wisconsin","Nebraska","New York")
data <- USArrests[states,]
dist <- dist(scale(data))
d2 <- aibd(1, seq_along(states), 1.0, dist)

Z1 <- matrix(c(1,1,0,1), nrow=4)

logProbabilityFeatureAllocation(Z1, d1)
logProbabilityFeatureAllocation(Z1, d2)
```

---

sampleFeatureAllocation — *Sample from a Feature Allocation Distribution*

Description

This function obtains a sample from a previously defined feature allocation distribution object using wither the `ibp` or the `aibd` functions.

Usage

```
sampleFeatureAllocation(
  nSamples,
  distribution,
  implementation = "scala",
  parallel = TRUE
)
```

Arguments

`nSamples`	An integer giving the number of samples
`distribution`	A feature allocation distribution object as defined in the functions `aibd` or `ibp`.
`implementation`	The default of "scala" should be used. The "R" option is not a supported implementation.
`parallel`	Whether multiple cores should be used to generate the samples.

Value

A list of feature allocation matrices sampled from the supplied distribution.

Examples

```
 # Regardless of size, the initial warmup can exceed CRAN's 5 seconds threshold
d1 <- ibp(1,4)

states <- c("California","Wisconsin","Nebraska","New York")
data <- USArrests[states,]
dist <- dist(scale(data))
d2 <- aibd(1, seq_along(states), 1.0, dist)

samples_ibp <- sampleFeatureAllocation(10, d1, parallel=FALSE)
samples_aibd <- sampleFeatureAllocation(15, d2, parallel=FALSE)
```

`samplePosteriorLGLFM` *Sample from the Posterior Distribution of the Linear Gaussian Feature Allocation Model*

Description

This function samples from the posterior distribution of the linear Gaussian latent feature model (LGLFM) using an Indian buffet process (IBP) or an Attraction Indian Buffet Distribution (AIBD) prior over possible feature allocations.

Usage

```
samplePosteriorLGLFM(
  featureAllocation,
  distribution,
  X,
  precisionX,
  precisionA,
  sdX = 1/sqrt(precisionX),
  sdA = 1/sqrt(precisionA),
  massPriorShape = -1,
  massPriorRate = -1,
  nPerShuffle = 0L,
  temperaturePriorShape = -1,
  temperaturePriorRate = -1,
  maxStandardDeviationX = sd(X),
  maxStandardDeviationA = maxStandardDeviationX,
  sdProposedTemperature = -1,
  sdProposedStandardDeviationX = -1,
  sdProposedStandardDeviationA = -1,
  corProposedSdXSdA = 0,
  newFeaturesTruncationDivisor = 1000,
  nOtherUpdatesPerAllocationUpdate = 10L,
  nSamples = 1L,
```

```
  thin = 1L,
  rankOneUpdates = FALSE,
  verbose = TRUE
)
```

Arguments

featureAllocation
: An N-by-K binary feature allocation matrix.

distribution
: A prior distribution of feature allocations, i.e., a result from `ibp` or `aibd`.

X
: An N-by-D matrix of observed data.

precisionX
: The scalar precision of the data error variance. This must be specified if `sdX` is missing.

precisionA
: The scalar precision of a latent feature. This must be specified if `sdA` is missing.

sdX
: The scalar standard deviation of the data error variance. This must be specified if `precisionX` is missing.

sdA
: The scalar precision of a latent feature. This must be specified if `precisionA` is missing.

massPriorShape
: Shape parameter of the gamma prior on the mass parameter, where the prior expected value is `massPriorShape/massPriorRate`. If either `massPriorShape` or `massPriorRate` is set to `-1`, then the mass parameter is assumed to be fixed (as defined in the `aibd` object).

massPriorRate
: Rate parameter of the gamma prior on the mass parameter, where the expected value if `massPriorShape/massPriorRate`.

nPerShuffle
: Number of items to randomly select and permute when proposing an update to the permutation associated with the attraction Indian buffet distribution (AIBD).

temperaturePriorShape
: Shape parameter of the gamma prior on the temperature parameter, where the prior expected value is `temperaturePriorShape/temperaturePriorRate`. If either `temperaturePriorShape` or `temperaturePriorRate` is set to `-1`, then the temperature parameter is assumed to be fixed (as defined in the `aibd` object).

temperaturePriorRate
: Rate parameter of the gamma prior on the temperature parameter, where the prior expected value is `temperaturePriorShape/temperaturePriorRate`.

maxStandardDeviationX
: Maximum value parameter of the uniform prior distribution on the standard deviation of `X`.

maxStandardDeviationA
: Maximum value parameter of the uniform prior distribution on the standard deviation of `A`.

sdProposedTemperature
: Standard deviation of the Gaussian random walk update for the standard deviation of the temperature.

sdProposedStandardDeviationX
: Standard deviation of the Gaussian random walk update for the standard deviation of `X`.

`sdProposedStandardDeviationA`
: Standard deviation of the Gaussian random walk update for the standard deviation of `A`.

`corProposedSdXSdA`
: Correlation of the multivariate Gaussian random walk updates for the standard deviations of `X` and `A`.

`newFeaturesTruncationDivisor`
: While in theory a countable infinite number of new features may be allocated to an item, the posterior simulation needs to limit the number of new features that are considered. The value of this argument controls when to stop considering additional features. Starting with 0 and 1 new features, the posterior probabilities are computed. Additional new features are considered but the algorithm stops when the posterior probabilities of the current number of new features is less than the maximum posterior probability (among the previous number of new features) divided by `newFeaturesTruncationDivisior`.

`nOtherUpdatesPerAllocationUpdate`
: This parameter controls how many additional MCMC updates occur for all other random model parameters for one update of the `featureAllocation` matrix. Using values of `nOtherUpdatesPerAllocationUpdate` > 1 will presumably improving the mixing of the MCMC with relatively minimal computational cost.

`nSamples`
: Number of feature allocations to return. The actual number of iterations of the algorithm is `thin*nSamples`.

`thin`
: Only save 1 in `thin` feature allocations.

`rankOneUpdates`
: Should rank one updates for the inverse and determinant be used? In some cases, this may be faster.

`verbose`
: Should a progress bar and information regarding lapse time and acceptance rates be displayed?

Examples

```
 # Regardless of size, the initial warmup can exceed CRAN's 5 seconds threshold
mass <- 1
sigx <- 0.1
siga <- 1.0
dimA <- 1
nItems <- 8
dist <- ibp(mass, nItems)
Z <- matrix(c(1,0,1,1,0,1,0,0),byrow=TRUE,nrow=nItems,ncol=2)
A <- matrix(rnorm(ncol(Z)*dimA,sd=siga),nrow=ncol(Z),ncol=dimA)
e <- rnorm(nrow(Z)*ncol(A),0,sd=sigx)
X <- Z %*% A + e
samples <- samplePosteriorLGLFM(Z, dist, X, sdX=sigx, sdA=siga, nSamples=1000, thin=1)
```

Index