Package 'UNCOVER'

July 21, 2025

Type Package

Title Utilising Normalisation Constant Optimisation via Edge Removal (UNCOVER)

Version 1.1.0

Author Samuel Emerson [aut, cre]

Maintainer Samuel Emerson <samuel.emerson@hotmail.co.uk>

Description Model data with a suspected clustering structure (either in co-variate space, regression space or both) using a Bayesian product model with a logistic regression likelihood. Observations are represented graphically and clusters are formed through various edge removals or additions. Cluster quality is assessed through the log Bayesian evidence of the overall model, which is estimated using either a Sequential Monte Carlo sampler or a suitable transformation of the Bayesian Information Criterion as a fast approximation of the former. The internal Iterated Batch Importance Sampling scheme (Chopin (2002 <doi:10.1093/biomet/89.3.539>)) is made available as a free standing function.

License GPL-2

Encoding UTF-8

Imports mvnfast, igraph, crayon, memoise, GGally, ggplot2, ggpubr, scales, stats, cachem, ggnewscale

RoxygenNote 7.2.3

NeedsCompilation no

Repository CRAN

Date/Publication 2023-08-25 13:50:11 UTC

Contents

`IBIS.logreg` *Logistic regression iterated batch importance sampling*

Description

This function uses an Iterated Batch Importance Sampling (IBIS) scheme with batch size one to go from prior to full posterior. We assume a Bayesian logistic regression model.

Usage

```
IBIS.logreg(
  X,
  y,
  options = IBIS.logreg.opts(),
  prior_mean = rep(0, ncol(X) + 1),
  prior_var = diag(ncol(X) + 1)
)
```

Arguments

`X`	Co-variate matrix
`y`	Binary response vector
`options`	Additional arguments that can be specified for `IBIS.logreg`. See `IBIS.logreg.opts()` for details. Can be ignored.
`prior_mean`	Mean for the multivariate normal prior used in the SMC sampler. See details. Defaults to the origin.
`prior_var`	Variance matrix for the multivariate normal prior used in the SMC sampler. See details. Defaults to the identity matrix.

Details

Details of the internal mechanisms of the SMC sampler such as the Metropolis-Hastings MCMC resample move can be found in Emerson and Aslett (2023) and Chopin (2002).

It is never recommended to use anything other than `IBIS.logreg.opts` to provide the `options` argument. See examples and `IBIS.logreg.opts()` for more information.

The prior used for the IBIS procedure will take the form of a multivariate normal, where the parameters can be specified directly by the user. It is however possible to override this default prior distributional form by specifying `prior.override=TRUE` and providing the relevant prior functions in `IBIS.logreg.opts`.

Value

An object of class "IBIS", which is a list consisting of:

`covariate_matrix` The co-variate matrix provided.

`response_vector` The binary response vector provided.

`samples` A matrix of samples from the posterior.

`log_Bayesian_evidence` An estimate of the log Bayesian evidence (or normalisation constant) of the posterior.

`diagnostics` A data frame recording the features of the SMC sampler as the observations were added.

If `weighted==TRUE` then an additional element of the list (`weights`) is added detailing the weights of the posterior samples.

References

- Emerson, S.R. and Aslett, L.J.M. (2023). Joint cohort and prediction modelling through graphical structure analysis (to be released)
- Chopin, N. (2002). A sequential particle filter method for static models. Biometrika, 89(3), 539-552, doi:10.1093/biomet/89.3.539

See Also

`IBIS.logreg.opts()`, `print.IBIS()`, `predict.IBIS()`, `plot.IBIS()`

Examples

```
require(graphics)
# First we generate a co-variate matrix X and binary response vector y
CM <- matrix(rnorm(200),100,2)
rv <- sample(0:1,100,replace=TRUE)

# Now we can obtain 1000 samples from the posterior from a standard
# multivariate normal prior
out.1 <- IBIS.logreg(X = CM,y = rv)
plot(out.1)
out.1$log_Bayesian_evidence

# We can specify that the samples be weighted
out.1.w <- IBIS.logreg(X = CM,y = rv,
                       options = IBIS.logreg.opts(weighted = TRUE))
out.1.w$weights
plot(out.1.w)

# We can also specify different arguments for a specific prior
out.2 <- IBIS.logreg(X = CM,y = rv,prior_mean = rep(-3,3),
                     prior_var = 0.1*diag(3))
samp.df <- data.frame(rbind(out.1$samples,out.2$samples))
```

```
colnames(samp.df) <- paste0("beta[",c(0:2),"]")
GGally::ggpairs(samp.df,
                labeller = "label_parsed",
                ggplot2::aes(color = as.factor(rep(c(1,2),each=1000))),
                upper = list(continuous = GGally::wrap("density")),
                lower = list(continuous = GGally::wrap("points",size=0.5)))
out.2$log_Bayesian_evidence
out.3 <- IBIS.logreg(X = CM,y = rv,prior_mean = rep(3,3),
                     prior_var = 0.1*diag(3))
samp.df <- data.frame(rbind(out.1$samples,out.2$samples,out.3$samples))
colnames(samp.df) <- paste0("beta[",c(0:2),"]")
GGally::ggpairs(samp.df,
                labeller = "label_parsed",
                ggplot2::aes(color = as.factor(rep(c(1,2,3),each=1000))),
                upper = list(continuous = GGally::wrap("density")),
                lower = list(continuous = GGally::wrap("points",size=0.5)))
out.3$log_Bayesian_evidence

# We can also change the prior, for example a multivariate independent
# uniform
rmviu <- function(n,a,b){
return(mapply(FUN = function(min.vec,max.vec,pn){stats::runif(pn,a,b)},
              min.vec=a,max.vec=b,MoreArgs = list(pn = n)))
}
dmviu <- function(x,a,b){
for(ii in 1:ncol(x)){
  x[,ii] <- dunif(x[,ii],a[ii],b[ii])
}
return(apply(x,1,prod))
}

out.4 <- IBIS.logreg(X = CM,y = rv,
                     options = IBIS.logreg.opts(prior.override = TRUE,
                                                rprior = rmviu,
                                                dprior = dmviu,a=rep(0,3),
                                                b=rep(1,3)))
samp.df <- data.frame(rbind(out.1$samples,out.4$samples))
colnames(samp.df) <- paste0("beta[",c(0:2),"]")
GGally::ggpairs(samp.df,
                labeller = "label_parsed",
                ggplot2::aes(color = as.factor(rep(c(1,4),each=1000))),
                upper = list(continuous = GGally::wrap("points",size=0.5)),
                lower = list(continuous = GGally::wrap("points",size=0.5)))
out.4$log_Bayesian_evidence
```

IBIS.logreg.opts *Additional argument generator for* IBIS.logreg()

Description

This function is used to specify additional arguments to `IBIS.logreg`.

Usage

```
IBIS.logreg.opts(
  N = 1000,
  ess = N/2,
  n_move = 1,
  weighted = FALSE,
  prior.override = FALSE,
  rprior = NULL,
  dprior = NULL,
  ...
)
```

Arguments

`N`	Number of particles for the SMC sampler. Defaults to 1000.
`ess`	Effective Sample Size Threshold: If the effective sample size of the particles falls below this value then a resample move step is triggered. Defaults to `N/2`.
`n_move`	Number of Metropolis-Hastings steps to apply each time a resample move step is triggered. Defaults to 1.
`weighted`	Should the outputted samples be weighted? Defaults to `FALSE`.
`prior.override`	Are you overriding the default multivariate normal form of the prior? Defaults to `FALSE`.
`rprior`	Function which produces samples from your prior if the default prior form is to be overridden. If using the default prior form this does not need to be specified.
`dprior`	Function which produces your specified priors density for inputted samples if the default prior form is to be overridden. If using the default prior form this does not need to be specified.
`...`	Additional arguments required for complete specification of the two prior functions given, if the default prior form is to be overridden.

Details

This function should only be used to provide additional control arguments to `IBIS.logreg`.

Specifying `rprior` and `dprior` will not override the default prior form unless `prior.override=TRUE`. If a multivariate normal form is required then the arguments for this prior should be specified in `IBIS.logreg`.

Value

A list consisting of:

`N` Number of particles for the SMC sampler

`ess` Effective Sample Size Threshold

`n_move` Number of Metropolis-Hastings steps

`rprior` Function which produces samples from your prior. `NULL` if `prior.override==FALSE`.

`dprior` Function which produces your specified priors density for inputted samples. `NULL` if `prior.override==FALSE`.

`prior.override` Logical value indicating if the prior has been overridden or not.

`weighted` Logical value indicating if the outputted particles of `IBIS.logreg` should be weighted or not.

`MoreArgs` A list of the additional arguments required for `rprior` and `dprior`. `NULL` if `prior.override==FALSE`.

See Also

`IBIS.logreg()`

Examples

```
#Specifying a multivariate independent uniform prior

rmviu <- function(n,a,b){
return(mapply(FUN = function(min.vec,max.vec,pn){stats::runif(pn,a,b)},
              min.vec=a,max.vec=b,MoreArgs = list(pn = n)))
}
dmviu <- function(x,a,b){
for(ii in 1:ncol(x)){
  x[,ii] <- dunif(x[,ii],a[ii],b[ii])
}
return(apply(x,1,prod))
}

IBIS.logreg.opts(prior.override = TRUE,rprior = rmviu,
                 dprior = dmviu,a=rep(0,3),b=rep(1,3))
```

---

plot.IBIS *Plot various outputs of IBIS*

Description

Allows visualisation of many aspects of IBIS, including co-variate, posterior and diagnostic plots.

Usage

```
## S3 method for class 'IBIS'
plot(x, type = "samples", plot_var = NULL, diagnostic_x_axis = "full", ...)
```

Arguments

x	Object of class "IBIS"
type	Can be one of; "samples" for posterior visualisation, "fitted" for co-variate visualisation or "diagnostics" for diagnostic plots. See details. Defaults to "samples".
plot_var	Vector specifying which columns (or associated logistic regression coefficients) of the co-variate matrix should be plotted. Does not apply when type=="diagnostics". Defaults to all columns being selected.
diagnostic_x_axis	Only applies if "type=="diagnostics". Either "full" (default) for all observations indices to be plotted on the x-axis or "minimal" for only some observations indices to be plotted on the x-axis.
...	Arguments to be passed to methods

Details

If type=="samples", the resulting plot will be a ggpairs plot giving the coefficient densities in the diagonal, points plots of the posterior samples in the lower triangle and contour plots in the upper triangle.

If "type=="fitted", the resulting plot will be a ggpairs plot. The diagonal entries will be two density plots, one for training data predicted to have response 0 by the model (red) and one for training data predicted to have response 1 by the model (green). The off-diagonal elements are scatter-plots of the observations, given a label according to their actual response and a colour scale based on their predicted response. If length(plot_var)==1 then the co-variate variable is plotted against it's index and a density plot is not provided. If length(plot_var)==1 then the density plot and the scatter-plot are combined. If a predicted class (0 or

1. contains less than two data points the density will not be plotted.

If "type==diagnostics", the resulting plot will be a combination of three plots; one tracking the log Bayesian evidence as observations are added to the posterior, one tracking the effective sample size of the particles for each step of the SMC sampler and one tracking the acceptance rate of the Metropolis-Hastings step when a resample-move is triggered. See Emerson and Aslett (2023) and Chopin (2002) for more details. Multiple Metropolis-Hastings steps can be performed when a resample-move step is triggered, and so for the acceptance rate plot observations are suffixed with "." and then the index of current Metropolis-Hastings step. For example the x-axis label for the acceptance rate of the 2nd Metropolis-Hastings step which was triggered by adding observation 1 to the posterior would be labelled "1.2". When the training data for the "IBIS" object created is large setting diagnostic_x_axis=="minimal" is recommended as it gives a more visually appealing output.

Value

No return value, called for side effects

References

- Emerson, S.R. and Aslett, L.J.M. (2023). Joint cohort and prediction modelling through graphical structure analysis (to be released)
- Chopin, N. (2002). A sequential particle filter method for static models. Biometrika, 89(3), 539-552, doi:10.1093/biomet/89.3.539

See Also

IBIS.logreg()

Examples

```
require(graphics)
# First we generate a co-variate matrix X and binary response vector y
CM <- matrix(rnorm(200),100,2)
rv <- sample(0:1,100,replace=TRUE)

# Now we can obtain 1000 samples from the posterior from a standard
# multivariate normal prior and plot the results
out <- IBIS.logreg(X = CM,y = rv)
plot(out,type = "samples")
plot(out,type = "fitted")
plot(out,type = "diagnostics",diagnostic_x_axis = "minimal")

# If we only wanted to view the second co-variate
plot(out,type = "samples",plot_var = 2)
plot(out,type = "fitted",plot_var = 2)
```

---

plot.UNCOVER *Plot various outputs of UNCOVER*

Description

Allows visualisation of many aspects of UNCOVER, including co-variate, posterior and diagnostic plots.

Usage

```
## S3 method for class 'UNCOVER'
plot(
  x,
  type = "covariates",
  plot_var = x$Minimum_Spanning_Tree_Variables,
  diagnostic_x_axis = "full",
  ...
)
```

Arguments

x	Object of class "UNCOVER"
type	Can be one of; "covariates" for cluster assignment visualisation for the co-variates, "fitted" for co-variate visualisation with respect to their fitted values, "samples" for posterior visualisation or "diagnostics" for diagnostic plots. See details. Defaults to "covariates".
plot_var	Vector specifying which columns (or associated logistic regression coefficients) of the co-variate matrix should be plotted. Does not apply when type=="diagnostics". Defaults to all columns being selected.
diagnostic_x_axis	Only applies if "type=="diagnostics". Either "full" (default) for all observations indices to be plotted on the x-axis or "minimal" for only some observations indices to be plotted on the x-axis.
...	Arguments to be passed to methods

Details

If type=="covariates", the resulting plot will be a ggpairs plot. The diagonal entries will be a plot of K density plots (K being the number of clusters). The off-diagonal elements are scatter-plots of the observations, given a label according to their true response and a colour based on their assigned cluster. If length(plot_var)==1 then the density plot and the scatter-plot are combined. If a cluster contains less than two data points the density will not be plotted.

If "type=="fitted", the resulting plot will be a ggpairs plot. The diagonal entries will be two density plots, one for data predicted to have response 0 by the model (red) and one for training data predicted to have response 1 by the model (green). The off-diagonal elements are scatter-plots of the observations, given a label according to their actual response and a colour scale based on their predicted response. If length(plot_var)==1 then the density plot and the scatter-plot are combined. If a predicted class (0 or 1) contains less than two data points the density will not be plotted.

If type=="samples", the resulting plot will be a ggpairs plot of the clusters posteriors, giving the coefficient densities in the diagonal and scatter-plots of the posterior samples in the off-diagonal. The transparency is increased in the upper triangle for scenarios when posteriors overlap.

If "type==diagnostics", the resulting plot depends on the deforestation criterion used to create the "UNCOVER" object:

"None" A plot tracking the overall log Bayesian evidence every time an action is executed.

"NoC" A plot tracking the overall log Bayesian evidence after every action and a plot tracking the number of clusters after every action.

"SoC" Three plots; one tracking the overall log Bayesian evidence after every action, one tracking the number of criterion breaking clusters after every action and one tracking the minimum cluster size after every action.

"MaxReg" A plot tracking the overall log Bayesian evidence every time an action is executed. Actions are coloured and each action has an associated coloured dashed line indicating the log Bayesian evidence plus the logarithm of the maximum tolerance provided.

"Validation" A plot tracking the overall log Bayesian evidence after every action (for both the training data and all of the data) and a plot tracking the robustness statistic after every deforestation action.

"Diverse" Three plots; one tracking the overall log Bayesian evidence after every action, one tracking the number of criterion breaking clusters after every action and one tracking the minimum minority class across clusters after every action.

Actions are defined as either edge removals, edge additions or edge additions in the deforestation stage. The syntax for an action will be the 'type_of_action.edge'. For example the removal of an edge connecting observation 1 and observation 2 will be displayed 'Rem.1-2'. If the edge was being added this would be displayed 'Def.Add.1-2' if in the deforestation stage and 'Add.1-2' otherwise. When the data for the "UNCOVER" object created is large setting diagnostic_x_axis=="minimal" is recommended as it gives a more visually appealing output.

Value

No return value, called for side effects

See Also

UNCOVER()

Examples

```
require(graphics)
# First we generate a co-variate matrix and binary response vector
CM <- matrix(rnorm(200),100,2)
rv <- sample(0:1,100,replace=TRUE)

# We can then run our algorithm for each of the different deforestation
# criteria
UN.none <- UNCOVER(X = CM,y = rv, deforest_criterion = "None", verbose = FALSE)
UN.noc <- UNCOVER(X = CM,y = rv, deforest_criterion = "NoC",
                  options = UNCOVER.opts(max_K = 3), verbose = FALSE)
UN.soc <- UNCOVER(X = CM,y = rv, deforest_criterion = "SoC",
                  options = UNCOVER.opts(min_size = 10), verbose = FALSE)
UN.maxreg <- UNCOVER(X = CM,y = rv, deforest_criterion = "MaxReg",
                     options = UNCOVER.opts(reg = 1), verbose = FALSE)
UN.validation <- UNCOVER(X = CM,y = rv, deforest_criterion = "Validation",
                         options = UNCOVER.opts(train_frac = 0.8),
                         verbose = FALSE)
UN.diverse <- UNCOVER(X = CM,y = rv, deforest_criterion = "Diverse",
                      options = UNCOVER.opts(n_min_class = 2), verbose = FALSE)
plot(UN.none,type = "covariates")
plot(UN.none,type = "fitted")
plot(UN.none,type = "samples")
plot(UN.none,type = "diagnostics",diagnostic_x_axis = "minimal")
plot(UN.noc,type = "diagnostics",diagnostic_x_axis = "minimal")
plot(UN.soc,type = "diagnostics",diagnostic_x_axis = "minimal")
plot(UN.maxreg,type = "diagnostics",diagnostic_x_axis = "minimal")
plot(UN.validation,type = "diagnostics",diagnostic_x_axis = "minimal")
```

```
plot(UN.diverse,type = "diagnostics",diagnostic_x_axis = "minimal")

# If we only wanted to view the second co-variate
plot(UN.none,type = "covariates",plot_var=2)
plot(UN.none,type = "fitted",plot_var=2)
plot(UN.none,type = "samples",plot_var=2)
```

`predict.IBIS` *Prediction method for IBIS*

Description

Predicts the response of new observations using an object of class `"IBIS"`.

Usage

```
## S3 method for class 'IBIS'
predict(object, newX = NULL, type = "prob", ...)
```

Arguments

`object`	Object of class `"IBIS"`
`newX`	Data frame containing new observations to predict. If not specified the fitted values will be returned instead.
`type`	Either `"prob"` for a probabilistic output or `"response"` for a hard output of the predicted response
`...`	Additional arguments affecting the predictions produced

Details

Note that this is a Bayesian prediction method as objects with class `"IBIS"` will provide samples from a posterior.

Value

Either a matrix of response probabilities for each observation or a vector of predicted responses for each observation.

See Also

`IBIS.logreg()`

Examples

```
# First we generate a co-variate matrix X and binary response vector y
CM <- data.frame(X1 = rnorm(100),X2 = rnorm(100))
rv <- sample(0:1,100,replace=TRUE)

# Now we can obtain 1000 samples from the posterior from a standard
# multivariate normal prior
out <- IBIS.logreg(X = CM,y = rv)

# The fitted values of out can be obtained
predict(out)
predict(out,type = "response")

# We can also predict the response for new data
CM.2 <- data.frame(X1 = rnorm(10),X2 = rnorm(10))
cbind(CM.2,predict(out,newX = CM.2))
```

`predict.UNCOVER` *Prediction method for UNCOVER*

Description

Predicts the response of new observations and their cluster assignment using an object of class `"UNCOVER"`.

Usage

```
## S3 method for class 'UNCOVER'
predict(object, newX = NULL, type = "prob", ...)
```

Arguments

`object`	Object of class `"UNCOVER"`
`newX`	Data frame containing new observations to predict. If not specified the fitted values will be returned instead.
`type`	Either `"prob"` for a probabilistic response prediction or `"response"` for a hard output of the predicted response
`...`	Additional arguments affecting the predictions produced

Details

Note that this is a Bayesian prediction method and so samples of the posterior, defined by `"UNCOVER"` object provided, will be obtained through SMC methods for prediction. See `IBIS.logreg()` for more details.

Value

Either a data frame of response probabilities with cluster assignment for each observation or a data frame of predicted responses with cluster assignment for each observation.

See Also

UNCOVER(), IBIS.logreg()

Examples

```
# First we generate a co-variate matrix and binary response vector
CM <- data.frame(X1 = rnorm(100),X2 = rnorm(100))
rv <- sample(0:1,100,replace=TRUE)

# We can then run UNCOVER with no deforestation criteria
UN.none <- UNCOVER(X = CM,y = rv, deforest_criterion = "None", verbose = FALSE)

# The fitted values of UN.none can then be obtained
predict(UN.none)
predict(UN.none,type = "response")

# We can also predict the response for new data
CM.2 <- data.frame(X1 = rnorm(10),X2 = rnorm(10))
cbind(CM.2,predict(UN.none,newX = CM.2))
```

---

print.IBIS *Print IBIS*

Description

Prints summary information from an IBIS object.

Usage

```
## S3 method for class 'IBIS'
print(x, ...)
```

Arguments

x	Object of class "IBIS"
...	Further arguments passed to or from other methods

Details

When running the function IBIS.logreg() the printed information will contain information regarding; the number of samples, the mean of those samples and the log Bayesian evidence of the posterior.

Value

No return value, called for side effects

See Also

IBIS.logreg()

---

print.UNCOVER *Print UNCOVER*

Description

Prints summary information from an UNCOVER object.

Usage

```
## S3 method for class 'UNCOVER'
print(x, ...)
```

Arguments

x	Object of class "UNCOVER"
...	Further arguments passed to or from other methods

Details

When running the function UNCOVER() the printed information will contain information regarding; the number of clusters, the cluster sizes, the sub-model log Bayesian evidences and the total model log Bayesian evidence.

Value

No return value, called for side effects

See Also

UNCOVER()

UNCOVER *Utilising Normalisation Constant Optimisation Via Edge Removal*

Description

Generates cohorts for a data set through removal of edges from a graphical representation of the co-variates. Edges are removed (or reintroduced) by considering the normalisation constant (or Bayesian evidence) of a multiplicative Bayesian logistic regression model.

The first stage of the function is concerned purely with a greedy optimisation of the Bayesian evidence through edge manipulation. The second stage then addresses any other criteria (known as deforestation conditions) expressed by the user through reintroduction of edges.

Usage

```
UNCOVER(
  X,
  y,
  mst_var = NULL,
  options = UNCOVER.opts(),
  stop_criterion = 5,
  deforest_criterion = "None",
  prior_mean = rep(0, ncol(X) + 1),
  prior_var = diag(ncol(X) + 1),
  verbose = TRUE
)
```

Arguments

X	Co-variate matrix
y	Binary response vector
mst_var	A vector specifying which variables of the co-variate matrix will be used to form the graph. If not specified all variables will be used.
options	Additional arguments that can be specified for UNCOVER. See UNCOVER.opts() for details. Can be ignored.
stop_criterion	What is the maximum number of clusters allowed before we terminate the first stage and begin deforestation. Defaults to 5.
deforest_criterion	Constraint type which the final model must satisfy. Can be one of "NoC", "SoC", "MaxReg", "Validation", "Diverse" or "None". See details. Defaults to "None".
prior_mean	Mean for the multivariate normal prior used in the SMC sampler. See details. Defaults to the origin.
prior_var	Variance matrix for the multivariate normal prior used in the SMC sampler. See details. Defaults to the identity matrix.
verbose	Do you want the progress of the algorithm to be shown? Defaults to TRUE.

Details

Assumes a Bayesian logistic regression model for each cohort, with the overall model being a product of these sub-models.

A minimum spanning tree graph is first constructed from a subset of the co-variates. Then at each iteration, each edge in the current graph is checked to see if removal to split a cohort is beneficial, and then either we selected the optimal edge to remove or we conclude it is not beneficial to remove any more edges. At the end of each iteration we also check the set of removed edges to see if it is beneficial to reintroduce any previously removed edges. After this process has ended we then reintroduce edges in the removed set specifically to meet the criteria set by the user in the most optimal manner possible through a greedy approach. For more details see the Emerson and Aslett (2023).

The graph can be undergo deforestation to meet 6 possible criteria:

1. `"NoC"`: Number of Clusters - we specify a maximum number of clusters (`options$max_K`) we can tolerate in the final output of the algorithm.
2. `"SoC"`: Size of Clusters - we specify a minimum number of observations (`options$min_size`) we can tolerate being assigned to a cluster in the final output of the algorithm.
3. `"MaxReg"`: Maximal Regret - we give a maximum tolerance (`exp(options$reg)`) that we allow the Bayesian evidence to decrease by reintroducing an edge.
4. `"Validation"`: Validation Data - we split (using `options$train_frac`) the data into training and validation data, apply the first stage of the algorithm on the training data and the introduce the validation data for the deforestation stage. Edges are reintroduced if they lead to improved prediction of the validation data using the training data model (i.e. we aim to maximise a robustness statistic).
5. `"Diverse"`: Diverse Response Class Within Clusters - We specify a minimum number of observations (`options$n_min_class`) in a cluster that have the minority response class associated to them (the minimum response class is determined for each cluster).
6. `"None"`: No Criteria Specified - we do not go through the second deforestation stage of the algorithm.

All deforestation criteria other than `"None"` require additional arguments to be specified in `options`. See examples and `UNCOVER.opts()` for more information. It is never recommended to use anything other than `UNCOVER.opts` to provide the `options` argument.

The prior used for the UNCOVER procedure will take the form of a multivariate normal, where the parameters can be specified directly by the user. It is however possible to override this default prior distributional form by specifying `prior.override=TRUE` and providing the relevant prior functions in `UNCOVER.opts`.

The diagnostic data frames will track various outputs of the UNCOVER procedure depending on the deforestation criterion. All data frames will contain an action (removal or addition of an edge to the graph) and the total log Bayesian evidence of the model gained through deployment of that action (for `"Validation"` this will be two columns, one for the training data and one for all of the data). `"NoC"` will also track the number of clusters, `"SoC"` will track the minimum cluster size and the number of criterion breaking clusters, `"Validation"` will track the robustness statistic and `"Diverse"` will track the minimum minority class across all clusters alongside the number of criterion breaking clusters.

Value

An object of class "UNCOVER", which is a list consisting of:

`Covariate_Matrix` The co-variate matrix provided.

`Response_Vector` The binary response vector provided.

`Minimum_Spanning_Tree_Variables` A vector of indices for the co-variates used to construct the minimum spanning tree.

`Control` A list of the additional arguments specified by `options`.

`Deforestation_Criterion` The deforestation criterion specified.

`Prior_Mean` The mean of multivariate normal prior. Meaningless if prior is overridden in `options`.

`Prior_Variance` The variance of multivariate normal prior. Meaningless if prior is overridden in `options`.

`Model` List containing; the cluster allocation of the training data, the log Bayesian evidences of the sub-models, the final graph of the clustered data, the number of clusters, the edges which were removed from the graph and a diagnostics data frame (the contents of which vary depending on the deforestation criterion).

If `deforest_criterion=="Validation"` then `Model` is instead a list of two lists; one containing the model information for the training data (`Training_Data`) and the other containing model information for all of the data (`All_Data`). Diagnostic information is only included in the `All_Data` list.

References

- Emerson, S.R. and Aslett, L.J.M. (2023). Joint cohort and prediction modelling through graphical structure analysis (to be released)

See Also

`UNCOVER.opts()`, `print.UNCOVER()`, `predict.UNCOVER()`, `plot.UNCOVER()`

Examples

```
# First we generate a co-variate matrix and binary response vector
CM <- matrix(rnorm(200),100,2)
rv <- sample(0:1,100,replace=TRUE)

# We can then run our algorithm to see what cohorts are selected for each
# of the different deforestation criteria
UN.none <- UNCOVER(X = CM,y = rv, deforest_criterion = "None",
                   verbose = FALSE)
UN.noc <- UNCOVER(X = CM,y = rv, deforest_criterion = "NoC",
                  options = UNCOVER.opts(max_K = 3), verbose = FALSE)
UN.soc <- UNCOVER(X = CM,y = rv, deforest_criterion = "SoC",
                  options = UNCOVER.opts(min_size = 10), verbose = FALSE)
UN.maxreg <- UNCOVER(X = CM,y = rv, deforest_criterion = "MaxReg",
                     options = UNCOVER.opts(reg = 1), verbose = FALSE)
```

```
UN.validation <- UNCOVER(X = CM,y = rv, deforest_criterion = "Validation",
                         options = UNCOVER.opts(train_frac = 0.8),
                         verbose = FALSE)
UN.diverse <- UNCOVER(X = CM,y = rv, deforest_criterion = "Diverse",
                      options = UNCOVER.opts(n_min_class = 2),
                      verbose = FALSE)
clu_al_mat <- rbind(UN.none$Model$Cluster_Allocation,
                    UN.noc$Model$Cluster_Allocation,
                    UN.soc$Model$Cluster_Allocation,
                    UN.maxreg$Model$Cluster_Allocation,
                    UN.validation$Model$All_Data$Cluster_Allocation,
                    UN.diverse$Model$Cluster_Allocation)
# We can create a matrix where each entry shows in how many of the methods
# did the indexed observations belong to the same cluster
obs_con_mat <- matrix(0,100,100)
for(i in 1:100){
for(j in 1:100){
obs_con_mat[i,j] <- length(which(clu_al_mat[,i]-clu_al_mat[,j]==0))/6
obs_con_mat[j,i] <- obs_con_mat[i,j]
}
}
head(obs_con_mat)

# We can also view the outputted overall Bayesian evidence of the five
# models as well
c(sum(UN.none$Model$Log_Marginal_Likelihoods),
  sum(UN.noc$Model$Log_Marginal_Likelihoods),
  sum(UN.soc$Model$Log_Marginal_Likelihoods),
  sum(UN.maxreg$Model$Log_Marginal_Likelihoods),
  sum(UN.validation$Model$All_Data$Log_Marginal_Likelihoods),
  sum(UN.diverse$Model$Log_Marginal_Likelihoods))

# If we don't assume the prior for the regression coefficients is a
# standard multivariate normal but instead a multivariate normal with
# different parameters
UN.none.2 <- UNCOVER(X = CM,y = rv, deforest_criterion = "None",
                     prior_mean = rep(1,3), prior_var = 0.5*diag(3),
                     verbose = FALSE)
c(sum(UN.none$Model$Log_Marginal_Likelihoods),
  sum(UN.none.2$Model$Log_Marginal_Likelihoods))
# We can also specify a completely different prior, for example a
# multivariate independent uniform
rmviu <- function(n,a,b){
return(mapply(FUN = function(min.vec,max.vec,pn){
                      stats::runif(pn,a,b)},min.vec=a,max.vec=b,
                                    MoreArgs = list(pn = n)))
}
dmviu <- function(x,a,b){
for(ii in 1:ncol(x)){
  x[,ii] <- dunif(x[,ii],a[ii],b[ii])
}
return(apply(x,1,prod))
}
```

```
UN.none.3 <- UNCOVER(X = CM,y = rv,deforest_criterion = "None",
                     options = UNCOVER.opts(prior.override = TRUE,
                                            rprior = rmviu,
                                            dprior = dmviu,a=rep(0,3),
                                            b=rep(1,3)),verbose = FALSE)
c(sum(UN.none$Model$Log_Marginal_Likelihoods),
  sum(UN.none.2$Model$Log_Marginal_Likelihoods),
  sum(UN.none.3$Model$Log_Marginal_Likelihoods))

# We may only wish to construct our minimum spanning tree based on the first
# variable
UN.none.4 <- UNCOVER(X = CM,y = rv,mst_var = 1,deforest_criterion = "None",
                     verbose = FALSE)
c(sum(UN.none$Model$Log_Marginal_Likelihoods),
  sum(UN.none.4$Model$Log_Marginal_Likelihoods))

# Increasing the stop criterion may uncover more clustering structure within
# the data, but comes with a time cost
system.time(UNCOVER(X = CM,y = rv,stop_criterion = 4,verbose = FALSE))
system.time(UNCOVER(X = CM,y = rv,stop_criterion = 6,verbose = FALSE))
```

UNCOVER.opts *Additional argument generator for* UNCOVER()

Description

This function is used to specify additional arguments to UNCOVER.

Usage

```
UNCOVER.opts(
  N = 1000,
  train_frac = 1,
  max_K = Inf,
  min_size = 0,
  reg = 0,
  n_min_class = 0,
  SMC_thres = 30,
  BIC_memo_thres = Inf,
  SMC_memo_thres = Inf,
  ess = N/2,
  n_move = 1,
  prior.override = FALSE,
  rprior = NULL,
  dprior = NULL,
  diagnostics = TRUE,
  RIBIS_thres = 30,
```

```
  BIC_cache = cachem::cache_mem(max_size = 1024 * 1024^2, evict = "lru"),
  SMC_cache = cachem::cache_mem(max_size = 1024 * 1024^2, evict = "lru"),
  ...
)
```

Arguments

N	Number of particles for the SMC sampler. Defaults to 1000.
train_frac	What fraction of the data should be used for training. Should only be directly specified if deforest_criterion == "Validation. Defaults to 1.
max_K	The maximum number of clusters allowed in the final output. Should only be directly specified if deforest_criterion == "NoC. Defaults to Inf.
min_size	The minimum number of observations allowed for any cluster in the final model. Should only be directly specified if deforest_criterion == "SoC. Defaults to 0.
reg	Numerical natural logarithm of the tolerance parameter. Must be positive. Should only be directly specified if deforest_criterion == "MaxReg. Defaults to 0.
n_min_class	Each cluster will have an associated minority class. n_min_class specifies a minimum number of observations that should have that class for each and every cluster. Should only be directly specified if deforest_criterion == "Diverse. Defaults to 0.
SMC_thres	The threshold for which the number of observations needs to exceed to consider using BIC as an estimator. Defaults to 30 if not specified.
BIC_memo_thres	Only used when estimating the log Bayesian evidence of a cluster using BIC. When the number of observations exceeds BIC_memo_thres the function checks for similar inputs evaluated previously. See details. Defaults to never checking.
SMC_memo_thres	Only used when estimating the log Bayesian evidence of a cluster using SMC. When the number of observations exceeds SMC_memo_thres the function checks for similar inputs evaluated previously. See details. Defaults to never checking.
ess	Effective Sample Size Threshold: If the effective sample size of the particles falls below this value then a resample move step is triggered. Defaults to N/2.
n_move	Number of Metropolis-Hastings steps to apply each time a resample move step is triggered. Defaults to 1.
prior.override	Are you overriding the default multivariate normal form of the prior? Defaults to FALSE.
rprior	Function which produces samples from your prior if the default prior form is to be overridden. If using the default prior form this does not need to be specified.
dprior	Function which produces your specified priors density for inputted samples if the default prior form is to be overridden. If using the default prior form this does not need to be specified.
diagnostics	Should diagnostic data be recorded and outputted? Defaults to TRUE.
RIBIS_thres	The threshold for which the number of observations needs to exceed to consider ever using RIBIS as an estimator. Defaults to 30 if not specified. See details.

BIC_cache	The cache for the function which estimates the log Bayesian evidence using BIC. Defaults to a cache with standard size and least recently used eviction policy.
SMC_cache	The cache for the function which estimates the log Bayesian evidence using SMC. Defaults to a cache with standard size and least recently used eviction policy.
...	Additional arguments required for complete specification of the two prior functions given, if the default prior form is to be overridden.

Details

This function should only be used to provide additional control arguments to UNCOVER. Arguments that are for a particular deforestation criteria should not be altered from the defaults for other deforestation criteria.

BIC refers to the Bayesian Information Criterion. The use of BIC when estimating the log Bayesian evidence is valid assuming the number of observations is large, and if specifying SMC_thres this should be balanced with computational expense (as the function which relies on BIC values is much faster than the SMC sampler).

In an attempt to improve computational time, the SMC sampler along with the function which uses BIC values are memoised, with the cache for each of these memoised functions be specified by SMC_cache and BIC_cache respectively. See memoise::memoise() for more details. If we do not get and each match from the function input to a previously evaluated input, we may wish to search the cache for similar inputs which could provide a reasonable starting point. Checking the cache however takes time, and so we allow the user to specify at which size of cluster to they deem it worthwhile to check. Which value threshold to select to optimise run time is problem specific, however for BIC_memo_thres it is almost always beneficial to never check the cache (the exception for this being when the cluster sizes are extremely large, for example containing a million observations). SMC_memo_thres can be much lower as the SMC sampler is a much more expensive function to run. See Emerson and Aslett (2023) for more details.

RIBIS_thres can be specified to have a higher value to ensure that the asymptotic properties which Reverse Iterated Batch Importance Sampling (RIBIS) relies upon hold. See Emerson and Aslett (2023) for more details.

Specifying rprior and dprior will not override the default prior form unless prior.override=TRUE. If a multivariate normal form is required then the arguments for this prior should be specified in UNCOVER.

Value

A list consisting of:

N Number of particles for the SMC sampler

train_frac Training data fraction

max_K Maximum number of clusters allowed

min_size Minimum size of clusters allowed

reg Log of the maximum regret tolerance parameter

n_min_class Minimum size of cluster minority class allowed

`SMC_thres` Threshold for when estimation with BIC is attempted

`BIC_memo_thres` Threshold for when we review previous inputs of the BIC function for similarities

`SMC_memo_thres` Threshold for when we review previous inputs of the SMC function for similarities

`ess` Effective Sample Size Threshold

`n_move` Number of Metropolis-Hastings steps

`rprior` Function which produces samples from your prior. `NULL` if `prior.override==FALSE`.

`dprior` Function which produces your specified priors density for inputted samples. `NULL` if `prior.override==FALSE`.

`prior.override` Logical value indicating if the prior has been overridden or not

`diagnostics` Logical value indicating whether diagnostic information should be included in the output of `UNCOVER`

`RIBIS_thres` The threshold for allowing the use of RIBIS

`BIC_cache` Cache for the memoised function which estimates the log Bayesian evidence using BIC

`SMC_cache` Cache for the memoised function which estimates the log Bayesian evidence using SMC

`MoreArgs` A list of the additional arguments required for `rprior` and `dprior`. `NULL` if `prior.override==FALSE`.

References

- Emerson, S.R. and Aslett, L.J.M. (2023). Joint cohort and prediction modelling through graphical structure analysis (to be released)

See Also

`UNCOVER()`

Examples

```
#Specifying a multivariate independent uniform prior

rmviu <- function(n,a,b){
return(mapply(FUN = function(min.vec,max.vec,pn){stats::runif(pn,a,b)},
              min.vec=a,max.vec=b,MoreArgs = list(pn = n)))
}
dmviu <- function(x,a,b){
for(ii in 1:ncol(x)){
  x[,ii] <- dunif(x[,ii],a[ii],b[ii])
}
return(apply(x,1,prod))
}

UNCOVER.opts(prior.override = TRUE,rprior = rmviu,
                dprior = dmviu,a=rep(0,3),b=rep(1,3))


# If we generate a co-variate matrix and binary response vector
```

```
CM <- matrix(rnorm(200),100,2)
rv <- sample(0:1,100,replace=TRUE)

# We can then run our algorithm with a SMC threshold of 50 and a SMC cache
# checking threshold of 25 to see if this is quicker than the standard
# version
system.time(UNCOVER(X = CM,y = rv,verbose = FALSE))
system.time(UNCOVER(X = CM,y = rv,
                    options = UNCOVER.opts(SMC_thres = 50),
                    verbose = FALSE))
system.time(UNCOVER(X = CM,y = rv,
                    options = UNCOVER.opts(SMC_thres = 50,
                                           SMC_memo_thres = 25),
                    verbose = FALSE))
```

Index