Package ‘sbfc’

June 21, 2020

Type Package

Title Selective Bayesian Forest Classifier

Version 1.0.2

Date 2020-06-21

Author Viktoriya Krakovna

Maintainer Viktoriya Krakovna <vkrakovna@gmail.com>

URL http://github.com/vkrakovna/sbfc

BugReports http://github.com/vkrakovna/sbfc/issues

Description An MCMC algorithm for simultaneous feature selection and classification, and visualization of the selected features and feature interactions. An implementation of SBFC by Krakovna, Du and Liu (2015), arXiv:1506.02371.

License GPL (>= 2)

Depends R (>= 2.10), DiagrammeR

Imports Rcpp (>= 0.12.2),
Matrix,
discretization

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 5.0.1

LazyData true

R topics documented:

sbfc-package 2
corral_augmented 2
data_disc 3
edge_density_plot 4
heart 5
logposterior_plot 5
madelon 6
sbfc 6

sbfc_graph 8
signal_size_plot 9
signal_var_proportion 9

Index **11**

`sbfc-package` *Selective Bayesian Forest Classifier*

Description

An MCMC algorithm for simultaneous feature selection and classification, and visualization of the selected features and feature interactions. An implementation of SBFC by Krakovna, Du and Liu (2015), arXiv:1506.02371.

Details

This package was not yet installed at build time.

Index: This package was not yet installed at build time.
Run the SBFC algorithm on a data set using the `sbfc` function. Make SBFC graphs based on the MCMC samples using the `sbfc_graph` function. Other analysis, e.g. feature selection plots using `signal_var_proportion` (based on how often each variable appeared in the signal group).

Author(s)

Viktoriya Krakovna Maintainer: Viktoriya Krakovna <vkrakovna@gmail.com>

`corral_augmented` *Augmented corral data set: synthetic data with correlated attributes augmented with noise features*

Description

This is an artificial domain where the target concept is (X1^X2) V (X3^X4).
Data set by R. Kohavi. Training and test splits from SGI.
The first 6 features are the real features from the original corral data set. The rest are noise features added by V. Krakovna by shuffling copies of real features.
The SBFC paper uses subsets of this data set with the first 100 and 1000 features.

This is an artificial domain where the target concept is (X1^X2) V (X3^X4).
Data set by R. Kohavi. Training and test splits from SGI.
The first 6 features are the real features from the original corral data set. The rest are noise features added by Author by shuffling copies of real features.
The SBFC paper uses subsets of this data set with the first 100 and 1000 features.

Usage

```
data(corral_augmented)

data(corral_augmented)
```

Format

`TrainX` A matrix with 128 rows and 10000 columns.

`TrainY` A vector with 128 rows.

References

SGI listing for corral data set

SBFC paper describing augmentation of corral data set

SGI listing for corral data set

Examples

```
corral_result = sbfc(data=list(TrainX=corral_augmented$TrainX[,1:6],
                                         TrainY = corral_augmented$TrainY))
corral100_result = sbfc(data=list(TrainX=corral_augmented$TrainX[,1:100],
                                  TrainY = corral_augmented$TrainY))
corral_result = sbfc(data=list(TrainX=corral_augmented$TrainX[,1:6],
                                         TrainY = corral_augmented$TrainY))
corral100_result = sbfc(data=list(TrainX=corral_augmented$TrainX[,1:100],
                                  TrainY = corral_augmented$TrainY))
```

`data_disc` *Data set discretization and formatting*

Description

Removes rows containing missing data, and discretizes the data set using Minimum Description Length Partitioning (MDLP).

Usage

```
data_disc(data, n_train = NULL, missing = "?")
```

Arguments

`data`	Data frame, where the last column must be the class variable.
`n_train`	Number of data frame rows to use as the training set - the rest are used for the test set. If NULL, all rows are used for training, and there is no test set (default=NULL).
`missing`	Label that denotes missing values in your data frame (default='?').

Value

A discretized data set:

`TrainX` Matrix containing the training data.

`TrainY` Vector containing the class labels for the training data.

`TestX` Matrix containing the test data (optional).

`TestY` Vector containing the class labels for the test data (optional).

Examples

```
data(iris)
iris_disc = data_disc(iris)
```

`edge_density_plot` *Plots the density of edges in a given group over the MCMC iterations*

Description

Plots the edge density for the given group for a range of the MCMC iterations (indicated by `start` and `end`).

Usage

```
edge_density_plot(sbfc_result, group, start = 0, end = 1)
```

Arguments

`sbfc_result` An object of class `sbfc`.

`group` Which group (0 or 1) to plot edge density for.

`start` The start of the included range of MCMC iterations (default=0, i.e. starting with the first iteration).

`end` The end of the included range of MCMC iterations (default=1, i.e. ending with the last iteration).

`heart` *Heart disease data set: disease outcomes given health attributes*

Description

Data set from UCI repository, discretized using the `mdlp` package.

Data set from UCI repository, discretized using the `mdlp` package.

Usage

```
data(heart)
```

```
data(heart)
```

Format

`TrainX` A matrix with 270 rows and 13 columns.

`TrainY` A vector with 270 rows.

References

UCI heart data set

SGI listing for heart data set

UCI heart data set

SGI listing for heart data set

`logposterior_plot` *Log posterior plot*

Description

Plots the log posterior for a range of the MCMC iterations (indicated by `start` and `end`).

Usage

```
logposterior_plot(sbfc_result, start = 0, end = 1, type = "trace")
```

Arguments

`sbfc_result` An object of class `sbfc`.

`start` The start of the included range of MCMC iterations (default=0, i.e. starting with the first iteration).

`end` The end of the included range of MCMC iterations (default=1, i.e. ending with the last iteration).

`type` Type of plot (either `trace` or `acf`, default=`trace`).

madelon *Madelon data set: synthetic data from NIPS 2003 feature selection challenge*

Description

This is a two-class classification problem. The difficulty is that the problem is multivariate and highly non-linear. Of the 500 features, 20 are real features, 480 are noise features.
Data set from UCI repository, discretized using median cutoffs.

This is a two-class classification problem. The difficulty is that the problem is multivariate and highly non-linear. Of the 500 features, 20 are real features, 480 are noise features.
Data set from UCI repository, discretized using median cutoffs.

Usage

```
data(madelon)
```

```
data(madelon)
```

Format

`TrainX` A matrix with 2000 rows and 500 columns.

`TrainY` A vector with 2000 rows.

`TestX` A matrix with 600 rows and 500 columns.

`TestY` A vector with 600 rows.

References

UCI madelon data set

UCI madelon data set

sbfc *Selective Bayesian Forest Classifier (SBFC) algorithm*

Description

Runs the SBFC algorithm on a discretized data set. To discretize your data, use the `data_disc` command.

Usage

```
sbfc(data, nstep = NULL, thin = 50, burnin_denom = 5, cv = T,
  thinoutputs = F, alpha = 5, y_penalty = 1, x_penalty = 4)
```

Arguments

`data` Discretized data set:

- `TrainX` Matrix containing the training data.
- `TrainY` Vector containing the class labels for the training data.
- `TestX` Matrix containing the test data (optional).
- `TestY` Vector containing the class labels for the test data (optional).

`nstep` Number of MCMC steps, default max(10000, 10 * ncol(TrainX)).

`thin` Thinning factor for the MCMC.

`burnin_denom` Denominator of the fraction of total MCMC steps discarded as burnin (default=5).

`cv` Do cross-validation on the training set (if test set is not provided).

`thinoutputs` Return thinned MCMC outputs (parents, groups, trees, logposterior), rather than all outputs (default=FALSE).

`alpha` Dirichlet hyperparameter(default=1)

`y_penalty` Prior coefficient for y-edges, which penalizes signal group size (default=1)

`x_penalty` Prior coefficient for x-edges, which penalizes tree size (default=4)

Details

Data needs to be discretized before running SBFC.
If the test data matrix TestX is provided, SBFC runs on the entire training set TrainX, and provides predicted class labels for the test data. If the test data class vector TestY is provided, the accuracy is computed. If the test data matrix TestX is not provided, and cv is set to TRUE, SBFC performs cross-validation on the training data set TrainX, and returns predicted classes and accuracy for the training data.

Value

An object of class `sbfc`:

`accuracy` Classification accuracy (on the test set if provided, otherwise cross-validation accuracy on training set).

`predictions` Vector of class label predictions (for the test set if provided, otherwise for the training set).

`probabilities` Matrix of class label probabilities (for the test set if provided, otherwise for the training set).

`runtime` Total runtime of the algorithm in seconds.

`parents` Matrix representing the structures sampled by MCMC, where parents[i,j] is the index of the parent of node j at iteration i (0 if node is a root).

`groups` Matrix representing the structures sampled by MCMC, where groups[i,j] indicates which group node j belongs to at iteration j (0 is noise, 1 is signal).

`trees` Matrix representing the structures sampled by MCMC, where trees[i,j] indicates which tree node j belongs to at iteration j.

`logposterior` Vector representing the log posterior at each iteration of the MCMC.

Parameters `nstep`, `thin`, `burnin_denom`, `cv`, `thinoutputs`, `alpha`, `y_penalty`, `x_penalty`.

If `cv=TRUE`, the MCMC samples from the first fold are returned (`parents`, `groups`, `trees`, `logposterior`).

Examples

```
data(madelon)
madelon_result = sbfc(madelon)
data(heart)
heart_result = sbfc(heart, cv=FALSE)
```

`sbfc_graph` *SBFC graph*

Description

Plots a sampled MCMC graph or an average of sampled graphs using Graphviz.
In average graphs, nodes are color-coded according to importance - the proportion of samples where the node appeared in Group 1 (dark-shaded nodes appear more often). In average graphs, thickness of edges also corresponds to importance: the proportion of samples where the edge appeared.

Usage

```
sbfc_graph(sbfc_result, iter = 10000, average = T, edge_cutoff = 0.1,
  single_noise_nodes = F, labels = paste0("X", 1:ncol(sbfc_result$parents)),
  save_graphviz_code = F, colorscheme = "blues", ncolors = 7,
  width = NULL, height = NULL)
```

Arguments

`sbfc_result` An object of class `sbfc`.

`iter` MCMC iteration of the sampled graph to plot, if `average=F` (default=10000).

`average` Plot an average of sampled MCMC graphs (default=TRUE).

`edge_cutoff` The average graph includes edges that appear in at least this fraction of the sampled graphs, if `average=T` (default=0.1).

`single_noise_nodes` Plot single-node trees that appear in the noise group (Group 0) in at least 80 percent of the samples, which can be numerous for high-dimensional data sets (default=FALSE).

`labels` A vector of node labels (default=`c("X1","X2",...)`).

`save_graphviz_code` Save the Graphviz source code in a .gv file (default=FALSE).

`colorscheme` Graphviz color scheme for the nodes (default="blues").

`ncolors` number of colors in the palette (default=7).

`width` An optional parameter for specifying the width of the resulting graphic in pixels.

`height` An optional parameter for specifying the height of the resulting graphic in pixels.

Examples

```
data(madelon)
madelon_result = sbfc(madelon)
sbfc_graph(madelon_result)
sbfc_graph(madelon_result, average=FALSE, iter=5000) # graph for 5000th iteration
sbfc_graph(madelon_result, single_noise_nodes=TRUE) # wide graph with 480 single nodes

data(heart)
heart_result = sbfc(heart)
heart_labels = c("Age", "Sex", "Chest Pain", "Rest Blood Pressure", "Cholesterol",
"Blood Sugar", "Rest ECG", "Max Heart Rate", "Angina", "ST Depression", "ST Slope",
"Fluoroscopy Colored Vessels", "Thalassemia")
sbfc_graph(heart_result, labels=heart_labels, width=700)
```

`signal_size_plot` *Trace plot of Group 1 size*

Description

Plots the Group 1 size for a range of the MCMC iterations (indicated by `start` and `end`).

Usage

```
signal_size_plot(sbfc_result, start = 0, end = 1, samples = F)
```

Arguments

`sbfc_result` An object of class `sbfc`.

`start` The start of the included range of MCMC iterations (default=0, i.e. starting with the first iteration).

`end` The end of the included range of MCMC iterations (default=1, i.e. ending with the last iteration).

`samples` Calculate signal group size based on sampled MCMC graphs after burn-in and thinning, rather than graphs from all iterations (default=FALSE).

`signal_var_proportion` *Signal variable proportion*

Description

For each variable, computes the proportion of the samples in which this variable is in the signal group (Group 1). Plots the top `nvars` variables in decreasing order of signal proportion.

Usage

```
signal_var_proportion(sbfc_result, nvars = 10, samples = F,
  labels = paste0("X", 1:ncol(sbfc_result$parents)), label_size = 1,
  rotate_labels = F)
```

Arguments

`sbfc_result` An object of class `sbfc`.

`nvars` Number of top signal variables to include in the plot (default=10).

`samples` Calculate signal variable proportion based on sampled MCMC graphs after burn-in and thinning, rather than graphs from all iterations (default=FALSE).

`labels` A vector of node labels (default=`c("X1","X2",...)`).

`label_size` Size of variable labels on the X-axis (default=1).

`rotate_labels` Rotate x-axis labels by 90 degrees to make them vertical (default=FALSE)

Value

Signal proportion for the top `nvars` variables in decreasing order.

Index

∗Topic **datasets**
    corral_augmented, 2
    heart, 5
    madelon, 6
∗Topic **package**
    sbfc-package, 2

corral_augmented, 2

data_disc, 3, *6*

edge_density_plot, 4

heart, 5

logposterior_plot, 5

madelon, 6

sbfc, 6
sbfc-package, 2
sbfc_graph, 8
signal_size_plot, 9
signal_var_proportion, 9