Package 'Condens8R'

September 24, 2026

Version 0.6.2

Date 2026-09-24

Title Classes and Methods for Learning Complex Decision Tree Models

Depends R (>= 4.4), Modeler (>= 3.4.9)

Imports methods, stats, logicFS, LogicReg, BimodalIndex, ClassDiscovery, Mercator (>= 1.1.7), cluster, apcluster

Description Learns decision tree models with support for complex models at each node.

License Apache License (== 2.0)

LazyLoad yes

URL http://silicovore.com/OOMPA/classpred.html

NeedsCompilation no

Author Kevin R. Coombes [aut, cre]

Maintainer Kevin R. Coombes <krc@silicovore.com>

Contents

`createTree` *Create a Decision tree*

Description

Functions to create a binary decision tree with a complex algorithm at each nternal node.

Usage

```
createTree(data, metric, label = "H", pcut = 0.05, N = 100, splitter =
"hc", modeler = "logic")
registerPredictor(tag, description, modeler)
availablePredictors()
```

Arguments

data	A data matrix where (numerical) features are rows and samples are columns.
metric	A distance metric. This metric should be one of the values supported by either the distanceMatrix function from the ClassDiscovery package or the binaryDistance function from the Mercator package. All metrics computed by the dist function are included.
label	Character use to mark the root note. (default is "H" for "head".)
pcut	Significance cutoff ("alpha") on the empirical p-value.
splitter	A tag that uniquely identifies one of the splitting functions that have been registered. Built-in registered "splitters", with their tags, include: (hc) agglomeratve hierarchical clustering; (km) k-means clustering; (dv) divisive heirarchical clustering, and (ap) affinity propagation clustering.
modeler	A tag that uniquely identifies one of the predictor algorithms that have been registered. Built-in registered "predictors", with their tags, include: (logic) logic regression; (svm) support vector machines; (lr) logitic regression, (rf) random forests, and (tr) tail rank.
N	Number of random splits used to estimate emoirical p-value.
tag	A character string representing a short abbreviation used to identify and eventually call the associated "predictor" method that can dichtomize the data.
description	A character string containing the full name of the prediction algorithm. Potentially, this can be used in data summaries or plots.

Details

The core idea of this package is to construct decision trees that simulataneouly learn how to cluster the items in a data set while also learning a classifier that can predict the cluster assignments of new data. We support four built-in clustering algortihms to learn how to split a single data set into two parts:

hc agglomerative hierarchical clustering as implemented in the hclust function, followed by cutree.

km kmeans clustering with k = 2.

dv divisive hierarchical clustering as implemented in the diana function.

ap affinity propagation clustering, as implemented in the apcluster function.

We also support multiple classification algorithms to learna model to predict the binary split at the node:

logic A modeler that implemts the "logic regression" approach to combine binary predictprs using logical operations, as implemented in the logicFS package.

svm Support vector machines.

lr The traditional logistic regression approach.

rf Random forests.

tr The "tail-rank" algorithm.

To get a list of the available methods, use the functions availableSplitters and availablePredictors. You can add your own algorithm with the functions registerSplitter pr registerPredictor. A "splitter" should accept a distance matrix as input and return a clustering vector with labels "1" or "2" for the classes. A "predictpr", by contrast, should be an object of the Modeler-class.

Value

Returns a BinaryNode object, which includes a

Author(s)

Kevin R. Coombes <krc@silicovore.com>

Examples

```
nr <- 200 # features
nc <- 60  # samples
set.seed(80123)
comat <- matrix(rnorm(nr*nc, 0, 1), nrow = nr)
dimnames(comat) <- list(paste0("F", 1:nr),
                        paste0("S", 1:nc))
splay <- rep(c(2, -2), each = nc/2)
for(J in 1:30) comat[J, ] <- comat[J, ] + splay
C.hc <- findSplit(comat, splitter = "hc")
evalSplit(C.hc, comat, "euclid", "sw")
```

---

learnConstant *Fit models and make predictions with a constant classifier*

Description

These functions are used to apply the generic train-and-test mechanism to a classifier that simply returns the same constant value for all inputs.

Usage

```
learnConstant(data, status, params, predfun)
predictConstant(newdata, details, status, ...)
makeLeaf(value, status = factor(c("L", "R")))
```

Arguments

data	The data matrix, with rows as features ("genes") and columns as the samples to be classified.
status	A factor, with two levels, classifying the samples. The length must equal the number of data columns.
params	A list of additional parameters used by the classifier; see Details.
predfun	The function used to make predictions on new data, using the trained classifier. Should always be set to predictConstant.
newdata	Another data matrix, with the same number of rows as data.
details	A list of additional parameters describing details about the particular classifier; see Details.
...	Optional extra parameters required by the generic "predict" method.
value	The constant value to be returned by the predictor. This should be one of the levels of the status vairable.

Details

The input arguments to both `learnConstant` and `predictConstant` are dictated by the requirements of the general train-and-test mechanism provided by the `Modeler-class`.

The "Constant" classifier is designed for use in the decision trees that are part of this package. They will be used as the "predictors" in leaf nodes, which always make the same prediction on any samples that make their way down to that node.

We provied the `makeLeaf` fnction as a convenience to create constant classifiers, by specifying their preferred value. (This value could also be provided as an extra/optional pafameter to the `Modeler` constructor.) For must uses, it is easier to simply use `makeLeaf` and ignore the `learnConstant` and `predictConstant`. The result of fitting the model using `learnConstant` or `makeLeaf` is a member of the `FittedModel-class`.

Value

The `learnConstant` and the `makeLeaf` functions return an object of the `FittedModel-class`, representing a Constant classifier that has been fitted on a training `data` set.

The `predictConstant` function returns a factor containing the predictions of the model when applied to the new data set, of length equal tothe number of columns (samples).

Author(s)

Kevin R. Coombes <krc@silicovore.com>

See Also

See `Modeler-class` and `Modeler` for details about how to train and test models. See `FittedModel-class` and `FittedModel` for details about the structure of the object returned by `learnConstant`.

Examples

```
# simulate some data
data <- matrix(rnorm(100*20), ncol=20)
status <- factor(rep(c("A", "B"), each=10))

# create a constnt modeler.
fm <- makeLeaf("A", factor(c("A", "B")))

# Make predictions on some new simulated data
predictConstant(data, fm@details, status)
```

`learnLogic` *Fit models and make predictions with a logic regression classifier*

Description

These functions are used to apply the generic train-and-test mechanism to a classifier that uses the concept of logic regression.

Usage

```
learnLogic(data, status, params, pfun, debug = FALSE)
predictLogic(newdata, details, status, debug = FALSE, ...)
```

Arguments

data	The data matrix, with rows as features ("genes") and columns as the samples to be classified.
status	A factor, with two levels, classifying the samples. The length must equal the number of data columns.
params	A list of additional parameters used by the classifier; see Details.
pfun	The function used to make predictions on new data, using the trained classifier. Should always be set to predictLogic.
debug	A logical value; should debug information be printer?
newdata	Another data matrix, with the same number of rows as data, and in the same order.
details	A list of additional parameters describing details about the particular classifier; see Details.
...	Optional extra parameters required by the generic "predict" method.

Details

The input arguments to both learnLogic and predictLogic are dictated by the requirements of the general train-and-test mechanism provided by the Modeler-class.

The Logic classifier is similar to the logistic regression classifiers already included in the Modeler package. The crucial difference is that the logic regression model assumes that all predictors are binary variables (thought of as logical vectors) and that logical combinations (based on and, or, and not operators) of predictors shouod also be considered. Internally, if the trainng data consist of continuous data, we use the bimodalIndex function to dichotomize the trainng data. The same cutpoints are later used to dichotomize the test data in the predict function.

Feature selection is handled by the bagging routines in the logicFS package. The default call to logic.bagging sets specific values for parameters B=20, nleaves=10, rand=54321 and for logreg.anneal.control. If you want to add additional parameters, you can use the params parameter in learnLogic.

The result of fitting the model using learnLogic is a member of the FittedModel-class. In additon to storing the prediction function (pfun) and the training data and status, the FittedModel stores those details about the model that are required in order to make predictions of the outcome on new data. The details object is appropriate for sending as the second argument to the predictLogic function in order to make predictions with the model on new data. Note that the status vector here is the one used for the *training* data, since the prediction function only uses the *levels* of this factor to make sure that the direction of the predictions is interpreted correctly.

Value

The learnLogic function returns an object of the FittedModel-class, representing a Logic classifier that has been fitted on a training data set.

The predictLogic function returns a factor containing the predictions of the model when applied to the new data set.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

See Also

See Modeler-class and Modeler for details about how to train and test models. See FittedModel-class and FittedModel for details about the structure of the object returned by learnLogic.

Examples

```
nr <- 100 # features
nc <- 40  # samples
set.seed(97531)
bimat <- matrix(rbinom(nr*nc, 1, 0.45), nrow = nr)
dimnames(bimat) <- list(paste0("B", 1:nr),
                        paste0("S", 1:nc))
stat <- rbinom(nc, 1, 0.37)

myMod <- learn(logicModeler, bimat, stat)
table(predict(myMod), stat) # on the diagonal
```

---

splitters *Bimodal Splitting of Data Sets*

Description

Functions to split a data set into two parts.

Usage

```
registerSplitter(tag, description, FUN)
availableSplitters()
findSplit(data, metric = "euclidean", splitter = names(splitters),
  LR = c("L", "R"))
evalSplit(split, data, metric, tool = c("sw", "ssq"), N = 100)
```

Arguments

tag	A character string representing a short abbreviation used to identify and eventually call the associated "splitter" functoin that can dichtomize the data.
description	A character string containing the full name of the splitting algorithm. Potentially, this can be used in data summaries or plots.
FUN	A "splitter"; that is, a function that takes a distance matrix as input and returns a numeric vector that assigns samples in the data set to one of two classes labeled "1" or "2".
data	The data set to be split into two parts.
metric	A distance metric. This metric should be one of the values supported by the distanceMatrix fuction from the ClassDiscovery package. All metrics computed by the dist function are supported.
splitter	A tag that uniquely identifies one of the splitting functions that have been registered. Built-in registered "splitters", with their tags, include: (hc) agglomeratve hierarchical clustering; (km) k-means clustering; (dv) divisive heirarchical clustering, and (ap) affinity propagation clustering.

LR	A character vector of length two denoting the laels to be used for the two classes after splitting the data.
split	A clustering factor, typically the output of a call to the findSplit fuction.
tool	The tool/splitter to be used to evalaute the quality of the split. Built-in splitters include the mean silhouette width (sw) or the within-group sum of square distances (ssq).
N	Number of random splits used to estimate emoirical p-value.

Details

The core idea of this package is to construct decision trees that simulataneouly learn how to cluster the items in a data set while also learning a classifier that can predict the cluster assignments of new data. We support four built-in clustering algortihms to learn how to split a single data set into two parts:

hc agglomerative hierarchical clustering as implemented in the hclust function, followed by cutree.

km kmeans clustering with k = 2.

dv divisive hierarchical clustering as implemented in the diana function.

ap affinity propagation clustering, as implemented in the apcluster function.

Value

The registerSplitter funciton invisibly returns the tag assigned to the new splitter.

The findSplit function returns a two-level factor with length equal to the number of columns (samples) in the data set.

The evalSplit returns a list of length two, containing a statistic (either the men silhouette width or the within-group sum of square distances, depending on the tool being used) along with an estimated empirical pvalue.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

Examples

```
nr <- 200 # features
nc <- 60  # samples
set.seed(80123)
comat <- matrix(rnorm(nr*nc, 0, 1), nrow = nr)
dimnames(comat) <- list(paste0("F", 1:nr),
                        paste0("S", 1:nc))
splay <- rep(c(2, -2), each = nc/2)
for(J in 1:30) comat[J, ] <- comat[J, ] + splay
C.hc <- findSplit(comat, splitter = "hc")
evalSplit(C.hc, comat, "euclid", "sw")
```

Index