Package 'copulaboost'

October 12, 2022

Type Package

Title Fitting Additive Copula Regression Models for Binary Outcome Regression

Version 0.1.0

Maintainer Simon Boge Brant <simbrant91@gmail.com>

Description Additive copula regression for regression problems with binary outcome via gradient boosting [Brant, Hobæk Haff (2022); <arXiv:2208.04669>]. The fitting process includes a specialised model selection algorithm for each component, where each component is found (by greedy optimisation) among all the D-vines with only Gaussian pair-copulas of a fixed dimension, as specified by the user. When the variables and structure have been selected, the algorithm then re-fits the component where the pair-copula distributions can be different from Gaussian, if specified.

License MIT + file LICENCE

Imports rvinecopulib (>= 0.5.4.1.0)

Encoding UTF-8

RoxygenNote 7.2.0

NeedsCompilation no

Author Simon Boge Brant [aut, cre] (<https://orcid.org/0000-0001-8049-7069>), Ingrid Hobæk Haff [aut]

Repository CRAN

Date/Publication 2022-08-23 08:30:11 UTC

R topics documented:

`copulaboost` *copulaboost*

Description

This is the main function of the package, which fits an additive model with a fixed number of components, each involving a fixed number of covariates, where each component is a copula regression model.

Usage

```
copulaboost(
  y,
  x,
  cov_types,
  n_models = 100,
  n_covs = 5,
  learning_rate = 0.33,
  eps = 0.05,
  verbose = FALSE,
  cont_method = "Localmedian",
  family_set = c("gaussian", "clayton", "gumbel"),
  jitter_sel = TRUE,
  ml_update = FALSE,
  ml_sel = FALSE,
  max_ml_scale = 1,
  keep_sel_struct = TRUE,
  approx_order = 2,
  parametric_margs = TRUE,
  parallel = FALSE,
  par_method_sel = "itau",
  update_intercept = TRUE,
  model = NULL,
  xtreme = FALSE
)
```

Arguments

`y` A vector of n observations of the (univariate) binary outcome variable y

`x` A (n x p) matrix of n observations of p covariates

`cov_types` A vector of p characters that have to take the value "c" or "d" to indicate whether each margin of the covariates is discrete or continuous.

`n_models` The number of model components to fit.

`n_covs` The number of covariates included in each component.

`learning_rate` Factor to scale (down) the each component.

`eps` Control parameter for the approximation to the conditional expectation (the prediction) for each copula model (component), which splits the interval [-1, 1] into equal pieces of eps length.

`verbose` Logical indicator of whether a progressbar should be shown in the terminal.

`cont_method` Method to use for the approximation of each conditional expectation, can either be "Localmedian" or "Trapezoidalsurv", for the former, see section 3.2 of https://arxiv.org/ftp/arxiv/papers/2208/2208.04669.pdf. The latter uses the so called "Darth vader rule" in conjuction with a simple translative transformation to write the conditional expectation as an integral along the conditional survival function, which is then approximated by the trapezoidal method.

`family_set` A vector of strings that specifies the set of pair-copula families that the fitting algorithm chooses from. For an overview of which values that can be specified, see the documentation for bicop.

`jitter_sel` Logical indicator of whether jittering should be used for any discrete covariates when selecting the variables for each component (improves computational speed).

`ml_update` Logical indicator of whether each new component should be scaled by a number between 0 and max_ml_scale by maximising the log-likelihood of the scaling factor given the current model and the new component.

`ml_sel` The same as ml_update, but for the variable selection algorithm.

`max_ml_scale` The maximum scaling factor allowed for each component.

`keep_sel_struct`

Logical indicator of whether the d-vine structures found by the model selection algorithm should be kept when fitting the components.

`approx_order` The order of the approximation used for evaluating the conditional expectations when selecting covariates for each component. The allowed values for approx_order are 1, 2, 3, 4, 5, and 6.

`parametric_margs`

Logical indicator of whether parametric (gaussian or bernoulli) models should be used for the marginal distributions of the covariates.

`parallel` (Experimental) Logical indicator of whether parallelization should be used when selecting covariates.

`par_method_sel` Estimation method for copulas used when selecting the model components, either "itau" or "mle", see the documentation for bicop.

`update_intercept`

Logical indicator of whether the intercept parameter should be updated (by univariate maximum likelihood) after each component is added.

`model` Initial copulaboost-model. If model is a copulaboost model with k components, the resulting model will have k + n_models components.

`xtreme` (Experimental) Logical indicator of whether a second order expansion of the log-likelihood should be used in each gradient boosting step, similar to the xgboost algorithm.

Value

A copulaboost object, which contains a nested list ’object$model’ which contains all of the model components. The first element of each list contains a copulareg object, and the second element contains a vector listing the indexes of the covariates that are a part of the component. The object also contains a list of the updated intercepts ’object$f0_updated’ at each stage of the fitting process, so that the j-th intercept is the intercept for the model that is the weighted sum of the j first components. ’object$scaling’ contains a vector of weights for each components, equal to the learning rate, possibly multiplied by an individual factor if ml_update = TRUE. In addition the object contains the values of the arguments learning_rate, cov_types, and eps that where used when calling copulaboost().

Examples

```
# Compile some test data
data('ChickWeight')
set.seed(10)
tr <- sample(c(TRUE, FALSE), nrow(ChickWeight), TRUE, c(0.7, 0.3))
y_tr <- as.numeric(ChickWeight$weight[tr] > 100)
y_te <- as.numeric(ChickWeight$weight[!tr] > 100)
x_tr <- apply(ChickWeight[tr, -1], 2, as.numeric)
x_te <- apply(ChickWeight[!tr, -1], 2, as.numeric)
cov_types <- apply(x_tr, 2,
                   function(x) if(length(unique(x)) < 10) ”d” else ”c”)

# Fit model to training data
md <- copulaboost::copulaboost(y_tr, x_tr, cov_types, n_covs = 2,
                               n_models = 5, verbose = TRUE)

# Out of sample predictions for a new data matrix
preds <- predict(md, new_x = x_te, all_parts = TRUE)

# Plot log-likelihood
plot(apply(preds, 2,
           function(eta) {
             sum(stats::dbinom(y_te, 1, stats::plogis(eta), log = TRUE))
             }),
     type = ”s”)
```

copulareg *copulareg*

Description

This function fits joint distributions with an R-vine pair copula structure, that is constructed in a specific way so that the conditional density and distribution of the variable y can be computed explicitly.

Usage

```
copulareg(
  y,
  x,
  var_type_y,
  var_type_x,
  distr_x = NULL,
  distr_y = NULL,
  dvine = FALSE,
  family_set = c(”gaussian”, ”clayton”, ”gumbel”)
)
```

Arguments

`y` A vector of n observations of the (univariate) outcome variable y

`x` A (n x p) matrix of n observations of p covariates

`var_type_y` A character that has to be specified as "d" or "c" to indicate whether y is discrete or continuous, respectively.

`var_type_x` A vector of p characters that have to take the value "c" or "d" to indicate whether each margin of the covariates is discrete or continuous.

`distr_x` Internally created object that contains a nested list. The first element of the ’outer’ list is a list where each element is a distribution object similar to that created by ecdf, the second element is a function ’transform’ that takes a matrix of values of x, and returns the corresponding cumulative distributions F(x).

`distr_y` Similar to distr_x, but for the outcome y.

`dvine` Logical variable indicating whether the function should fit a canonical d-vine to (y, x) with the ordering (x_1, ..., x_p, y).

`family_set` A vector of strings that specifies the set of pair-copula families that the fitting algorithm chooses from. For an overview of which values that can be specified, see the documentation for bicop.

Value

A copulareg object. Consists of ’model’, an rvinecopulib ’vinecop’ object for the copula-model for (x_1, ..., x_p, y), a hash table containing all of the conditionals for the model (for the training set), objects distr_x and distr_y that contain the marginal distributions of the covariates and the outcome, and y, the y-values for the training data.

`predict.copulaboost` *predict.copulaboost*

Description

Function that computes predictions for a copulaboost model.

Usage

```
## S3 method for class 'copulaboost'
predict(
  object,
  new_x = NULL,
  verbose = FALSE,
  all_parts = FALSE,
  impute_na = TRUE,
  ...
)
```

Arguments

`object` A copulaboost model object returned from the copulaboost function.

`new_x` A matrix of covariate values to compute predictions for. If new_x is not provided, the function will return predictions for the data used for fitting the model.

`verbose` A logical indicator of whether a progressbar should be shown in the terminal.

`all_parts` A logical indicator of whether predictions for the k first components for k = 1, ..., n_models should be returned as a matrix. If all_parts = FALSE, only the prediction with all of the model components is returned.

`impute_na` A logical indicator of whether any potential NA values from any of the predictions from each component should be replaced by the median prediction for that component.

`...` further arguments passed to or from other methods.

Value

If all_parts=FALSE this function will return a list of predictions for each row of new_x (if specified, otherwise predictions for the training data will be returned). If all_parts=TRUE a matrix will be returned, where the j-th column contains predictions using the j first model components.

`predict.copulareg` *predict.copulareg*

Description

Computes predictions based on a fitted copulareg model.

Usage

```
## S3 method for class 'copulareg'
predict(object, new_x = NULL, eps = 0.01, cont_method = ”Localmedian”, ...)
```

Arguments

`object` Model fit as returned by copulareg

`new_x` optional matrix of covariate values to compute the predicted values of the outcome for. If not specified, the predicted values for the training sample is returned.

`eps` Interval between each interpolation point when integrating to evaluate the predicted value of y, in the case where y is continuous. If y is discrete this parameter is ignored.

`cont_method` Specifies the method used to compute the expected values. Can be specified as 'Localmedian' or 'Trapezoidalsurv'. The first method divides the range of the observed values of y into subintervals according to the argument 'eps', where the sub-integral is approximated as the measure of the interval weighted by the local median on the interval. The second method computes the integral by integrating the survival function using the trapezoidal rule, by transforming the outcome into a positive variable by adding a constant.

`...` further arguments passed to or from other methods.

Value

A list of predicted values for each row of new_x, if specified, otherwise, predictions for each row of the training data is returned.

Index