Package: agtboost (via r-universe)

September 8, 2024

Type Package

Title Adaptive and Automatic Gradient Boosting Computations

Version 0.9.3

Date 2021-11-23

Author Berent Ånund Strømnes Lunde

Maintainer Berent Ånund Strømnes Lunde <lundeberent@gmail.com>

Description Fast and automatic gradient tree boosting designed to avoid manual tuning and cross-validation by utilizing an information theoretic approach. This makes the algorithm adaptive to the dataset at hand; it is completely automatic, and with minimal worries of overfitting. Consequently, the speed-ups relative to state-of-the-art implementations can be in the thousands while mathematical and technical knowledge required on the user are minimized.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.6.0)

Imports methods, Rcpp (>= 1.0.1)

LinkingTo Rcpp, RcppEigen

RcppModules aGTBModule

RoxygenNote 7.1.2

Suggests testthat

NeedsCompilation yes

Date/Publication 2021-11-23 21:10:02 UTC

Repository https://blunde1.r-universe.dev

RemoteUrl https://github.com/cran/agtboost

RemoteRef HEAD

RemoteSha 742d340ce2bf75b9d2235978d9a500a1cff88ae3

Contents

`agtboost` *Adaptive and automatic gradient boosting computations.*

Description

Adaptive and Automatic Gradient Boosting Computations

Details

agtboost is a lightning fast gradient boosting library designed to avoid manual tuning and cross-validation by utilizing an information theoretic approach. This makes the algorithm adaptive to the dataset at hand; it is completely automatic, and with minimal worries of overfitting. Consequently, the speed-ups relative to state-of-the-art implementations are in the thousands while mathematical and technical knowledge required on the user are minimized.

Important functions:

- `gbt.train`: function for training an **agtboost** ensemble
- `predict.Rcpp_ENSEMBLE`: function for predicting from an **agtboost** ensemble

See individual function documentation for usage.

Author(s)

Berent Ånund Strømnes Lunde

`caravan.train` *The Insurance Company (TIC) Benchmark*

Description

`caravan.train` and `caravan.test` both contain a design matrix with 85 columns and a response vector. The train set consists of 70% of the data, with 4075 rows. The test set consists of the remaining 30% with 1747 rows. The following references the documentation within the **ISLR** package: The original data contains 5822 real customer records. Each record consists of 86 variables, containing sociodemographic data (variables 1-43) and product ownership (variables 44-86). The sociodemographic data is derived from zip codes. All customers living in areas with the same zip code have the same sociodemographic attributes. Variable 86 (`Purchase`) indicates whether the customer purchased a caravan insurance policy. Further information on the individual variables can be obtained at http://www.liacs.nl/~putten/library/cc2000/data.html

Usage

```
caravan.train; caravan.test
```

Format

Lists with a design matrix `x` and response `y`

Source

The data was originally supplied by Sentient Machine Research and was used in the CoIL Challenge 2000.

References

P. van der Putten and M. van Someren (eds) . CoIL Challenge 2000: The Insurance Company Case. Published by Sentient Machine Research, Amsterdam. Also a Leiden Institute of Advanced Computer Science Technical Report 2000-09. June 22, 2000. See http://www.liacs.nl/~putten/library/cc2000/

P. van der Putten and M. van Someren. A Bias-Variance Analysis of a Real World Learning Problem: The CoIL Challenge 2000. Machine Learning, October 2004, vol. 57, iss. 1-2, pp. 177-195, Kluwer Academic Publishers

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013) *An Introduction to Statistical Learning with applications in R*, https://trevorhastie.github.io/ISLR/, Springer-Verlag, New York

Examples

```
summary(caravan.train)
summary(caravan.test)
```

gbt.complexity *Return complexity of model in terms of hyperparameters.*

Description

gbt.complexity creates a list of hyperparameters from a model

Usage

```
gbt.complexity(model, type)
```

Arguments

model	object or pointer to object of class ENSEMBLE
type	currently supports "xgboost" or "lightgbm"

Details

Returns the complexity of model in terms of hyperparameters associated to model type.

Value

list with type hyperparameters.

Examples

```
set.seed(123)
library(agtboost)
n <- 10000
xtr <- as.matrix(runif(n, 0, 4))
ytr <- rnorm(n, xtr, 1)
xte <- as.matrix(runif(n, 0, 4))
yte <- rnorm(n, xte, 1)
model <- gbt.train(ytr, xtr, learning_rate = 0.1)
gbt.complexity(model, type="xgboost")
gbt.complexity(model, type="lightgbm")
## See demo(topic="gbt-complexity", package="agtboost")
```

gbt.convergence *Convergence of agtboost model.*

Description

gbt.convergence calculates loss of data over iterations in the model

Usage

```
gbt.convergence(object, y, x)
```

Arguments

object	Object or pointer to object of class ENSEMBLE
y	response vector
x	design matrix for training. Must be of type matrix.

Details

Computes the loss on supplied data at each boosting iterations of the model passed as object. This may be used to visually test for overfitting on test data, or the converce, to check for underfitting or non-convergence.

Value

vector with $K+1$ elements with loss at each boosting iteration and at the first constant prediction

Examples

```
## Gaussian regression:
x_tr <- as.matrix(runif(500, 0, 4))
y_tr <- rnorm(500, x_tr, 1)
x_te <- as.matrix(runif(500, 0, 4))
y_te <- rnorm(500, x_te, 1)
mod <- gbt.train(y_tr, x_tr)
convergence <- gbt.convergence(mod, y_te, x_te)
which.min(convergence) # Should be fairly similar to boosting iterations + 1
mod$get_num_trees() +1 # num_trees does not include initial prediction
```

gbt.importance *Importance of features in a model.*

Description

gbt.importance creates a data.frame of feature importance in a model

Usage

```
gbt.importance(feature_names, object)
```

Arguments

feature_names	character vector of feature names
object	object or pointer to object of class ENSEMBLE

Details

Sums up "expected reduction" in generalization loss (scaled using learning_rate) at each node for each tree in the model, and attributes it to the feature the node is split on. Returns result in terms of percents.

Value

data.frame with percentwise reduction in loss of total attributed to each feature.

Examples

```
## Load data
data(caravan.train, package = "agtboost")
train <- caravan.train
mod <- gbt.train(train$y, train$x, loss_function = "logloss", verbose=10)
feature_names <- colnames(train$x)
imp <- gbt.importance(feature_names, mod)
imp
```

gbt.ksval *Kolmogorov-Smirnov validation of model*

Description

gbt.ksval transforms observations to U(0,1) if the model is correct and performs a Kolmogorov-Smirnov test for uniformity.

Usage

```
gbt.ksval(object, y, x)
```

Arguments

object	Object or pointer to object of class ENSEMBLE
y	Observations to be tested
x	design matrix for training. Must be of type matrix.

Details

Model validation of model passed as object using observations y. Assuming the loss is a negative log-likelihood and thus a probabilistic model, the transformation

$$u = F_Y(y; x, \theta) \sim U(0, 1),$$

is usually valid. One parameter, $\mu = g^{-1}(f(x))$, is given by the model. Remaining parameters are estimated globally over feature space, assuming they are constant. This then allow the above transformation to be exploited, so that the Kolmogorov-Smirnov test for uniformity can be performed.

If the response is a count model (poisson or negbinom), the transformation

$$u_i = F_Y(y_i - 1; x, \theta) + U f_Y(y_i, x, \theta),\ U \sim U(0, 1)$$

is used to obtain a continuous transformation to the unit interval, which, if the model is correct, will give standard uniform random variables.

Value

Kolmogorov-Smirnov test of model

Examples

```
## Gaussian regression:
x_tr <- as.matrix(runif(500, 0, 4))
y_tr <- rnorm(500, x_tr, 1)
x_te <- as.matrix(runif(500, 0, 4))
y_te <- rnorm(500, x_te, 1)
mod <- gbt.train(y_tr, x_tr)
gbt.ksval(mod, y_te, x_te)
```

gbt.load *Load an aGTBoost Model*

Description

gbt.load is an interface for loading a **agtboost** model.

Usage

```
gbt.load(file)
```

Arguments

file	Valid file-path to a stored aGTBoost model

Details

The load function for **agtboost**. Loades a GTB model from a txt file.

Value

Trained aGTBoost model.

See Also

gbt.save

gbt.save *Save an aGTBoost Model*

Description

gbt.save is an interface for storing a **agtboost** model.

Usage

```
gbt.save(gbt_model, file)
```

Arguments

gbt_model	Model object or pointer to object of class ENSEMBLE
file	Valid file-path

Details

The model-storage function for **agtboost**. Saves a GTB model as a txt file. Might be retrieved using gbt.load

Value

Txt file that can be loaded using `gbt.load`.

See Also

`gbt.load`

`gbt.train` *aGTBoost Training.*

Description

`gbt.train` is an interface for training an **agtboost** model.

Usage

```
gbt.train(
  y,
  x,
  learning_rate = 0.01,
  loss_function = "mse",
  nrounds = 50000,
  verbose = 0,
  gsub_compare,
  algorithm = "global_subset",
  previous_pred = NULL,
  weights = NULL,
  force_continued_learning = FALSE,
  offset = NULL,
  ...
)
```

Arguments

`y`	response vector for training. Must correspond to the design matrix `x`.
`x`	design matrix for training. Must be of type `matrix`.
`learning_rate`	control the learning rate: scale the contribution of each tree by a factor of `0 < learning_rate < 1` when it is added to the current approximation. Lower value for `learning_rate` implies an increase in the number of boosting iterations: low `learning_rate` value means model more robust to overfitting but slower to compute. Default: 0.01
`loss_function`	specify the learning objective (loss function). Only pre-specified loss functions are currently supported. • `mse` regression with squared error loss (Default). • `logloss` logistic regression for binary classification, output score before logistic transformation.

- poisson Poisson regression for count data using a log-link, output score before natural transformation.
- gamma::neginv gamma regression using the canonical negative inverse link. Scaling independent of y.
- gamma::log gamma regression using the log-link. Constant information parametrisation.
- negbinom Negative binomial regression for count data with overdispersion. Log-link.
- count::auto Chooses automatically between Poisson or negative binomial regression.

nrounds — a just-in-case max number of boosting iterations. Default: 50000

verbose — Enable boosting tracing information at i-th iteration? Default: 0.

gsub_compare — Deprecated. Boolean: Global-subset comparisons. FALSE means standard GTB, TRUE compare subset-splits with global splits (next root split). Default: TRUE.

algorithm — specify the algorithm used for gradient tree boosting.

- vanilla ordinary gradient tree boosting. Trees are optimized as if they were the last tree.
- global_subset function-change to target maximized reduction in generalization loss for individual datapoints

previous_pred — prediction vector for training. Boosted training given predictions from another model.

weights — weights vector for scaling contributions of individual observations. Default NULL (the unit vector).

force_continued_learning — Boolean: FALSE (default) stops at information stopping criterion, TRUE stops at nround iterations.

offset — add offset to the model g(mu) = offset + F(x).

... — additional parameters passed.

- if loss_function is 'negbinom', dispersion must be provided in ...

Details

These are the training functions for an **agtboost**.

Explain the philosophy and the algorithm and a little math

gbt.train learn trees with adaptive complexity given by an information criterion, until the same (but scaled) information criterion tells the algorithm to stop. The data used for training at each boosting iteration stems from a second order Taylor expansion to the loss function, evaluated at predictions given by ensemble at the previous boosting iteration.

Value

An object of class ENSEMBLE with some or all of the following elements:

- handle a handle (pointer) to the **agtboost** model in memory.

- `initialPred` a field containing the initial prediction of the ensemble.
- `set_param` function for changing the parameters of the ensemble.
- `train` function for re-training (or from scratch) the ensemble directly on vector `y` and design matrix `x`.
- `predict` function for predicting observations given a design matrix
- `predict2` function as above, but takes a parameter max number of boosting ensemble iterations.
- `estimate_generalization_loss` function for calculating the (approximate) optimism of the ensemble.
- `get_num_trees` function returning the number of trees in the ensemble.

References

Berent Ånund Strømnes Lunde, Tore Selland Kleppe and Hans Julius Skaug, "An Information Criterion for Automatic Gradient Tree Boosting", 2020, https://arxiv.org/abs/2008.05926

See Also

predict.Rcpp_ENSEMBLE

Examples

```
## A simple gtb.train example with linear regression:
x <- runif(500, 0, 4)
y <- rnorm(500, x, 1)
x.test <- runif(500, 0, 4)
y.test <- rnorm(500, x.test, 1)

mod <- gbt.train(y, as.matrix(x))
y.pred <- predict( mod, as.matrix( x.test ) )

plot(x.test, y.test)
points(x.test, y.pred, col="red")
```

---

predict.Rcpp_ENSEMBLE *aGTBoost Prediction*

Description

`predict` is an interface for predicting from a **agtboost** model.

Usage

```
## S3 method for class 'Rcpp_ENSEMBLE'
predict(object, newdata, ...)
```

Arguments

`object`	Object or pointer to object of class `ENSEMBLE`
`newdata`	Design matrix of data to be predicted. Type `matrix`
`...`	additional parameters passed. Currently not in use.

Details

The prediction function for **agtboost**. Using the generic `predict` function in R is also possible, using the same arguments.

Value

For regression or binary classification, it returns a vector of length `nrows(newdata)`.

References

Berent Ånund Strømnes Lunde, Tore Selland Kleppe and Hans Julius Skaug, "An Information Criterion for Automatic Gradient Tree Boosting", 2020, https://arxiv.org/abs/2008.05926

See Also

`gbt.train`

Examples

```
## A simple gtb.train example with linear regression:
x <- runif(500, 0, 4)
y <- rnorm(500, x, 1)
x.test <- runif(500, 0, 4)
y.test <- rnorm(500, x.test, 1)

mod <- gbt.train(y, as.matrix(x))

## predict is overloaded
y.pred <- predict( mod, as.matrix( x.test ) )

plot(x.test, y.test)
points(x.test, y.pred, col="red")
```

predict.Rcpp_GBT_COUNT_AUTO

aGTBoost Count-Regression Auto Prediction

Description

predict is an interface for predicting from a agtboost model.

Usage

```
## S3 method for class 'Rcpp_GBT_COUNT_AUTO'
predict(object, newdata, ...)
```

Arguments

object	Object or pointer to object of class GBT_ZI_MIX
newdata	Design matrix of data to be predicted. Type matrix
...	additional parameters passed. Currently not in use.

Details

The prediction function for agtboost. Using the generic predict function in R is also possible, using the same arguments.

Value

For regression or binary classification, it returns a vector of length nrows(newdata).

References

Berent Ånund Strømnes Lunde, Tore Selland Kleppe and Hans Julius Skaug, "An Information Criterion for Automatic Gradient Tree Boosting", 2020, https://arxiv.org/abs/2008.05926

See Also

gbt.train

Examples

```
## A simple gtb.train example with linear regression:
## Random generation of zero-inflated poisson
2+2
```

Index