Package ‘irboost’

February 16, 2022

Type Package
Title Iteratively Reweighted Boosting for Robust Analysis
Version 0.1-1.1
Date 2022-02-15
Author Zhu Wang [aut, cre] (<https://orcid.org/0000-0002-0773-0052>)
Maintainer Zhu Wang <zhuwang@gmail.com>
Description Fit a predictive model with the iteratively reweighted boosting (IRBoost) that mini-
mizes the robust loss functions in the CC-family (concave-convex). The convex optimiza-
tion is conducted by functional descent boosting algorithm in the R package xgboost. The IR-
Boost reduces the weight of the observation that leads to a large loss; it also pro-
vides weights to help identify outliers. Applications include the robust generalized linear mod-
els and extensions, where the mean is related to the predictors by boosting, and robust acceler-
ated failure time models. The package supersedes the R package cc-

Depends R (>= 3.5.0)
Imports mpath (>= 0.4-2.21), xgboost
Suggests R.rsp, DiagrammeR, survival, Hmisc
VignetteBuilder R.rsp
License GPL (>= 3)
Encoding UTF-8
LazyLoad yes
RoxygenNote 7.1.2
NeedsCompilation no
Repository CRAN
Date/Publication 2022-02-16 20:10:05 UTC

R topics documented:

dataLS ................................................................. 2
irboost ............................................................... 3
irboost_aft ......................................................... 6
dataLS

**Description**

generate random data for classification as in Long and Servedio (2010)

**Usage**

dataLS(ntr, ntu = ntr, nte, percon)

**Arguments**

- `ntr` number of training data
- `ntu` number of tuning data, default is the same as `ntr`
- `nte` number of test data
- `percon` proportion of contamination, must between 0 and 1. If `percon > 0`, the labels of the corresponding perenrage of response variable in the training and tuning data are flipped.

**Value**

a list with elements xtr, xtu, xte, ytr, ytu, yte for predictors of disjoint training, tuning and test data, and response variable -1/1 of training, tuning and test data.

**Author(s)**

Zhu Wang
Maintainer: Zhu Wang <zhuwang@gmail.com>

**References**


**Examples**

dat <- dataLS(ntr=100, nte=100, percon=0)
**irboost**

*fit a robust predictive model with iteratively reweighted boosting algorithm*

**Description**

Fit a predictive model with the iteratively reweighted convex optimization (IRCO) that minimizes the robust loss functions in the CC-family (concave-convex). The convex optimization is conducted by functional descent boosting algorithm in the R package *xgboost*. The iteratively reweighted boosting (IRBoost) algorithm reduces the weight of the observation that leads to a large loss; it also provides weights to help identify outliers. Applications include the robust generalized linear models and extensions, where the mean is related to the predictors by boosting, and robust accelerated failure time models.

**Usage**

```r
irboost(
  x, y, weights, 
  cfun = "ccave", 
  s = 1, delta = 0.1, 
  dfun = "reg:squarederror", 
  iter = 10, 
  nrounds = 100, 
  del = 1e-10, 
  trace = FALSE, ...
)
```

**Arguments**

- **x**: input matrix, of dimension nobs x nvars; each row is an observation vector. Can accept *dgCMatrix*
- **y**: response variable. Quantitative for dfun="reg:squarederror", dfun="count:poisson" (non-negative counts) or dfun="reg:gamma" (positive). For dfun="binary:logitraw" or "binary:hinge", y should be a factor with two levels
- **weights**: vector of nobs with non-negative weights
- **s**: tuning parameter of cfun. s > 0 and can be equal to 0 for cfun="tcave". If s is too close to 0 for cfun="acave", "bcave", "ccave", the calculated weights can become 0 for all observations, thus crash the program
- **delta**: a small positive number provided by user only if cfun="gcave" and 0 < s < 1
dfun

type of convex component in the CC-family, the second C, or convex down, that's where the name dfun comes from. It is the same as objective in the xgboost package.

- **reg:squarederror** Regression with squared loss.
- **binary:logitraw** logistic regression for binary classification, predict linear predictor, not probabilities.
- **binary:hinge** hinge loss for binary classification. This makes predictions of -1 or 1, rather than producing probabilities.
- **multi:softprob** softmax function for multiclass problems. The result contains predicted probabilities of each data point in each class, say \( p_k \), \( k=0, ..., n\text{class}-1 \). Note, label is coded as in \([0, ..., n\text{class}-1]\). The loss function cross-entropy for the \( i \)-th observation is computed as \(-\log(p_k)\) with \( k=\text{label}_i, i=1, ..., n \).
- **count:poisson** Poisson regression for count data, predict mean of poisson distribution.
- **reg:gamma** gamma regression with log-link, predict mean of gamma distribution. The implementation in xgboost takes a parameterization in the exponential family:
  
exgboost/src/src/metric/elementwise_metric.cu.
In particularly, there is only one parameter psi and set to 1. The implementation of the IRCO algorithm follows this parameterization. See Table 2.1, McCullagh and Nelder, Generalized linear models, Chapman & Hall, 1989, second edition.
- **reg:tweedie** Tweedie regression with log-link. See also tweedie_variance_power in range: (1,2). A value close to 2 is like a gamma distribution. A value close to 1 is like a Poisson distribution.

**iter**
number of iteration in the IRCO algorithm

**nrounds**
boosting iterations within each IRCO iteration

**del**
convergency criteria in the IRCO algorithm, no relation to delta

**trace**
if TRUE, fitting progress is reported

... other arguments passing to xgboost

**Value**

An object with S3 class xgboost with the additional elements:

- **weight_update_log** a matrix of \( nobs \) row by iter column of observation weights in each iteration of the IRCO algorithm
- **weight_update** a vector of observation weights in the last IRCO iteration that produces the final model fit
- **loss_log** sum of loss value of the composite function \( \text{cfun}(\text{dfun}) \) in each IRCO iteration.
  
  Note, \( \text{cfun} \) requires \( \text{dfun} \) non-negative in some cases. Thus some \( \text{dfun} \) needs attentions. For instance, with \( \text{dfun} = \text{"reg:gamma"} \), the loss value is defined gamma-nloglik - (1+log(min(y)))). The second term is introduced such that the loss value is non-negative. In fact, gamma-nloglik=y/ypre + log(ypre) in the xgboost, where ypre is the mean prediction value, can be negative. It can be derived that for fixed y, the minimum value of gamma-nloglik is
achieved at \( y = \gamma \text{erf}^{-1}(y) \) or \( 1 + \log(y) \). Thus, among all \( y \) values, the minimum of gamma-nloglik is \( 1 + \log(\min(y)) \).

**Author(s)**

Zhu Wang  
Maintainer: Zhu Wang <zhuwang@gmail.com>

**References**


**Examples**

# regression, logistic regression, hinge regression, Poisson regression  
x <- matrix(rnorm(100*2),100,2)  
g2 <- sample(c(0,1),100,replace=TRUE)  
fit1 <- irboost(x, g2, cfun="acave", s=0.5, dfun="reg:squarederror", trace=TRUE,  
verbose=0, max.depth=1, nrounds=50)  
fit2 <- irboost(x, g2, cfun="acave", s=0.5, dfun="binary:logitraw", trace=TRUE,  
verbose=0, max.depth=1, nrounds=50)  
fit3 <- irboost(x, g2, cfun="acave", s=0.5, dfun="binary:hinge", trace=TRUE,  
verbose=0, max.depth=1, nrounds=50)  
fit4 <- irboost(x, g2, cfun="acave", s=0.5, dfun="count:poisson", trace=TRUE,  
verbose=0, max.depth=1, nrounds=50)

# Gamma regression  
x <- matrix(rnorm(100*2),100,2)  
g2 <- sample(rgamma(100, 1))  
library("xgboost")  
fit5 <- xgboost(x, g2, objective="reg:gamma", max.depth=1, nrounds=50)  
fit6 <- irboost(x, g2, cfun="acave", s=5, dfun="reg:gamma", trace=TRUE,  
verbose=0, max.depth=1, nrounds=50)  
plot(predict(fit5, x), predict(fit6, x))  
hist(fit6$weight_update)  
plot(fit6$loss_log)  
summary(fit6$weight_update)

# Tweedie regression  
fit6t <- irboost(x, g2, cfun="acave", s=5, dfun="reg:tweedie", trace=TRUE,  
verbose=0, max.depth=1, nrounds=50)  
# Gamma vs Tweedie regression  
hist(fit6$weight_update)  
hist(fit6t$weight_update)  
plot(predict(fit6t, x), predict(fit6t, x))

# multiclass classification in iris dataset:  
lb <- as.numeric(iris$Species)-1  
num_class <- 3  
set.seed(11)
# xgboost
bst <- xgboost(data=as.matrix(iris[, -5]), label=lb, max_depth=4, eta=0.5, nthread=2, nrounds=10, subsample=0.5, objective="multi:softprob", num_class=num_class)
# predict for softmax returns num_class probability numbers per case:
pred <- predict(bst, as.matrix(iris[, -5]))
# reshape it to a num_class-columns matrix
pred <- matrix(pred, ncol=num_class, byrow=TRUE)
# convert the probabilities to softmax labels
pred_labels <- max.col(pred)-1
# classification error
sum(pred_labels!=lb)/length(lb)

# irboost
fit7 <- irboost(x=as.matrix(iris[, -5]), y=lb, cfun="acave", s=50, dfun="multi:softprob", trace=TRUE, verbose=0, max.depth=4, eta=0.5, nthread=2, nrounds=10, subsample=0.5, num_class=num_class)
pred7 <- predict(fit7, as.matrix(iris[, -5]))
pred7 <- matrix(pred7, ncol=num_class, byrow=TRUE)
# convert the probabilities to softmax labels
pred7_labels <- max.col(pred7) - 1
# classification error: 0!
sum(pred7_labels != lb)/length(lb)
table(pred_labels, pred7_labels)
hist(fit6$weight_update)

---

**irboost_aft**

*fit a robust accelerated failure time model with iteratively reweighted boosting algorithm*

### Description

Fit an accelerated failure time model with the iteratively reweighted convex optimization (IRCO) that minimizes the robust loss functions in the CC-family (concave-convex). The convex optimization is conducted by functional descent boosting algorithm in the R package **xgboost**. The iteratively reweighted boosting (IRBoost) algorithm reduces the weight of the observation that leads to a large loss; it also provides weights to help identify outliers. For time-to-event data, an accelerated failure time model (AFT model) provides an alternative to the commonly used proportional hazards models. Note, **irboost** with dfun=survival:aft is the wrapper of **irboost_aft**, which was developed to facilitate a different data input format used in xgb.train not in xgboost at the time.

### Usage

```r
irboost_aft(
  params,
  data,
  cfun = "ccave",
```
irboost_aft

s = 1,
delta = 0.1,
iter = 10,
nrounds = 100,
del = 1e-10,
trace = FALSE,
...
)

Arguments

params the list of parameters used in xgb.train of xgboost. Must include aft_loss_distribution, aft_loss_distribution_scale, but there is no need to include objective. The complete list of parameters is available in the online documentation.
data training dataset. irboost_aft accepts only an xgb.DMatrix as the input.
s tuning parameter of cfun. s > 0 and can be equal to 0 for cfun="tcave". If s is too close to 0 for cfun="acave", "bcave", "ccave", the calculated weights can become 0 for all observations, thus crash the program.
delta a small positive number provided by user only if cfun="gcave" and 0 < s < 1
iter number of iteration in the IRCO algorithm
nrounds boosting iterations in xgb.train within each IRCO iteration
del convergencery criteria in the IRCO algorithm, no relation to delta
trace if TRUE, fitting progress is reported
...
other arguments passing to xgb.train

Value

An object of class xgb.Booster with additional elements:

- weight_update_log a matrix of nobs row by iter column of observation weights in each iteration of the IRCO algorithm
- weight_update a vector of observation weights in the last IRCO iteration that produces the final model fit
- loss_log sum of loss value of the composite function cfun(survival_aft_distribution) in each IRCO iteration

Author(s)

Zhu Wang
Maintainer: Zhu Wang <zhuwang@gmail.com>

References

See Also

`irboost`

Examples

```r
library("xgboost")
X <- matrix(1:5, ncol=1)

# Associate ranged labels with the data matrix.
# This example shows each kind of censored labels.
# uncensored right left interval
y_lower = c(10, 15, -Inf, 30, 100)
y_upper = c(Inf, Inf, 20, 50, Inf)
dtrain <- xgb.DMatrix(data=X, label_lower_bound=y_lower, label_upper_bound=y_upper)
  params = list(objective="survival:aft", aft_loss_distribution="normal",
                aft_loss_distribution_scale=1, max_depth=3, min_child_weight= 0)
watchlist <- list(train = dtrain)
bst <- xgb.train(params, dtrain, nrounds=15, watchlist=watchlist)
predict(bst, dtrain)
bst_cc <- irboost_aft(params, dtrain, nrounds=15, watchlist=watchlist, cfun="hcave",
                      s=1.5, trace=TRUE, verbose=0)
bst_cc$weight_update
predict(bst_cc, dtrain)
```
Index

* classification
  dataLS, 2
  irboost, 3

* regression
  irboost, 3
  irboost_aft, 6

* survival
  irboost_aft, 6

dataLS, 2

irboost, 3, 8
irboost_aft, 6