Package 'gbts'

October 17, 2016

Type Package

Title Hyperparameter Search for Gradient Boosted Trees

Version 1.0.1

Date 2016-10-14

Author Waley W. J. Liang

Maintainer Waley W. J. Liang <wliang10@gmail.com>

Description An implementation of hyperparameter optimization for Gradient Boosted Trees on binary classification and regression problems. The current version provides two optimization methods: Bayesian optimization and random search.

License GPL (>= 2) | file LICENSE

LazyData true

Imports doParallel, doRNG, foreach, gbm, xgboost, earth

Depends R (>= 3.3.0)

Suggests testthat

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-10-17 01:03:17

R topics documented:

boston_housing *Boston housing data*

Description

This dataset concerns the values of 506 houses in suburbs of Boston.

Usage

```
boston_housing
```

Format

A list of 4 components:

train A `data.frame` of the training dataset which contains 354 rows and 14 columns.

test A `data.frame` of the test dataset which contains 152 rows and 14 columns.

target_idx A column index of the target (response) varible.

pred_idx A set of column indicies of the predictors.

Source

https://archive.ics.uci.edu/ml/datasets/Housing

comperf *Compute model performance*

Description

This function computes model performance given a vector of response values and a vector of predictions.

Usage

```
comperf(y, yhat, w = rep(1, length(y)), pfmc = NULL, cdfx = "fpr",
  cdfy = "tpr", cutoff = 0.5)
```

Arguments

y	a vector of numeric response values.
yhat	a vector of model predictions.
w	an optional vector of observation weights.
pfmc	a character of the performance metric to be computed. For binary classification, pfmc accepts: • "acc": accuracy. • "dev": deviance. • "ks": Kolmogorov-Smirnov (KS) statistic. • "auc": area under the ROC curve. The default ROC curve is given by true positive rate (on the y-axis) vs. false positive rate (on the x-axis). A different curve can be obtained by setting the cdfx and cdfy arguments described below. • "roc": ROC curve given by true positive rate vs. false positive rate (default). A different curve can be obtained by setting the cdfx and cdfy arguments described below. If input to the argument cutoff is missing (default), the return value is a list of two components x and y representing the ROC curve. Otherwise, the return value is a single or a vector of evaluation(s) of the ROC curve at the cutoff. For regression, pfmc accepts: • "mse": mean squared error. • "mae": mean absolute error. • "rsq": r-squared (coefficient of determination).
cdfx	a character of the cumulative distribution for the x-axis. Supported values are • "fpr": false positive rate. • "fnr": false negative rate. • "rpp": rate of positive prediction.
cdfy	a character of the cumulative distribution for the y-axis. Supported values are • "tpr": true positive rate. • "tnr": true negative rate.
cutoff	a value in [0, 1] used for binary classification. If pfmc="acc", negative prediction has predicted probability <= cutoff and positive prediction has predicted probability > cutoff. If pfmc="roc", then this is used in conjunction with the cdfx and cdfy arguments (described above) which specify the cumulative distributions for the x-axis and y-axis of the ROC curve. For example, if the desired performance metric is the true positive rate at the 5% false positive rate, specify pfmc="roc", cdfx="fpr", cdfy="tpr", and cutoff=0.05.

Value

A single or a vector of numeric values of model performance, or a list of two components x and y representing the ROC curve.

Author(s)

Waley W. J. Liang <<wliang10@gmail.com>>

See Also

gbts, predict.gbts

Examples

```
y = c(0, 1, 0, 1, 1, 1)
yhat = c(0.5, 0.9, 0.2, 0.7, 0.6,  0.4)
comperf(y, yhat, pfmc = "auc")
# 0.875

y = 1:10
yhat = c(1:5 - 0.1, 6:10 + 0.1)
comperf(y, yhat, pfmc = "mse")
# 0.01
```

---

gbts *Hyperparameter Search for Gradient Boosted Trees*

Description

This package provides hyperparameter optimization for Gradient Boosted Trees (GBT) on binary classification and regression problems. The current version provides two optimization methods:

- Bayesian optimization:
  1. A probabilistic model is built to capture the relationship between hyperparameters and their predictive performance.
  2. Select the most predictive hyperparameter values (as suggested by the probabilistic model) to try in the next iteration.
  3. Train a GBT with the selected hyperparameter settings and compute its out-of-sample predictive performance.
  4. Update the probabilistic model with the new performance measure. Go back to step 2 and repeat.
- Random search: hyperparameters are selected uniformly at random in each iteration.

In both approaches, each iteration uses cross-validation (CV) to develop GBTs with the selected hyperparameter values on the training datasets followed by performance assessment on the validation datasets. For Bayesian optimization, validation performance is used to update the model of the relationship betwen hyperparameters and performance. The final result is a set of CV models having the best average validation performance. It does not re-run a new GBT with the best hyperparameter values on the full training data. Prediction is computed as the average of the predictions from the CV models.

Usage

```
gbts(x, y, w = rep(1, nrow(x)), nitr = 100, nlhs = floor(nitr/2),
  nprd = 1000, kfld = 10, nwrk = 2, srch = c("bayes", "random"),
  rpkg = c("gbm", "xgb"), pfmc = c("acc", "dev", "ks", "auc", "roc", "mse",
  "rsq", "mae"), cdfx = "fpr", cdfy = "tpr", cutoff = 0.5,
  max_depth_range = c(2, 10), leaf_size_range = c(10, 200),
  bagn_frac_range = c(0.1, 1), coln_frac_range = c(0.1, 1),
  shrinkage_range = c(0.01, 0.1), num_trees_range = c(50, 1000),
  scl_pos_w_range = c(1, 10), print_progress = TRUE)
```

Arguments

x	a data.frame of predictors. If rpkg (described below) is set to "gbm", then x is allowed to have categorical predictors represented as factors. Otherwise, all predictors in x must be numeric.
y	a vector of response values. For binary classification, y must contain values of 0 and 1. There is no need to convert y to a factor variable. For regression, y must contain at least two unique values.
w	an optional vector of observation weights.
nitr	an integer of the number of iterations for the optimization.
nlhs	an integer of the number of Latin Hypercube samples (each sample is a combination of hyperparameter values) used to generate the initial performance model. This is used for Bayesian optimization only. Random search ignores this argument.
nprd	an integer of the number of samples (each sample is a combination of hyperparameter values) at which performance prediction is made, and the best is selected to run the next iteration of GBT.
kfld	an integer of the number of folds for cross-validation used at each iteration.
nwrk	an integer of the number of computing workers (CPU cores) to be used. If nwrk is less than the number of available cores on the machine, it uses all available cores.
srch	a character indicating the search method such that srch="bayes" uses Bayesian optimization (default), and srch="random" uses random search.
rpkg	a character indicating which package of GBT to use. Setting rpkg="gbm" uses the gbm R package (default). Setting rpkg="xgb" uses the xgboost R package. Note that with gbm, predictors can be categorical represented as factors, as opposed to xgboost which requires all predictors to be numeric.
pfmc	a character of the performance metric to optimize. For binary classification, pfmc accepts: • "acc": accuracy. • "dev": deviance. • "ks": Kolmogorov-Smirnov (KS) statistic. • "auc": area under the ROC curve. This is used in conjunction with the cdfx and cdfy arguments (described below) which specify the cumulative

distributions for the x-axis and y-axis of the ROC curve, respectively. The default ROC curve is given by true positive rate (on the y-axis) vs. false positive rate (on the x-axis).

- `"roc"`: this is used when a point on the ROC curve is used as the performance metric, such as the true positive rate at a fixed false positive rate. This is used in conjunction with the `cdfx`, `cdfy`, and `cutoff` arguments which specify the cumulative distributions for the x-axis and y-axis of the ROC curve, and the cutoff (value on the x-axis) at which evaluation of the ROC curve is obtained as a performance metric. For example, if the desired performance metric is the true positive rate at the 5% false positive rate, specify `pfmc="roc"`, `cdfx="fpr"`, `cdfy="tpr"`, and `cutoff=0.05`.

For regression, `pfmc` accepts:

- `"mse"`: mean squared error.
- `"mae"`: mean absolute error.
- `"rsq"`: r-squared (coefficient of determination).

`cdfx` a character of the cumulative distribution for the x-axis. Supported values are

- `"fpr"`: false positive rate.
- `"fnr"`: false negative rate.
- `"rpp"`: rate of positive prediction.

`cdfy` a character of the cumulative distribution for the y-axis. Supported values are

- `"tpr"`: true positive rate.
- `"tnr"`: true negative rate.

`cutoff` a value in [0, 1] used for binary classification. If `pfmc="acc"`, instances with probabilities `<= cutoff` are predicted as negative, and those with probabilities `> cutoff` are predicted as positive. If `pfmc="roc"`, `cutoff` can be used in conjunction with the `cdfx` and `cdfy` arguments (described above) to specify the operating point. For example, if the desired performance metric is the true positive rate at the 5% false positive rate, specify `pfmc="roc"`, `cdfx="fpr"`, `cdfy="tpr"`, and `cutoff=0.05`.

`max_depth_range` a vector of the minimum and maximum values for: maximum tree depth.

`leaf_size_range` a vector of the minimum and maximum values for: leaf node size.

`bagn_frac_range` a vector of the minimum and maximum values for: bag fraction.

`coln_frac_range` a vector of the minimum and maximum values for: fraction of predictors to try for each split.

`shrinkage_range` a vector of the minimum and maximum values for: shrinkage.

`num_trees_range` a vector of the minimum and maximum values for: number of trees.

`scl_pos_w_range` a vector of the minimum and maximum values for: scale of weights for positive cases.

`print_progress` a logical of whether optimization progress should be printed to the console.

Value

A list of information with the following components:

- `best_perf`: a numeric value of the best average validation performance.
- `best_idx`: an integer of the iteration index for the best average validation performance.
- `best_model_cv`: a list of cross-validation models with the best average validation performance.
- `perf_val_cv`: a matrix of cross-validation performances where the rows correspond to iterations and the columns correspond to CV runs.
- `params`: a data.frame of hyperparameter values visited during the search. Each row of the data.frame comes from an iteration.
- `total_time`: a numeric value of the total time used in minutes.
- `objective`: a character of the objective function used.
- `...`: the rest of the output are echo of the input arguments (except for `x`, `y`, and `w`). See input argument documentation for details.

Author(s)

Waley W. J. Liang <<wliang10@gmail.com>>

See Also

`predict.gbts`, `comperf`

Examples

```
## Not run:
# Binary classification

# Load German credit data
data(german_credit)
train <- german_credit$train
test <- german_credit$test
target_idx <- german_credit$target_idx
pred_idx <- german_credit$pred_idx

# Train a GBT model with optimization on AUC
model <- gbts(train[, pred_idx], train[, target_idx], nitr = 200, pfmc = "auc")

# Predict on test data
prob_test <- predict(model, test[, pred_idx])

# Compute AUC on test data
comperf(test[, target_idx], prob_test, pfmc = "auc")


# Regression

# Load Boston housing data
```

```
data(boston_housing)
train <- boston_housing$train
test <- boston_housing$test
target_idx <- boston_housing$target_idx
pred_idx <- boston_housing$pred_idx

# Train a GBT model with optimization on MSE
model <- gbts(train[, pred_idx], train[, target_idx], nitr = 200, pfmc = "mse")

# Predict on test data
prob_test <- predict(model, test[, pred_idx])

# Compute MSE on test data
comperf(test[, target_idx], prob_test, pfmc = "mse")

## End(Not run)
```

`german_credit` *German credit data*

Description

This dataset classifies 1,000 people described by a set of attributes as good or bad credit risks.

Usage

```
german_credit
```

Format

A list of 4 components:

train A `data.frame` of the training dataset which contains 700 rows and 21 columns.

test A `data.frame` of the test dataset which contains 300 rows and 21 columns.

target_idx A column index of the target (response) varible.

pred_idx A set of column indicies of the predictors.

Source

https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)

predict.gbts *Predict method for optimized Gradient Boosted Trees*

Description

This function generates predictions for a given dataset using the returned object from gbts. It first generates a set of predictions for each cross-validation model, and then averaging them on the log-odds scale (for binary classification) or on the response scale (for regression) to produce the final prediction.

Usage

```
## S3 method for class 'gbts'
predict(object, x, nwrk = 2, ...)
```

Arguments

object	a model object returned from gbts.
x	a data.frame of predictors. It has to follow the same format and restrictions as the training dataset that generated the model.
nwrk	an integer of the number of computing cores to be used. If nwrk is less than the number of available cores on the machine, it uses all available cores.
...	further arguments passed to or from other methods.

Value

A numeric vector of predictions. In the case of binary classification, predictions are probabilities.

Author(s)

Waley W. J. Liang <<wliang10@gmail.com>>

See Also

gbts, comperf

Index