inside-R

A Community Site for R – Sponsored by Revolution Analytics

train {caret}

Fit Predictive Models over Different Tuning Parameters

`Package: caret`
Version: 6.0-24

Description

This function sets up a grid of tuning parameters for a number of classification and regression routines, fits each model and calculates a resampling based performance measure.

Usage

```
train(x, ...)

## S3 method for class 'default':
train((x, y,
      method = "rf",
      preProcess = NULL,
      ...,
      weights = NULL,
      metric = ifelse(is.factor(y), "Accuracy", "RMSE"),
      maximize = ifelse(metric == "RMSE", FALSE, TRUE),
      trControl = trainControl(),
      tuneGrid = NULL,
      tuneLength = 3))

## S3 method for class 'formula':
train((form, data, ..., weights, subset, na.action, contrasts = NULL))
```

Arguments

`x`
a data frame containing training data where samples are in rows and features are in columns.

`y`
a numeric or factor vector containing the outcome for each sample.

`form`
A formula of the form `y ~ x1 + x2 + ...`

`data`
Data frame from which variables specified in `formula` are preferentially to be taken.

`weights`
a numeric vector of case weights. This argument will only affect models that allow case weights.

`subset`
An index vector specifying the cases to be used in the training sample. (NOTE: If given, this argument must be named.)

`na.action`
A function to specify the action to be taken if NAs are found. The default action is for the procedure to fail. An alternative is na.omit, which leads to rejection of cases with missing values on any required variable. (NOTE: If given, this argument must be named.)

`contrasts`
a list of contrasts to be used for some or all of the factors appearing as variables in the model formula.

`method`
a string specifying which classification or regression model to use. Possible values are found using `names(getModelInfo())`. See http://caret.r-forge.r-project.org/bytag.html. A list of funciotns can also be passed for a custom model function. See http://caret.r-forge.r-project.org/custom_models.html for

details.

`...`
arguments passed to the classification or regression routine (such as `randomForest`). Errors will occur if values for tuning parameters are passed here.

`preProcess`
a string vector that defines an pre-processing of the predictor data. Current possibilities are "BoxCox", "YeoJohnson", "expoTrans", "center", "scale", "range", "knnImpute", "bagImpute", "medianImpute", "pca", "ica" and "spatialSign". The default is no pre-processing. See `preProcess` and `trainControl` on the procedures and how to adjust them.

`metric`
a string that specifies what summary metric will be used to select the optimal model. By default, possible values are "RMSE" and "Rsquared" for regression and "Accuracy" and "Kappa" for classification. If custom performance metrics are used (via the `summaryFunction` argument in `trainControl`, the value of `metric` should match one of the arguments. If it does not, a warning is issued and the first metric given by the `summaryFunction` is used. (NOTE: If given, this argument must be named.)

`maximize`
a logical: should the metric be maximized or minimized?

`trControl`
a list of values that define how this function acts. See `trainControl` and http://caret.r-forge.r-project.org/training.html#custom. (NOTE: If given, this argument must be named.)

`tuneGrid`
a data frame with possible tuning values. The columns are named the same as the tuning parameters. Use `getModelInfo` to get a list of tuning parameters for each model or see http://caret.r-forge.r-project.org/modelList.html. (NOTE: If given, this argument must be named.)

`tuneLength`
an integer denoting the number of levels for each tuning parameters that should be generated by `train`. (NOTE: If given, this argument must be named.)

Details

`train` can be used to tune models by picking the complexity parameters that are associated with the optimal resampling statistics. For particular model, a grid of parameters (if any) is created and the model is trained on slightly different data for each candidate combination of tuning parameters. Across each data set, the performance of held-out samples is calculated and the mean and standard deviation is summarized for each combination. The combination with the optimal resampling statistic is chosen as the final model and the entire training set is used to fit a final model.

More details on this function can be found at http://caret.r-forge.r-project.org/training.html.

A variety of models are currently available and are enumerated by tag (i.e. their model characteristics) at http://caret.r-forge.r-project.org/bytag.html.

Values

A list is returned of class `train` containing:

`method`
the chosen model.

`modelType`
an identifier of the model type.

`results`
a data frame the training error rate and values of the tuning parameters.

`bestTune`
a data frame with the final parameters.

`call`
the (matched) function call with dots expanded

`dots`
a list containing any ... values passed to the original call

`metric`
a string that specifies what summary metric will be used to select the optimal model.

`control`
the list of control parameters.

`preProcess`
either `NULL` or an object of class `preProcess`

`finalModel`
an fit object using the best parameters

`trainingData`
a data frame

`resample`
A data frame with columns for each performance metric. Each row corresponds to each resample. If leave-one-out cross-validation or out-of-bag estimation methods are requested, this will be `NULL`. The `returnResamp` argument of `trainControl` controls how much of the resampled results are saved.

`perfNames`
a character vector of performance metrics that are produced by the summary function

`maximize`
a logical recycled from the function arguments.

`yLimits`
the range of the training set outcomes.

`times`
a list of execution times: `everything` is for the entire call to `train`, `final` for the final model fit and, optionally, `prediction` for the time to predict new samples (see `trainControl`)

References

http://caret.r-forge.r-project.org/training.html

Kuhn (2008), “Building Predictive Models in R Using the caret” (http://www.jstatsoft.org/v28/i05/)

See Also

`trainControl`, `update.train`, `modelLookup`, `createFolds`

Examples

```
## Not run:

#####################################
## Classification Example

data(iris)
TrainData <- iris[,1:4]
TrainClasses <- iris[,5]

knnFit1 <- train(TrainData, TrainClasses,
                 method = "knn",
                 preProcess = c("center", "scale"),
                 tuneLength = 10,
                 trControl = trainControl(method = "cv"))

knnFit2 <- train(TrainData, TrainClasses,
                 method = "knn",
                 preProcess = c("center", "scale"),
                 tuneLength = 10,
                 trControl = trainControl(method = "boot"))


library(MASS)
nnetFit <- train(TrainData, TrainClasses,
                 method = "nnet",
                 preProcess = "range",
                 tuneLength = 2,
                 trace = FALSE,
                 maxit = 100)

#####################################
## Regression Example

library(mlbench)
data(BostonHousing)

lmFit <- train(medv ~ . + rm:lstat,
               data = BostonHousing,
               "lm")

library(rpart)
rpartFit <- train(medv ~ .,
                  data = BostonHousing,
                  "rpart",
                  tuneLength = 9)

#####################################
## Example with a custom metric
```

```
madSummary <- function (data,
                        lev = NULL,
                        model = NULL)
{
  out <- mad(data$obs - data$pred,
             na.rm = TRUE)
  names(out) <- "MAD"
  out
}

robustControl <- trainControl(summaryFunction = madSummary)
marsGrid <- expand.grid(degree = 1, nprune = (1:10) * 2)

earthFit <- train(medv ~ .,
                  data = BostonHousing,
                  "earth",
                  tuneGrid = marsGrid,
                  metric = "MAD",
                  maximize = FALSE,
                  trControl = robustControl)

#####################################
## Parallel Processing Example via multicore package

## library(doMC)
## registerDoMC(2)

## NOTE: don't run models form RWeka when using
### multicore. The session will crash.

## The code for train() does not change:
set.seed(1)
usingMC <-  train(medv ~ .,
                  data = BostonHousing,
                  "glmboost")

## or use:
## library(doMPI) or
## library(doSMP) and so on

## End(Not run)
```

Author(s)

Max Kuhn (the guts of `train.formula` were based on Ripley's `nnet.formula`)

Documentation reproduced from package caret, version 6.0-24. License: GPL-2
