Package 'spFSR'

May 11, 2018

Type Package

Title Feature Selection and Ranking by Simultaneous Perturbation Stochastic Approximation

Version 1.0.0

Date 2018-05-10

Description An implementation of feature selection and ranking via simultaneous perturbation stochastic approximation (SPSA-FSR) based on works by V. Aksakalli and M. Malekipirbazari (2015) <arXiv:1508.07630> and Zeren D. Yenice and et al. (2018) <arXiv:1804.05589>. The SPSA-FSR algorithm searches for a locally optimal set of features that yield the best predictive performance using a specified error measure such as mean squared error (for regression problems) and accuracy rate (for classification problems). This package requires an object of class 'task' and an object of class 'Learner' from the 'mlr' package.

License GPL-3

Encoding UTF-8

Depends mlr (>= 2.11), parallelMap (>= 1.3), parallel (>= 3.4.2), tictoc (>= 1.0)

Imports ggplot2 (>= 2.2.1), class (>= 7.3), mlbench (>= 2.1)

Suggests caret (>= 6.0), MASS (>= 7.3), knitr, rmarkdown

URL https://www.featureranking.com/, https://arxiv.org/abs/1804.05589

BugReports https://github.com/yongkai17/spFSR/issues

LazyData true

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Vural Aksakalli [aut, cre],
Babak Abbasi [aut, ctb],
Yong Kai Wong [aut, ctb],
Zeren D. Yenice [ctb]

Maintainer Vural Aksakalli <vaksakalli@gmail.com>

Repository CRAN

Date/Publication 2018-05-11 13:15:16 UTC

R topics documented:

`getBestModel` *Extract the wrapped model of the best performing features from an spFSR object*

Description

Returns a fitted model which uses the best performing feature subset generated by spFSR. It inherits all methods or functions applied to a `WrappedModel` objects. For example, the `predict` function can be used on the fitted model. If it is a classifcation model, a confusion matrix can be obtained by calling the calculateConfusionMatrix function. See spFeatureSelection for a more detailed example.

Usage

```
getBestModel(x)
```

Arguments

`x`	an `spFSR` object

Value

A `WrappedModel` object of the best performing features.

Author(s)

Yong Kai Wong <yongkai1017@gmail.com>

See Also

spFeatureSelection

getImportance *Extract feature importance data from a spFSR object*

Description

Returns importance ranks of best performing features. See spFeatureSelection for a more detailed example.

Usage

```
getImportance(x)
```

Arguments

x	a spFSR object

Value

A data.frame of features and feature importance

Author(s)

Yong Kai Wong <yongkai1017@gmail.com>

See Also

plotImportance and spFeatureSelection.

plot.spFSR *Plot an spFSR object*

Description

Plot for an spFSR object. It returns a scatterplot of measure values vs. iteration. The error bar of measure values at each iteration can be included. It also allows user to identify the iteration which yields the best measure value. See spFeatureSelection for a more detailed example.

Usage

```
## S3 method for class 'spFSR'
plot(x, errorBar = FALSE, annotateBest = FALSE,
  se = FALSE, ...)
```

Arguments

x	a spFSR object
errorBar	If TRUE, an error bar of +/- 1 standard deviation will be included around the mean error measure value at each iteration. When it is TRUE, the ylim argument cannot be used. The default is FALSE.
annotateBest	If TRUE, the best result will be highlighted and annotated. The default is FALSE.
se	If TRUE, an error bar of $\pm$ standard error will be included around the mean error measure value at each iteration. When it is TRUE, the ylim argument cannot be used. The se does not produce any error bar if errorBar is set as FALSE. Note that if the standard error is used, the error bar has a narrower range. The default is FALSE.
...	Additional plot parameters that can be passed into the plot function.

Value

Plot of error measure values vs iterations of a spFSR object with an error bar (if included).

Author(s)

Yong Kai Wong <yongkai1017@gmail.com>

See Also

plotImportance and spFeatureSelection.

---

plotImportance *Plot importance ranks of best performing features from a spFSR object*

Description

Return a vertical bar chart of features vs. feature importance. See spFeatureSelection for a more detailed example.

Usage

```
plotImportance(x, low = "darkblue", high = "black")
```

Arguments

x	an spFSR object
low	Color for the lowest importance. The default is darkblue.
high	Color for the highest importance. The default is black.

Value

a ggplot object: a vertical bar chart of features and feature importance.

Author(s)

Yong Kai Wong <yongkai1017@gmail.com>

See Also

plotImportance, spFSR.default, and spFeatureSelection.

`spFeatureSelection` *Feature selection and ranking by SPSA-FSR*

Description

Searches for the best performing set of features, either automatically or for a given number of features, and ranks them by their importance via the simultaneous perturbation stochastic approximation (SPSA) algorithm for given a task, wrapper, and performance criterion. The task, the wrapper, and the performance criterion are defined using the **mlr** package.

Usage

```
spFeatureSelection(task, wrapper, measure, norm.method = "standardize",
  num.features.selected, ...)
```

Arguments

`task`	A `task` object created using **mlr** package. It must be either a `ClassifTask` or `RegrTask` object.
`wrapper`	A `Learner` object created using **mlr** package. Multiple learners object is not supported.
`measure`	A performance measure supported by the `task`.
`norm.method`	Normalization method for features. NULL value is allowed. Supported methods are 'standardize', 'range', 'center', and 'scale'. Default value is 'standardize'.
`num.features.selected`	Number of features to be selected. Must be between zero and total number of features in the task. A value of zero results in automatic feature selection.
`...`	Additional arguments. For more details, see spFSR.default.

Value

`spFSR` returns an object of class "spFSR". An object of class "spFSR" consists of the following:

`task.spfs`	An **mlr** package `task` object defined on the best performing features.
`wrapper`	An **mlr** package `learner` object as specified by the user.
`measure`	An **mlr** package performance measure as specified by the user.
`param best.model`	An **mlr** package `WrappedModel` object trained by the `wrapper` using `task.spfs`.

`iter.results`	A `data.frame` object containing detailed information on each iteration.
`features`	Names of the best performing features.
`num.features`	The number of best performing features.
`importance`	A vector of importance ranks of the best performing features.
`total.iters`	The total number of iterations executed.
`best.iter`	The iteration where the best performing feature subset was encountered.
`best.value`	The best measure value encountered during execution.
`best.std`	The standard deviation corresponding to the best measure value encountered.
`run.time`	Total run time in minutes
`rdesc.feat.eval`	Resampling specification
`call`	Call

Author(s)

Vural Aksakalli <vaksakalli@gmail.com>

Babak Abbasi <babak.abbasi@rmit.edu.au>, <b.abbasi@gmail.com>

Yong Kai Wong <yongkai1017@gmail.com>

References

V. Aksakalli and M. Malekipirbazari (2015) Feature Selection via Binary Simultaneous Perturbation Stochastic Approximation, *Pattern Recognition Letters*, **Vol. 75**, 41 – 47. See https://doi.org/10.1016/j.patrec.2016.03.002

See Also

makeClassifTask, makeRegrTask, makeLearner and spFSR.default.

Examples

```
data(iris)         # load the data
library(mlr)       # load the mlr package

if( requireNamespace('class', quietly = TRUE) ){

  # Load class so that a knn classifier can be defined
  library('class')

  # define classification task on 20 random samples
  task    <- makeClassifTask(data = iris[sample(150, 20),],
                             target = 'Species')

  # define a wrapper (1-KNN classifier)
  wrapper <- makeLearner('classif.knn', k = 1)
```

```
# run spsa with 2 iterations
# to select 1 out of 4 features
spsaMod <- spFeatureSelection( task = task,
                               wrapper = wrapper,
                               measure = mmce,
                               num.features.selected = 1,
                               num.cores = 1,
                               iters.max = 2)

# obtain summary
summary(spsaMod)


# plot with error bars
plot(spsaMod, errorBar = TRUE)

# obtain the wrapped model with the best performing features
bestMod <- getBestModel(spsaMod)

# predict using the best mod
pred    <- predict(bestMod, task = spsaMod$task.spfs )

# Obtain confusion matrix
calculateConfusionMatrix( pred )

# Get the importance ranks of best performing features
getImportance(spsaMod)
plotImportance(spsaMod)

}
```

---

spFSR.default *Default function of feature selection and ranking by SP-FSR*

Description

This is the default function of spFeatureSelection. See spFeatureSelection for more details.

Usage

```
## Default S3 method:
spFSR(task, wrapper, measure, norm.method = "standardize",
  num.features.selected = 0L, features.to.keep = NULL, iters.max = 100L,
  stall.limit = 20L, stall.tolerance = 10^(-7), num.grad.avg = 3L,
  num.gain.smoothing = 3L, perturb.amount = 0.05, gain.min = 0.01,
```

```
  gain.max = 1, perf.eval.method = "cv", num.cv.folds = 5L,
  num.cv.reps.grad.avg = 3L, num.cv.reps.feat.eval = 3L,
  cv.stratify = TRUE, run.parallel = TRUE, num.cores = NULL,
  show.info = TRUE, print.freq = 1L)
```

Arguments

task	A task object created using **mlr** package. It must be either a ClassifTask or RegrTask object.
wrapper	A Learner object created using **mlr** package. Multiple learners object is not supported.
measure	A performance measure within the **mlr** package supported by the task.
norm.method	Normalization method for features. NULL value is allowed. Supported methods are 'standardize', 'range', 'center', and 'scale'. Default value is 'standardize'.
num.features.selected	Number of features selected. It must be a nonnegative integer and must not exceed the total number of features in the task. A value of 0 results in automatic feature selection. Default value is 0L.
features.to.keep	Names of features to keep in addition to num.features.selected. Default value is NULL.
iters.max	Maximum number of iterations to execute. The minimum value is 2L. Default value is 100L.
stall.limit	Number of iterations to stall, that is, to continue without at least stall.tolerance improvement to the measure value. The mininum value is 2L. Default value is 20L.
stall.tolerance	Value of stall tolerance. It must be strictly positive. Default value is 1/10^7.
num.grad.avg	Number of gradients to average for gradient approximation. It must be a positive integer. Default value is 3L.
num.gain.smoothing	Number of most recent gains to use in gain smoothing. It must be a positive integer. Default value is 3L.
perturb.amount	Perturbation amount for feature importances during gradient approximation. It must be a value between 0.01 and 0.1. Default value is 0.05.
gain.min	The minimum gain value. It must be greater than or equal to 0.001. Default value is 0.01.
gain.max	The maximum gain value. It must be greater than or equal to gain.min. Default value is 1.0.
perf.eval.method	Performance evaluation method. It must be either 'cv' for cross-validation or 'resub' for resubstitution. Default is 'cv'.
num.cv.folds	The number of cross-validation folds when 'cv' is selected as perf.eval.method. The minimum value is 3L. Default value is 5L.

num.cv.reps.grad.avg
: The number of cross-validation repetitions for gradient averaging. It must be a positive integer. Default value is 3L.

num.cv.reps.feat.eval
: The number of cross-validation repetitions for feature subset evaluation. It must be a positive integer. Default value is 3L.

cv.stratify
: Logical argument. Stratify cross-validation? Default value is TRUE.

run.parallel
: Logical argument. Perform cross-validations in parallel? Default value is TRUE.

num.cores
: Number of cores to use in case of a parallel run. It must be less than or equal to the total number of cores on the host machine. If set to NULL when run.parallel is TRUE, it is taken as one less of the total number of cores.

show.info
: If set to TRUE, iteration information is displayed at print frequency.

print.freq
: Iteration information printing frequency. It must be a positive integer. Default value is 1L.

Value

spFSR returns an object of class "spFSR". An object of class "spFSR" consists of the following:

task.spfs
: An **mlr** package task object defined on the best performing features.

wrapper
: An **mlr** package learner object as specified by the user.

measure
: An **mlr** package performance measure as specified by the user.

param best.model
: An **mlr** package WrappedModel object trained by the wrapper using task.spfs.

iter.results
: A data.frame object containing detailed information on each iteration.

features
: Names of the best performing features.

num.features
: The number of best performing features.

importance
: A vector of importance ranks of the best performing features.

total.iters
: The total number of iterations executed.

best.iter
: The iteration where the best performing feature subset was encountered.

best.value
: The best measure value encountered during execution.

best.std
: The standard deviation corresponding to the best measure value encountered.

run.time
: Total run time in minutes.

call
: Call.

Author(s)

Vural Aksakalli <vaksakalli@gmail.com>

Babak Abbasi <babak.abbasi@rmit.edu.au>, <b.abbasi@gmail.com>

Yong Kai Wong <yongkai.wong@rmit.edu.au>, <yongkai1017@gmail.com>

References

V. Aksakalli and M. Malekipirbazari (2015) Feature Selection via Binary Simultaneous Perturbation Stochastic Approximation, *Pattern Recognition Letters*, **Vol. 75**, 41 – 47. See https://doi.org/10.1016/j.patrec.2016.03.002

See Also

spFeatureSelection.

`summary.spFSR` *Summarising an spFSR object*

Description

Summarising an spFSR object

Usage

```
## S3 method for class 'spFSR'
summary(object, ...)
```

Arguments

`object`	A `spFSR` object
`...`	Additional arguments

Value

Summary of an spFSR object consisting of number of features selected, wrapper type, total number of iterations, the best performing features, and the descriptive statistics of the best iteration result (the iteration where the best performing features are found).

Author(s)

Yong Kai Wong <yongkai1017@gmail.com>

See Also

getImportance, spFSR.default, and spFeatureSelection.

Index