Package 'tree.interpreter'

October 30, 2019

Type Package

Title Random Forest Prediction Decomposition and Feature Importance Measure

Version 0.1.0

Date 2019-10-15

Author Qingyao Sun

Maintainer Qingyao Sun <sunqingyao19970825@gmail.com>

Description An R re-implementation of the 'treeinterpreter' package on PyPI <https://pypi.org/project/treeinterpreter/>. Each prediction can be decomposed as 'prediction = bias + feature_1_contribution + ... + feature_n_contribution'. This decomposition is then used to calculate the Mean Decrease Impurity (MDI) and Mean Decrease Impurity using out-of-bag samples (MDI-oob) feature importance measures based on the work of Li et al. (2019) <arXiv:1906.10845>.

Encoding UTF-8

License MIT + file LICENSE

Imports Rcpp (>= 1.0.2)

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.1.1

Suggests MASS, randomForest, ranger, testthat (>= 2.1.0), knitr, rmarkdown, covr

URL https://github.com/nalzok/tree.interpreter

BugReports https://github.com/nalzok/tree.interpreter/issues

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-10-30 17:40:02 UTC

R topics documented:

`featureContribTree` *Feature Contribution*

Description

Contribution of each feature to the prediction.

Usage

```
featureContribTree(tidy.RF, tree, X)

featureContrib(tidy.RF, X)
```

Arguments

`tidy.RF`	A tidy random forest. The random forest to make predictions with.
`tree`	An integer. The index of the tree to look at.
`X`	A data frame. Features of samples to be predicted.

Details

Recall that each node in a decision tree has a prediction associated with it. For regression trees, it's the average response in that node, whereas in classification trees, it's the frequency of each response class, or the most frequent response class in that node.

For a tree in the forest, the contribution of each feature to the prediction of a sample is the sum of differences between the predictions of nodes which split on the feature and those of their children, i.e. the sum of changes in node prediction caused by spliting on the feature. This is the calculated by `featureContribTree`.

For a forest, the contribution of each feature to the prediction if a sample is the average contribution across all trees in the forest. This is because the prediction of a forest is the average of the predictions of its trees. This is calculated by `featureContrib`.

Together with `trainsetBias(Tree)`, they can decompose the prediction by feature importance:

$$prediction(MODEL, X) = trainsetBias(MODEL) + featureContrib_1(MODEL, X) + ... + featureContrib_p(MO$$

where MODEL can be either a tree or a forest.

Value

A cube (3D array). The content depends on the type of the response.

- Regression: A P-by-1-by-N array, where P is the number of features in `X`, and N the number of samples in `X`. The pth row of the nth slice stands for the contribution of feature p to the prediction for response n.
- Classification: A P-by-D-by-N array, where P is the number of features in `X`, D is the number of response classes, and N is the number of samples in `X`. The pth row of the nth slice stands for the contribution of feature p to the prediction of each response class for response n.

Functions

- `featureContribTree`: Feature contribution to prediction within a single tree
- `featureContrib`: Feature contribution to prediction within the whole forest

References

Interpreting random forests http://blog.datadive.net/interpreting-random-forests/

Random forest interpretation with scikit-learn http://blog.datadive.net/random-forest-interpretation-with-sci

See Also

`trainsetBias`

`MDI`

Examples

```
library(ranger)
test.id <- 50 * seq(3)
rfobj <- ranger(Species ~ ., iris[-test.id, ], keep.inbag=TRUE)
tidy.RF <- tidyRF(rfobj, iris[-test.id, -5], iris[-test.id, 5])
featureContribTree(tidy.RF, 1, iris[test.id, -5])
featureContrib(tidy.RF, iris[test.id, -5])
```

`MDIoobTree` *Debiased Mean Decrease in Impurity*

Description

Calculate the MDI-oob feature importance measure.

Usage

```
MDIoobTree(tidy.RF, tree, trainX, trainY)

MDIoob(tidy.RF, trainX, trainY)
```

Arguments

tidy.RF	A tidy random forest. The random forest to calculate MDI-oob from.
tree	An integer. The index of the tree to look at.
trainX	A data frame. Train set features, such that the Tth tree is trained with X[tidy.RF$inbag.counts[[T]],].
trainY	A data frame. Train set responses, such that the Tth tree is trained with Y[tidy.RF$inbag.counts[[T]],

Details

It has long been known that MDI incorrectly assigns high importance to noisy features, leading to systematic bias in feature selection. To address this issue, Li et al. proposed a debiased MDI feature importance measure using out-of-bag samples, called MDI-oob, which has achieved state-of-the-art performance in feature selection for both simulated and real data.

See `vignette('MDI',package='tree.interpreter')` for more context.

Value

A matrix. The content depends on the type of the response.

- Regression: A P-by-1 matrix, where P is the number of features in X. The pth row contains the MDI-oob of feature p.
- Classification: A P-by-D matrix, where P is the number of features in X and D is the number of response classes. The dth column of the pth row contains the MDI-oob of feature p to class d. You can get the MDI-oob of each feature by calling rowSums on the result.

Functions

- MDIoobTree: Debiased mean decrease in impurity within a single tree
- MDIoob: Debiased mean decrease in impurity within the whole forest

References

A Debiased MDI Feature Importance Measure for Random Forests https://arxiv.org/abs/1906.10845

See Also

MDI

`vignette('MDI',package='tree.interpreter')`

Examples

```
library(ranger)
rfobj <- ranger(Species ~ ., iris, keep.inbag=TRUE)
tidy.RF <- tidyRF(rfobj, iris[, -5], iris[, 5])
MDIoobTree(tidy.RF, 1, iris[, -5], iris[, 5])
MDIoob(tidy.RF, iris[, -5], iris[, 5])
```

MDITree *Mean Decrease in Impurity*

Description

Calculate the MDI feature importance measure.

Usage

```
MDITree(tidy.RF, tree, trainX, trainY)

MDI(tidy.RF, trainX, trainY)
```

Arguments

tidy.RF	A tidy random forest. The random forest to calculate MDI from.
tree	An integer. The index of the tree to look at.
trainX	A data frame. Train set features, such that the Tth tree is trained with X[tidy.RF$inbag.counts[[T]],].
trainY	A data frame. Train set responses, such that the Tth tree is trained with Y[tidy.RF$inbag.counts[[T]],

Details

MDI stands for Mean Decrease in Impurity. It is a widely adopted measure of feature importance in random forests. In this package, we calculate MDI with a new analytical expression derived by Li et al. (See references)

See `vignette('MDI',package='tree.interpreter')` for more context.

Value

A matrix. The content depends on the type of the response.

- Regression: A P-by-1 matrix, where P is the number of features in X. The pth row contains the MDI of feature p.
- Classification: A P-by-D matrix, where P is the number of features in X and D is the number of response classes. The dth column of the pth row contains the MDI of feature p to class d. You can get the MDI of each feature by calling rowSums on the result.

Functions

- MDITree: Mean decrease in impurity within a single tree
- MDI: Mean decrease in impurity within the whole forest

References

A Debiased MDI Feature Importance Measure for Random Forests https://arxiv.org/abs/1906.10845

See Also

MDIoob

vignette('MDI',package='tree.interpreter')

Examples

```
library(ranger)
rfobj <- ranger(Species ~ ., iris, keep.inbag=TRUE)
tidy.RF <- tidyRF(rfobj, iris[, -5], iris[, 5])
MDITree(tidy.RF, 1, iris[, -5], iris[, 5])
MDI(tidy.RF, iris[, -5], iris[, 5])
```

---

tidyRF *Tidy Random Forest*

Description

Converts random forest objects from various libraries into a common structure, i.e. a "tidy" random forest, calculating absent auxiliary information on demand, in order to provide a uniform interface for other `tree.interpreter` functions. Note that the output is of a private format, and is subject to change.

Usage

```
tidyRF(rfobj, trainX, trainY)
```

Arguments

`rfobj`	A random forest object. Currently, supported libraries include `randomForest` and `ranger`.
`trainX`	A data frame. Train set features.
`trainY`	A data frame. Train set responses.

Value

A list of class `tidyRF`, with entries:

`num.classes` An integer. For classification trees, number of classes of the response. For regression trees, this field will always be 1.

`num.trees` An integer. Number of trees.

`feature.names` A vector. Names of features.

`inbag.counts` A list. For each tree, a vector of the number of times the observations are in-bag in the trees.

`left.children` A list. For each tree, a vector of the left children of its nodes.

`right.children` A list. For each tree, a vector of the right children of its nodes.

`split.features` A list. For each tree, a vector of the indices of features used at its nodes. Indices start from 0. A value of 0 means the node is terminal. This does not cause ambiguity, because the root node will never be a child of other nodes.

`split.values` A list. For each tree, a vector of the values of features used at its nodes.

`node.sizes` A list. For each tree, a vector of the sizes of its nodes.

`node.resp` A list. For each tree, a vector of the responses of its nodes.

`delta.node.resp.left` A list. For each tree, a vector of the difference between the responses of the left children of its nodes and themselves.

`delta.node.resp.right` A list. For each tree, a vector of the difference between the responses of the right children of its nodes and themselves.

Examples

```
library(ranger)
rfobj <- ranger(Species ~ ., iris, keep.inbag=TRUE)
tidy.RF <- tidyRF(rfobj, iris[, -5], iris[, 5])
str(tidy.RF, max.level=1)
```

`trainsetBiasTree` *Trainset Bias*

Description

For a tree in the forest, trainset bias is the prediction of its root node, or the unconditional prediction of the tree, or the average response of the samples used to train the tree.

Usage

```
trainsetBiasTree(tidy.RF, tree)

trainsetBias(tidy.RF)
```

Arguments

`tidy.RF`	A tidy random forest. The random forest to extract train set bias from.
`tree`	An integer. The index of the tree to look at.

Details

For a forest, the trainset bias is simply the average trainset bias across all trees. This is because the prediction of a forest is the average of the predictions of its trees.

Together with `featureContrib(Tree)`, they can decompose the prediction by feature importance:

$$prediction(MODEL, X) = trainsetBias(MODEL) + featureContrib_1(MODEL, X) + ... + featureContrib_p(MO$$

where MODEL can be either a tree or a forest.

Value

A matrix. The content depends the type of the response.

- Regression: A 1-by-1 matrix. The trainset bias for the prediction of the response.
- Classification: A 1-by-D matrix, where D is the number of response classes. Each column of the matrix stands for the trainset bias for the prediction of each response class.

Functions

- `trainsetBiasTree`: Trainset bias within a single tree
- `trainsetBias`: Trainset bias within the whole forest

References

Interpreting random forests http://blog.datadive.net/interpreting-random-forests/

Random forest interpretation with scikit-learn http://blog.datadive.net/random-forest-interpretation-with-scil

See Also

featureContrib

Examples

```
library(ranger)
rfobj <- ranger(Species ~ ., iris, keep.inbag=TRUE)
tidy.RF <- tidyRF(rfobj, iris[, -5], iris[, 5])
trainsetBiasTree(tidy.RF, 1)
trainsetBias(tidy.RF)
```

---

tree.interpreter *Random Forest Prediction Decomposition and Feature Importance Measure*

Description

An R re-implementation of the 'treeinterpreter' package on PyPI. <https://pypi.org/project/treeinterpreter/>. Each prediction can be decomposed as 'prediction = bias + feature_1_contribution + ... + feature_n_contribution'. This decomposition is then used to calculate the Mean Decrease Impurity (MDI) and Mean Decrease Impurity using out-of-bag samples (MDI-oob) feature importance measures based on the work of Li et al. (2019) <arXiv:1906.10845>.

`tidyRF`

The function `tidyRF` can turn a `randomForest` or `ranger` object into a package-agnostic random forest object. All other functions in this package operate on such a `tidyRF` object.

The `featureContrib` and `trainsetBias` families

The `featureContrib` and `trainsetBias` families can decompose the prediction of regression/classification trees/forests into bias and feature contribution components.

The `MDI` and `MDIoob` families

The `MDI` family can calculate the good old MDI feature importance measure, which unfortunately has some feature selection bias. MDI-oob is a debiased MDI feature importance measure that has achieved state-of-the-art performance in feature selection for both simulated and real data. It can be calculated with functions from the `MDIoob` family.

Examples

```
library(ranger)
rfobj <- ranger(mpg ~ ., mtcars, keep.inbag = TRUE)
tidy.RF <- tidyRF(rfobj, mtcars[, -1], mtcars[, 1])
MDIoob(tidy.RF, mtcars[, -1], mtcars[, 1])
```

Index