Package ‘ciu’

November 20, 2020

Type Package
Title Contextual Importance and Utility
Version 0.1.0
Author Kary Främling
Maintainer Kary Främling <Kary.Framling@umu.se>
Description Implementation of the Contextual Importance and Utility (CIU) concepts for Explainable AI (XAI). A recent description of CIU can be found in e.g. Främling (2020) <arXiv:2009.13996>.
License MIT + file LICENSE
Encoding UTF-8
LazyData true
Imports graphics, stats, Rcpp, grDevices, ggplot2
Suggests MASS, caret, gbm
RoxygenNote 7.1.1
NeedsCompilation no
Repository CRAN
Date/Publication 2020-11-20 08:50:02 UTC

R topics documented:

  ciu-package ...................................................... 2
  barplot.ciu ...................................................... 3
  ciu.blackbox.new .............................................. 4
  ciu.new .......................................................... 6
  ciu.relative .................................................... 9
  ciu.result.new ................................................ 9
  explain .......................................................... 10
  ggplot.col.ciu ................................................ 11
  pie.ciu ........................................................ 13
  plot.ciu ......................................................... 14
  plot.ciu.3D ..................................................... 15

Index ............................................................. 17
ciu-package

Description


Details

This package implements the Contextual Importance and Utility (CIU) concepts for Explainable AI (XAI). CIU allows explaining outputs values of any regression or classification systems, no matter if it is a "black-box" or a "white-box" AI, or anything between black and white. CIU is entirely model-agnostic. Contrary to most (all?) other XAI methods, CIU provides explanations directly based on the observed input-output behavior without building an intermediate "interpretable" model for doing it.

CIU was developed by Kary Främling in his PhD thesis, which was presented in 1996 (in French). CIU was first presented in 1995 at the International Conference on Artificial Neural Networks (ICANN).

The ciu package supports models from caret and at least lda natively, but can easily be made to work with any model.

Main functions:

Use of ciu starts by calling the function ciu.new that returns an object of class CIU. If the ciu object is created by ciu <- ciu.new(...), then different methods can be called as ciu$explain(), ciu$barplot.ciu() etc. for obtaining explanations in different forms.

ciu is implemented using an "old style" (?) R object orientation. However, it provides object-oriented encapsulation of variables and methods of the CIU object, which presumably helps to avoid name conflicts with other packages or user code.

References


**barplot.ciu**  

*Barplot CIU explanation for specific instance*

**Description**

Create a barplot showing CI as the length of the bar and CU on color scale from red to green, via yellow, for the given inputs and the given output. First get a CIU object by calling `ciu.new` as e.g. `ciu <- ciu.new(...)`, then call `ciu.res <- ciu$barplot.ciu(...)`. *Usage* section is in *Details* section because Roxygen etc. don’t support documentation of functions within functions.

**Arguments**

- **instance**: Instance to explain. See `explain`.
- **ind.inputs**: `vector` of indices for the inputs to be included in the plot. If NULL then all inputs will be included.
- **ind.output**: Index of output to be explained.
- **in.min.max.limits**: See `explain`.
- **n.samples**: See `explain`.
- **neutral.CU**: Indicates when the Contextual Utility is considered to be "negative". The default value of 0.5 seems quite logical for most cases.
- **show.input.values**: Include input values after input labels or not. Default is TRUE.
- **concepts.to.explain**: List of concepts to use in the plot, as defined by vocabulary provided as argument to `ciu.new`. If `ind.inputs=NULL`, then use `concepts.to.explain` instead. If both are NULL, then use all inputs.
- **target.concept**: See `explain`.
- **target.ciu**: See `explain`.
- **color.ramp.below.neutral**: Color ramp function as returned by function `colorRamp()`. Default color ramp is from red3 to yellow.
- **color.ramp.above.neutral**: Color ramp function as returned by function `colorRamp()`. Default color ramp is from yellow to darkgreen.
- **sort**: NULL, "CI" or "CU". No sorting by default, other options are sorting by CI or CU.
- **decreasing**: Set to TRUE for decreasing sort.
- **main, xlab, xlim, ...**: Usual plot parameters, possible to override the default ones provided here if needed.
Details

Usage

barplot.ciu(
  instance,
  ind.inputs=NULL,
  ind.output=1,
  in.min.max.limits=NULL,
  n.samples=100,
  neutral.CU=0.5,
  show.input.values=TRUE,
  concepts.to.explain=NULL,
  target.concept=NULL,
  target.ciu=NULL,
  color.ramp.below.neutral=NULL,
  color.ramp.above.neutral=NULL,
  sort=NULL,
  decreasing=FALSE,
  main=NULL,
  xlab=NULL,
  xlim=NULL,
  ...)

Value

"void", i.e. whatever happens to be result of last instruction.

Author(s)

Kary Främling

See Also

ggplot.col.ciu pie.ciu ciu.new explain

ciu.blackbox.new

---

Description

This method mainly serves as an "interface specification" for objects of class CIU.BlackBox, i.e. it defines what method(s) have to be implemented by any object of class CIU.BlackBox. A CIU.BlackBox object is actually a list.

Usage

ciu.blackbox.new()
Details

An alternative and simpler (but less flexible) way to do the same is to use the predict.function parameter of ciu.new, where predict.function <- function(model, inputs) {predict(model, inputs, n.trees=10000)} would accomplish the same as for the Example below. An example using this approach is also included in Examples.

The advantage of using a CIU.BlackBox wrapper (rather than the simple predict.function approach) is that it is possible to keep object variables or maintain whatever state information might be needed between calls.

The only things that are actually required from a CIU.BlackBox object is:

1. That it is a list with an element called eval.
2. That the value of eval element is a function of the form eval = function(inputs)
3. That it inherits the class CIU.BlackBox.

Value

Object of class CIU.BlackBox.

Author(s)

Kary Främling

Examples

# Create CIU.BlackBox wrapper for Gradient Boosting
library(MASS) # Just in case Boston is not already available
library(gbm)
gbm.ciu.bb <- function(gbm, n.trees=1) {
o.gbm <- gbm
o.n.trees <- n.trees
pub <- list(eval = function(inputs) { predict(o.gbm, inputs, n.trees=o.n.trees) })
class(pub) <- c("CIU.BlackBox", class(pub))
return(pub)
}

# Train and explain. We don't care about training/test sets here.
gbm.Boston <- gbm(medv ~ ., data = Boston, distribution = "gaussian",
n.trees=10000, shrinkage = 0.01, interaction.depth = 4)
gbm.ciu <- gbm.ciu.bb(gbm.Boston, 10000)
ciu <- ciu.new(gbm.ciu, medv~., Boston)
ciu$barplot.ciu(Boston[370,1:13], sort = "CI")

# Same but using 'predict.function' parameter in 'ciu.new'.
# Using 'ggplot.col.ciu' here for a change.
predict.function <- function(model, inputs) {predict(model, inputs, n.trees=10000)}
ciu <- ciu.new(gbm.Boston, medv~., Boston, predict.function=predict.function)
ciu$ggplot.col.ciu(Boston[370,1:13], sort = "CI")
ciu.new

Create CIU object

Description

Sets up a CIU object with the given parameters. CIU objects have "public" and "private" methods. A CIU object is actually a list whose elements are the public functions (methods).

Usage

ciu.new(
  bb,
  formula = NULL,
  data = NULL,
  in.min.max.limits = NULL,
  abs.min.max = NULL,
  input.names = NULL,
  output.names = NULL,
  predict.function = NULL,
  vocabulary = NULL
)

Arguments

bb         Model/"black-box" object. At least all caret models, the lda model from MASS, and the lm model are supported. Otherwise, the prediction function to be used can be gives as value of the predict.function parameter. A more powerful way is to inherit from FunctionApproximator class and implement an "eval" method.

formula    Formula that describes input versus output values. Only to be used together with data parameter.

data       The training data used for training the model. If this parameter is provided, a formula MUST be given also. ciu.new attempts to infer the other parameters from data and formula, i.e. in.min.max.limits, abs.min.max, input.names and output.names. If those parameters are provided, then they override the inferred ones.

in.min.max.limits matrix with one row per output and two columns, where the first column indicates the minimal value and the second column the maximal value for that input.

abs.min.max data.frame or matrix of min-max values of outputs, one row per output, two columns (min, max).

input.names labels of inputs.

output.names labels of outputs.
predict.function

can be supplied if a model that is not supported by ciu should be used. As an example, this is the function for lda:

```r
o.predict.function <- function(model, inputs) {
  pred <- predict(model,inputs)
  return(pred$posterior)
}
```

vocabulary

list of labels/concepts to be used when producing explanations and what combination of inputs they correspond to. Example of two intermediate concepts and a higher-level one that combines them: `list(intermediate.concept1=c(1,2,3),intermediate.concept2=c(4,5),higher.level.concept=c(1,2,3,4,5))`

Details

CIU is implemented in an object-oriented manner, where a CIU object is a list whose methods are made visible as elements of the list. The general way for using CIU objects is to first get a CIU object by calling `ciu.new` as e.g. `ciu <- ciu.new(...)`, then call `ciu.res <- ciu$<method>(...)`. The methods that can be used in `<method>` are:

- `explain`
- `barplot.ciu`
- `ggplot.col.ciu`
- `pie.ciu`
- `plot.ciu`
- `plot.ciu.3D`

"Usage" section is here in "Details" section because Roxygen etc. don’t support documentation of functions within functions.

Value

Object of class CIU.

Author(s)

Kary Främling

References


Examples

# Explaining the classification of an Iris instance with lda model.
# We use a versicolor (instance 100).
library(MASS)
test.ind <- 100
iris_test <- iris[test.ind, 1:4]
iris_train <- iris[-test.ind, 1:4]
iris_lab <- iris[[5]][-test.ind]
model <- lda(iris_train, iris_lab)

# Create CIU object
ciu <- ciu.new(model, Species~., iris)

# This can be used with explain method for getting CIU values
# of one or several inputs. Here we get CIU for all three outputs
# with input feature "Petal.Length" that happens to be the most important.
ciu$explain(iris_test, 1)

# It is, however, more convenient to use one of the graphical visualisations.
# Here's one using ggplot.
ciu$ggplot.col.ciu(iris_test)

ciu$plot.ciu.3D(iris_test,c(3,4),1,main=levels(iris$Species)[1],)
ciu$plot.ciu.3D(iris_test,c(3,4),2,main=levels(iris$Species)[2])
ciu$plot.ciu.3D(iris_test,c(3,4),3,main=levels(iris$Species)[3])

# Same thing with a regression task, the Boston Housing data set. Instance
# #370 has the highest valuation (50k$). Model is gbm, which performs
# decently here. Plotting with "standard" bar plot this time.
library(caret)
library(gbm)

gbm <- train(medv ~ ., Boston, method="gbm", trControl=trainControl(method="cv", number=10))
ciu <- ciu.new(gbm, medv~., Boston)

ciu$barplot.ciu(Boston[370,1:13])
ciu$barplot.ciu(Boston[370,1:13], sort = "CI")

ciu$ggplot.col(Boston[370,1:13])
ciu$pie.ciu(Boston[370,1:13])

ciu$plot.ciu(Boston[370,1:13],13)
ciu.relative

Calculate CIU of a sub-concept/input relative to an intermediate concept (or output).

Description

Calculate CIU of a sub-concept/input relative to an intermediate concept (or output). The parameters must be of class "ciu.result" or a data.frame with compatible columns.

Usage

ciu.relative(sub.ciu.result, sup.ciu.result)

Arguments

sub.ciu.result ciu.result object of sub-concept/input.
sup.ciu.result ciu.result object of intermediate concept/output.

ciu.result.new

CIU result object

Description

Create object of class ciu.result, which stores results of CIU calculations. The explain() method returns a ciu.result object.

Usage

ciu.result.new(ci, cu, cmin, cmax, outval)

Arguments

ci vector of CI values, one per output
cu vector of CU values, one per output
cmin vector of cmin values, one per output
cmax vector of cmax values, one per output
outval vector of black-box output values, one per output

Value

An object of class ciu.result, which is a data.frame with (at least) five columns:

- CI values: one row per output of the black-box model
- CU values: one row per output of the black-box model
- cmin values: one row per output of the black-box model
- cmax values: one row per output of the black-box model
- outval values: one row per output of the black-box model
**Author(s)**

Kary Främling

---

**explain**

*Calculate CIU for specific instance*

**Description**

Calculate Contextual Importance (CI) and Contextual Utility (CU) for an instance (Context) using the given "black-box" model. First get a CIU object by calling `ciu.new` as e.g. `ciu <- ciu.new(...)`, then call as `ciu.res <- ciu$explain(...)`. "Usage" section is in "Details" section because Roxygen etc. don’t support documentation of functions within functions.

**Arguments**

- **instance**
  Input values for the instance to explain. Should be a *data.frame* even though a *vector* or *matrix* might work too if input names and other needed metadata can be deduced from the dataset or other parameters given to `ciu.new`.

- **ind.inputs.to.explain**
  *vector* of indices for the inputs to be explained, i.e. for which CIU should be calculated. If NULL, then all inputs will be included.

- **in.min.max.limits**
  *data.frame* or *matrix* with one row per output and two columns, where the first column indicates the minimal value and the second column the maximal value for that output. ONLY NEEDED HERE IF not given as parameter to `ciu.new` or if the limits are different for this specific instance than the default ones.

- **n.samples**
  How many instances to generate for estimating CI and CU. For inputs of type *factor*, all possible combinations of input values is generated, so this parameter only influences how many instances are (at least) generated for continuous-valued inputs.

- **target.concept**
  If provided, then calculate CIU of inputs `ind.inputs.to.explain` relative to the given concept rather than relative to the actual output(s). `ind.inputs.to.explain` should normally be a subset (or all) of the inputs that `target.concept` consists of, even though that not required by the CIU calculation. If a "target.ciu" is provided, then the "target.concept" doesn’t have to be included in the vocabulary gives as parameter to `ciu.new` (at least for the moment).

- **target.ciu**
  CIU result object previously calculated for `target.concept`. If a `target.concept` is provided but `target.ciu=NULL`, then `target.ciu` is estimated by a call to `explain` with the `n.samples` value given as a parameter to this call. It may be useful to provide `target.ciu` if it should be estimated using some other (typically greater) value for `n.samples` than the default one, or if it has already been calculated for some reason.
ggplot.col.ciu

Details

Usage

explain(
  instance,
  ind.inputs.to.explain,
  in.min.max.limits=NULL,
  n.samples=100,
  target.concept=NULL,
  target.ciu=NULL
)

Value

A ciu.result object as returned by ciu.result.new

Author(s)

Kary Främling

See Also

ciu.result.new

description

Create a barplot showing CI as the length of the bar and CU on color scale from red to green, via yellow, for the given inputs and the given output. First get a CIU object by calling ciu.new as e.g. ciu<-ciu.new(...), then call ciu.res<-ciu$barplot.ciu(...). "Usage" section is in 'Details' section because Roxygen etc. don't support documentation of functions within functions.

Arguments

instance Instance to explain. See explain.
ind.inputs vector of indices for the inputs to be included in the plot. If NULL then all inputs will be included.
output.names Vector with names of outputs to include. If NULL (default), then include all.
in.min.max.limits See explain.
n.samples See explain.
neutral.CU Indicates when the Contextual Utility is considered to be "negative". The default value of 0.5 seems quite logical for most cases.
show.input.values
Include input values after input labels or not.

corcepts.to.explain
List of concepts to use in the plot, as defined by vocabulary provided as argument to ciu.new. If ind.inputs=NULL, then use concepts.to.explain instead. If both are NULL, then use all inputs.

target.concept
See explain.
target.ciu
See explain.
low.color
Color to use for CU=0. Default is red.
mid.color
Color to use for CU=neutral.CU. Default is yellow.
high.color
Color to use for CU=1. Default is darkgreen.
sort
NOT USED FOR THE MOMENT! Features are in the same order for all facets, sorted by mean importance over all facets, which feels like a decent behaviour. NULL, "CI" or "CU". No sorting by default, other options are sorting by CI or CU.
decreasing
NOT USED FOR THE MOMENT. Set to TRUE for decreasing sort.
main
Replace default main title of plot.

Details
First get a CIU object by calling ciu.new as e.g. ciu <-ciu.new(...), then call ciu.res <-ciu$ggplot.col.ciu(...).

“Usage” section is here in “Details” section because Roxygen etc. don’t support documentation of functions within functions.

Usage

ggplot.col.ciu(
  instance,
  ind.inputs=NULL,
  output.names=NULL,
  in.min.max.limits=NULL,
  n.samples=100,
  neutral.CU=0.5,
  show.input.values=TRUE,
  concepts.to.explain=NULL,
  target.concept=NULL,
  target.ciu=NULL,
  low.color="red",
  mid.color="yellow",
  high.color="darkgreen",
  sort=NULL,
  decreasing=FALSE,
  main=NULL)

Value
Created ggplot object.
pie.ciu

Author(s)
Kary Främling

See Also
barplot.ciu ciu.new explain

---

table

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pie.ciu</td>
<td>Pie Chart CIU explanation for specific instance</td>
</tr>
</tbody>
</table>

Description
Create a pie chart showing CI as the area of the slice and CU on color scale from red to green, via yellow, for the given inputs and the given output. First get a CIU object by calling ciu.new as e.g. ciu <- ciu.new(...), then call ciu.res <- ciu$pie.ciu(...). "Usage" section is in "Details" section because Roxygen etc. don't support documentation of functions within functions.

Arguments
Same as for barplot.ciu.

Details

Usage

```
pie.ciu(
  instance,
  ind.inputs=NULL,
  ind.output=1,
  in.min.max.limits=NULL,
  n.samples=100,
  neutral.CU=0.5,
  show.input.values=TRUE,
  concepts.to.explain=NULL,
  target.concept=NULL,
  target.ciu=NULL,
  color.ramp.below.neutral=NULL,
  color.ramp.above.neutral=NULL,
  sort=NULL,
  decreasing=FALSE,
  main=NULL,
  xlab=NULL,
  xlim=NULL,
  ...
)
```

Value

"void", i.e. whatever happens to be result of last instruction.
**Author(s)**

Kary Främling

**See Also**

barplot.ciu ciu.new explain

| plot.ciu | Plot output value as a function of one input for a specific instance |

**Description**

Plot how the value of one output changes as a function of one input, as a line chart. The current input/output values are indicated by a red dot. The values of all other inputs are the ones given by the instance parameter. This method is not specific for CIU but it allows to study the behaviour of the underlying "black-box model". It also makes it easy to understand how CI and CU values have been calculated.

**Arguments**

- **instance**: Instance to explain. See explain.
- **ind.input**: Index of the input to plot
- **ind.output**: Index of the output to plot
- **in.min.max.limits**: See explain.
- **n.points**: The number of points to use on X-axis for plotting.
- **main, xlab, ylab, ylim, ...**: Usual plot parameters, possible to override the default ones provided here if needed.

**Details**

First get a CIU object by calling ciu.new as e.g. `ciu <- ciu.new(...)`, then call `ciu.res <- ciu$plot.ciu(...)`. "Usage" section is here in "Details" section because Roxygen etc. don’t support documentation of functions within functions. **Usage**

```r
plot.ciu(
  instance,
  ind.input,
  ind.output,
  in.min.max.limits=NULL,
  n.points=40,
  main=NULL,
  xlab=NULL,
  ylab=NULL,
  ylim=NULL,
  ...
)
```
Value

"void", i.e. whatever happens to be result of last instruction.

Author(s)

Kary Främling

---

plot.ciu.3D  

Plot output value as a function of two inputs for a specific instance

Description

Plot how the value of one output changes as a function of two inputs using `persp`. The current input/output values are indicated by a red dot. The values of all other inputs are the ones given by the `instance` parameter. This method is not specific for CIU but it allows to study the behaviour of the underlying "black-box model". It also makes it easy to understand how CI and CU values have been calculated.

Arguments

- `instance`: Instance to explain. See `explain`.
- `ind.inputs`: Index of the inputs to plot
- `ind.output`: Index of the output to plot
- `in.min.max.limits`: See `explain`.
- `n.points`: The number of points to use on X/Y-axis for plotting.
- `main, xlab, ylab, zlab, zlim, ...`: Usual plot parameters, possible to override the default ones provided here if needed.

Details

First get a CIU object by calling `ciu.new` as e.g. `ciu <- ciu.new(...)`, then call `ciu.res <- ciu$plot.ciu.3D(...)`. "Usage" section is here in "Details" section because Roxygen etc. don't support documentation of functions within functions.

Usage

```r
plot.ciu.3D(
  instance,
  ind.inputs,
  ind.output,
  in.min.max.limits=NULL,
  n.points=40,
  main=NULL,
  xlab=NULL,
  ylab=NULL,
  zlab=NULL,
  zlim=NULL,
  ...
)```
Value

"void", i.e. whatever happens to be result of last instruction.

Author(s)

Kary Främling
Index

_INDEX (ciu-package), 2

barplot.ciu, 3, 7, 13, 14

.ciu-package, 2
.ciu.blackbox.new, 4
.ciu.new, 2–5, 6, 7, 10–15
.ciu.relative, 9
.ciu.result.new, 9, 11

data.frame, 6, 9, 10

explain, 3, 4, 7, 10, 11–15
explain(), 9

factor, 10

ggplot.col.ciu, 4, 7, 11

list, 4–7

matrix, 6, 10

persp, 15

pie.ciu, 4, 7, 13
plot.ciu, 7, 14
plot.ciu.3D, 7, 15

vector, 3, 10, 11