Package ‘deepboost’

November 8, 2017

**Type** Package  
**Title** Deep Boosting Ensemble Modeling  
**Version** 0.1.6  
**Date** 2017-11-08  
**Author** Daniel Marcous [aut, cre], Yotam Sandbank [aut], Google Inc. [cph]  
**Maintainer** Daniel Marcous <dmarcous@gmail.com>  
**URL** https://github.com/dmarcous/CRAN_deepboost  
**BugReports** https://github.com/dmarcous/CRAN_deepboost/issues  
**License** Apache License (== 2.0)  
**LazyData** TRUE  
**Suggests** testthat, ada, caret  
**Depends** R (>= 3.1)  
**Imports** Rcpp (>= 0.12.2), methods  
**LinkingTo** Rcpp  
**RoxygenNote** 6.0.1  
**NeedsCompilation** yes  
**Repository** CRAN  
**Date/Publication** 2017-11-08 18:22:33 UTC

R topics documented:

- adult ................................................................. 2  
- australian ........................................................... 3  
- banana .............................................................. 3  
- bupa ................................................................. 4


### Description

A dataset containing adult population personal details

### Usage

```r
adult
```

### Format

A data frame with 32560 rows and 15 variables:

- **Adm.clerical**  unknown
- **Bachelors**  person is a bachelor
- **Male**  gender
- **Never.married**  did person marry?
- **Not.in.family**  is person a part of a family
- **State.gov**  state
- **United.States**  is from the united states
- **White**  is white
- **X..50K**  unknown
- **X0**  unknown
Source

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

---

<table>
<thead>
<tr>
<th>Australian</th>
<th>australian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Australian</td>
</tr>
<tr>
<td>Usage</td>
<td>australian</td>
</tr>
<tr>
<td>Format</td>
<td>An object of class data.frame with 689 rows and 15 columns.</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>banana</th>
<th>banana</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>banana</td>
</tr>
<tr>
<td>Usage</td>
<td>banana</td>
</tr>
<tr>
<td>Format</td>
<td>An object of class data.frame with 5299 rows and 3 columns.</td>
</tr>
</tbody>
</table>
**bupa**

**Description**

bupa

**Usage**

bupa

**Format**

An object of class `data.frame` with 344 rows and 7 columns.

---

**coli2000**

**Description**

coli2000

**Usage**

coli2000

**Format**

An object of class `data.frame` with 9821 rows and 86 columns.

---

**deepboost**

*Main function for deepboost model creation*

**Description**

Main function for deepboost model creation

**Usage**

```
depboost(formula, data, instance_weights = NULL, tree_depth = 5, num_iter = 1, beta = 0, lambda = 0.05, loss_type = "l", verbose = TRUE)
```
Deepboost-class

Arguments

- **formula**: A R Formula object see: ?formula
- **data**: A data.frame of samples to train on
- **instance_weights**: The weight of each example
- **tree_depth**: maximum depth for a single decision tree in the model
- **num_iter**: number of iterations = number of trees in ensemble
- **beta**: regularisation for scores (L1)
- **lambda**: regularisation for tree depth
- **loss_type**: - "l" logistic, "e" exponential
- **verbose**: - print extra data while training TRUE / FALSE

Value

A trained Deepboost model

Examples

```r
deepboost(y ~ ., data.frame(x1=rep(c(0,0,1,1),2),x2=rep(c(0,1,0,1),2),y=factor(rep(c(0,0,0,1),2))), num_iter=1)
deepboost(y ~ ., data.frame(x1=rep(c(0,0,1,1),22),x2=rep(c(0,1,0,1),2),y=factor(rep(c(0,0,0,1),2))), num_iter=2, beta=0.1, lambda=0.00125)
```

Deepboost-class

An S4 class to represent a deepboost model.

Description

An S4 class to represent a deepboost model.

Slots

- **tree_depth**: maximum depth for a single decision tree in the model
- **num_iter**: number of iterations = number of trees in ensemble
- **beta**: regularisation for scores (L1)
- **lambda**: regularisation for tree depth
- **loss_type**: "l" logistic, "e" exponential
- **verbose**: print extra data while training TRUE / FALSE
- **examples**: data.frame with instances used for model training
- **model**: Deepboost model as used by C code serialised to R List
- **classes**: a vector of factors representing the classes used for classification with this model
deepboost.default  
Main function for deepboost model creation

Description

Main function for deepboost model creation

Usage

depboost.default(x, y, instance_weights = NULL, tree_depth = 5,  
num_iter = 1, beta = 0, lambda = 0.05, loss_type = "l",  
verbose = TRUE)

Arguments

x  
A data.frame of samples' values

y  
A data.frame of samples's labels

instance_weights  
The weight of each example

tree_depth  
maximum depth for a single decision tree in the model

num_iter  
number of iterations = number of trees in ensemble

beta  
regularisation for scores (L1)

lambda  
regularisation for tree depth

loss_type  
- "l" logistic, "e" exponential

verbose  
- print extra data while training TRUE / FALSE

Value

A trained Deepboost model

Examples

depboost.default(data.frame(x1=rep(c(0,0,1,1,2)),x2=rep(c(0,1,0,1,2))),  
factor(rep(c(0,0,0,1,2)),num_iter=1)
depboost.default(data.frame(x1=rep(c(0,0,1,1,2)),x2=rep(c(0,1,0,1,2))),  
factor(rep(c(0,0,0,1,2)),  
nue_iter=2, beta=0.1, lambda=0.00125)
deepboost.evaluate  Evaluates and prints statistics for a deepboost model

Description

Evaluates and prints statistics for a deepboost model

Usage

deepboost.evaluate(object, data)

Arguments

object  A Deepboost S4 class object
data    a data.frame object to evaluate with the model

Value

a list with model statistics - error, avg_tree_size, num_trees

Examples

dpb <- deepboost(y ~ .
  data.frame(x1=rep(c(0,0,1,1,2),x2=rep(c(0,1,0,1,2),y=factor(rep(c(0,0,0,1,2)))),
  num_iter=2,tree_depth=2)
depth evaluate(dpb, data.frame(x1=rep(c(1,1,1,0,2),x2=rep(c(1,1,1,1,2))))

deepboost.formula  Main function for deepboost model creation, using a formula

Description

Main function for deepboost model creation, using a formula

Usage

deepboost.formula(formula, data, instance_weights = NULL, tree_depth = 5,
  num_iter = 1, beta = 0, lambda = 0.05, loss_type = "1",
  verbose = TRUE)
Arguments

- **formula**: A R Formula object see: ?formula
- **data**: A data.frame of samples to train on
- **instance_weights**: The weight of each example
- **tree_depth**: maximum depth for a single decision tree in the model
- **num_iter**: number of iterations = number of trees in ensemble
- **beta**: regularisation for scores (L1)
- **lambda**: regularisation for tree depth
- **loss_type**: - "l" logistic, "e" exponential
- **verbose**: - print extra data while training TRUE / FALSE

Value

A trained Deepboost model

Examples

```
deepboost.formula(y ~ , 
data.frame(x1=rep(c(0,0,1,1),2),x2=rep(c(0,1,0,1),2),y=factor(rep(c(0,0,0,1),2))),
num_iter=1)
deeboost.formula(y ~ ,
data.frame(x1=rep(c(0,0,1,1),2),x2=rep(c(0,1,0,1),2),y=factor(rep(c(0,0,0,1),2))),
num_iter=2, beta=0.1, lambda=0.00125)
```

deeboost.gridSearch (Returns optimised parameter list for deepboost model on given data)

Description

Returns optimised parameter list for deepboost model on given data

Usage

```
deeboost.gridSearch(formula, data, k = 10, seed = 666, logging_level = 1)
```

Arguments

- **formula**: A R Formula object see: ?formula
- **data**: input data.frame as training for model
- **k**: number of folds (default = 10) for cross validation optimisation
- **seed**: for random split to train / test (default 666)
- **logging_level**: print extra data while training 0 - no data, 1 - gridSearch data (default), 2 - all data
**deepboost.predict**

**Details**

Finds optimised parameters for deepboost training, using grid search techniques over:
- predefined, battle tested parameter possible values
- cross validation over k folds

**Value**

vector with average accuracy for chosen parameters, and a list of the best parameter combination:
(accuracy, (num_iter, beta, lambda, loss_type))

**Examples**

depth.boost.gridSearch(y ~ .,
data.frame(x1=rep(c(0,0,1,1),2),x2=rep(c(0,1,0,1),2),y=factor(rep(c(0,0,0,1),2))), k=2)

depth.boost.predict

Predicts instances responses based on a deepboost model

**Description**

Predicts instances responses based on a deepboost model

**Usage**

depth.boost.predict(object, newdata, type = "terms")

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>A Deepboost S4 class object</td>
</tr>
<tr>
<td>newdata</td>
<td>A data.frame to predict responses for</td>
</tr>
<tr>
<td>type</td>
<td>Type of prediction: &quot;terms&quot; - for class labels, &quot;response&quot; for probabilities</td>
</tr>
</tbody>
</table>

**Value**

A vector of responses

**Examples**

dpb <- depth.boost(y ~ .,
data.frame(x1=rep(c(0,0,1,1),2),x2=rep(c(0,1,0,1),2),y=factor(rep(c(0,0,0,1),2))),
num_iter=2,tree_depth=2)
depth.boost.predict(dpb,data.frame(x1=rep(c(1,1,1,0),5),x2=rep(c(1,1,1,1),5)))
deepboost.train

---

**deepboost.print**

*Evaluates and prints statistics for a deepboost model on the train set*

**Description**

Evaluates and prints statistics for a deepboost model on the train set

**Usage**

```r
deepboost.print(object)
```

**Arguments**

- `object` A Deepboost S4 class object

**Value**

List with model_statistics to console the model evaluation string

**Examples**

```r
dpb <- deepboost(y ~ ., data.frame(x1=rep(c(0,0,1,1),2),x2=rep(c(0,1,0,1),2),y=factor(rep(c(0,0,0,1),2))), num_iter=2, tree_depth=2)
deepboost.print(dpb)
```

---

**deepboost.train**

*Trains a deepboost model*

**Description**

Trains a deepboost model

**Usage**

```r
deepboost.train(object, data, tree_depth, num_iter, beta, lambda, loss_type, verbose, classes)
```

**Arguments**

- `object` A Deepboost S4 class object
- `data` input data.frame as training for model
- `tree_depth` maximum depth for a single decision tree in the model
- `num_iter` number of iterations = number of trees in ensemble
- `beta` regularisation for scores (L1)
lambda: regularisation for tree depth
loss_type: "l" logistic, "e" exponential
verbose: print extra data while training TRUE / FALSE
classes: a vector of factors representing the classes used for classification with this model

Details

(beta,lambda) = (0,0) - adaboost, (>0,0) - L1, (0,>0) deepboost, (>0, >0) deepboost+L1

Value

A trained Deepboost model

Description

haberman

Usage

haberman

Format

An object of class data.frame with 305 rows and 4 columns.

Description

heart

Usage

heart

Format

An object of class data.frame with 269 rows and 14 columns.
Description

Predicted values based on deepboost model object.

Usage

```r
## S4 method for signature 'Deepboost'
predict(object, newdata, type = "terms")
```
show,Deepboost-method

Arguments

- object: Object of class "Deepboost"
- newdata: takes data.frame.
- type: Type of prediction

Details

The option ntreelimit purpose is to let the user train a model with lots of trees but use only the first trees for prediction to avoid overfitting (without having to train a new model with less trees).

Examples

dpb <- deepboost(y ~ ., 
data.frame(x1=rep(c(0,0,1,1),2),x2=rep(c(0,1,0,1),2),y=factor(rep(c(0,0,0,1),2))), 
num_iter=2,tree_depth=2)
predict(dpb,data.frame(x1=rep(c(1,1,1,0),2),x2=rep(c(1,1,1,1),2)))
Description

sonar

Usage

sonar

Format

An object of class `data.frame` with 207 rows and 61 columns.
Index

*Topic datasets
  adult, 2
  australian, 3
  banana, 3
  bupa, 4
  coli2000, 4
  haberman, 11
  heart, 11
  magic, 12
  pima, 12
  sonar, 14

  adult, 2
  australian, 3

  banana, 3
  bupa, 4

  coli2000, 4

deepboost, 4
  Deepboost-class, 5
  deepboost.default, 6
  deepboost.evaluate, 7
  deepboost.formula, 7
  deepboost.gridSearch, 8
  deepboost.predict, 9
  deepboost.print, 10
  deepboost.train, 10

  haberman, 11
  heart, 11

  magic, 12

  pima, 12
  predict, Deepboost-method, 12

  show, Deepboost-method, 13
  sonar, 14