Package 'IBHM'

February 19, 2015

Type Package

Title Approximation using the IBHM method

Version 1.1-11

Date 2014-01-18

Author Pawel Zawistowski

Maintainer Pawel Zawistowski <dratewka@gmail.com>

Description Implementation of an incremental model construction method called IBHM which stands for Incrementally Built Heterogeneous Model. The method is designed for solving real number approximation problems in a highly automated fashion.

License GPL (>= 2)

Depends R(>= 2.15.2), compiler, DEoptim (>= 2.2-1), cmaes (>= 1.0-11), Rcpp (>= 0.10.3), methods (>= 3.0.1)

LinkingTo Rcpp

RcppModules evaluator

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-01-19 00:29:25

R topics documented:

IBHM-package *Approximation using the IBHM method*

Description

Implementation of an incremental model construction method called IBHM which stands for Incrementally Built Heterogeneous Model. The method is designed for solving real-number approximation problems in a highly automated fashion and should be considered one of black-box modelling approaches such as Artificial Neural Networks or Support Vector Machines.

Details

Package:	IBHM
Type:	Package
Version:	1.0
Date:	2013-03-21
License:	GPL (>= 2)

The package may be used to produce small, robust computational models using real valued empirical data. The most important functions are `TrainIBHM` which constructs an IBHM model and `ConfigureIBHM` which sets the methods parameters.

Author(s)

Pawel Zawistowski

Maintainer: Pawel Zawistowski <dratewka@gmail.com>

References

Zawistowski, P. and Arabas, J.: "Benchmarking IBHM method using NN3 competition dataset." In *Proc. 6th int. conf. on Hybrid artificial intelligent systems* - Vol. 1, HAIS'11, pp 263–270, 2011. Springer-Verlag.

ConfigureIBHM *ConfigureIBHM*

Description

Creates a configuration object for `TrainIBHM` function which creates IBHM approximation models.

Usage

```
ConfigureIBHM( stop.criterion = IterationSC(3),
         weighting.function = function(y, w.par){ 0.01+dnorm(y,sd=abs(w.par))},
         scal.optim = 'multi.CMAES',
       scal.optim.params = list(retries=3, inner=list(maxit=50, stopfitness=-1)),
         scal.candidates = c('dot.pr','radial','root.radial'),
         activ.optim = 'multi.CMAES',
       activ.optim.params = list(retries=3, inner=list(maxit=100, stopfitness=-1)),
         activ.candidates = c('tanh','logsig','lin'),
         jit=TRUE,
         verbose=FALSE,
         final.estimation = 'all',
         final.estimation.x = NULL,
         final.estimation.y = NULL,
         final.estimation.maxit = 100
       )
```

Arguments

`stop.criterion`	The stop criterion for the model construction process. Possible values include objects created using `IterationSC` (the default) or `ValidationSC`. For simplicity, the default value causes the method to construct a model with a fixed number of components (3), however it's actually best to use `ValidationSC` criterion with a separate validation set.
`weighting.function`	Definition of the weighting function used during model construction. This function puts emphasis on local features of the approximated function (see details).
`scal.optim`	The optimization method used to estimate scalarization functions' parameters . Possible values are: `CMAES`, `multi.CMAES`(default), `DE`, `multi.DE`. The parameter values set the optimization methods as follows: `CMAES` uses covariance matrix evolution strategy implemented in the `cmaes` package, `multi.CMAES` is multistart `cmaes`, `DE` uses differential evolution implemented in `DEoptim` package, `multi.DE` is multistart `DE`.
`scal.optim.params`	The parameters passed to the optimization method used to estimate scalarization functions' parameters. In case of `CMAES` this should be a list, in case of `DE` this should be a `DEoptim::DEoptim.control` object. In case of multistart versions of optimization methods this parameter is a `list` with two fields: `retries` denoting the number of restarts and `inner` containing the parameters passed during each restart to the underlying optimization method.
`scal.candidates`	Candidate scalarization functions (see details).
`activ.optim`	The optimization method used to estimate activation functions' parameters - see description of `scal.optim`.
`activ.optim.params`	The parameters passed to the optimization method used to estimate activation functions' parameters - see description of the `scal.optim.params`.

`activ.candidates`
: Candidate activation functions (see details).

`jit`
: Enables the just-in-time compilation feature provided by the `compiler` package.

`verbose`
: Enables verbose output (disabled by default).

`final.estimation`
: The type of final parameter estimation step. Possible values are: `weights` and `all` for all parameters.

`final.estimation.x`
: The `x` values used during the final estimation step. If not given then the training data is used.

`final.estimation.y`
: The `y` values used during the final estimation step. If not given then the training data is used.

`final.estimation.maxit`
: The number of iterations (of the optimizer) during final parameter estimation.

Details

The model constructed by IBHM has the following form:

$$\hat{f}(x) = w_0 + \sum_{i=1}^{m} w_i g(a_i h(x, d_i) + b_i) \ ,$$

where $h : R^n \rightarrow R$ is a scalarization function, $g : R \rightarrow R$ is an activation function, d_i is a parameter vector, and a_i, b_i, w_i are scalar parameters.

The parameter estimation is based on optimizing weighted correlation measures between the model output and the approximation residual. This allows for an iterative model construction process which estimates both model structure and parameter values. For more details see [Zawistowski and Arabas].

Value

A configuration object for `TrainIBHM`.

References

Zawistowski, P. and Arabas, J.: "Benchmarking IBHM method using NN3 competition dataset." In *Proc. 6th int. conf. on Hybrid artificial intelligent systems* - Vol. 1, HAIS'11, pp 263–270, 2011. Springer-Verlag.

See Also

`TrainIBHM`, `ValidationSC`,`IterationSC`

Examples

```
x <- seq(-3,3,length.out=400)
y <- tanh(x)

x.val <- runif(50,min=-6,max=6)
y.val <- tanh(x.val)

m <- TrainIBHM(x,y, ConfigureIBHM(  scal.candidates = 'dot.pr',
                                    activ.candidates = 'tanh',
                                    stop.criterion = ValidationSC(x.val, y.val)))

summary(m)
plot(y.val,predict(m,x.val),asp=1)
```

IterationSC *IterationSC*

Description

An iteration counting stop criterion for TrainIBHM function creating IBHM approximation models. Should be passed to ConfigureIBHMwhile creating a configuration object.

Usage

```
IterationSC(max.iterations)
```

Arguments

max.iterations	The number of iterations after which model construction should be stopped.

See Also

ValidationSC,ConfigureIBHM, TrainIBHM

Examples

```
x.train <-seq(-2,2,length.out=100)
y.train <-tanh(x.train)

m <- IBHM::TrainIBHM(x.train,y.train, ConfigureIBHM(stop.criterion=IterationSC(2)))
length(m)
```

length.IBHM *length.IBHM*

Description

Returns the number of components in a IBHM model.

Usage

```
## S3 method for class 'IBHM'
length(x)
```

Arguments

x	An object inheriting from class "IBHM".

Value

The number of components in the given IBHM model.

See Also

TrainIBHM

Examples

```
x.train <-seq(-2,2,length.out=100)
y.train <-tanh(x.train)

m <- IBHM::TrainIBHM(x.train,y.train)
length(m)
```

predict.IBHM *predict.IBHM*

Description

Predicted values based on IBHM model object.

Usage

```
## S3 method for class 'IBHM'
predict(object, x, ...)
```

Arguments

object	Object of class inheriting from "ibhm"
x	An object for which the predictions will be calculated - should be convertible to a matrix. If not given the values from the training data are used.
...	Further arguments.

Value

predict.IBHM returns a vector of predicted values with length corresponding to the number of rows of the x parameter.

See Also

TrainIBHM

Examples

```
x.train <-seq(-2,2,length.out=100)
y.train <-tanh(x.train)

m <- IBHM::TrainIBHM(x.train,y.train)
cat('Train MSE:',mean((y.train-predict(m))^2),'\n')

x.test <- runif(100, min=-4, max=4)
yh.test <- predict(m, x.test)

plot(x.test, yh.test)
```

---

summary.IBHM *summary.IBHM*

Description

summary method for class "IBHM".

Usage

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

## S3 method for class 'summary.IBHM'
print(x, ...)
```

Arguments

object	An object of class "IBHM", usually created using TrainIBHM.
x	An object of class "summary.IBHM" created using summary.
...	Further arguments.

Value

An object of class 'summary.IBHM' containing the following fields:

`model`	Equation stating the obtained model in human readable form.
`model.size`	Number of components in the model.
`TrainSize`	Size of the training set used to construct the model.
`TrainDim`	Number of input attributes in the training set.
`mse`	Mean squared error.
`se.sd`	Standard deviation of the squared error.
`rmse`	Root mean squared error.
`cor`	Linear correlation between the actual and predicted values on the train data set.

See Also

`TrainIBHM`

Examples

```
x.train <-seq(-2,2,length.out=100)
y.train <-tanh(x.train)

m <- TrainIBHM(x.train,y.train)
summary(m)
```

`TrainIBHM` *TrainIBHM*

Description

Trains an IBHM model using training data with input variables `x` and a single output variable `y`.

Usage

```
TrainIBHM(x, y, config = ConfigureIBHM())
```

Arguments

`x`	Training data input variables. Should be convertible to a `matrix` with each row corresponding to a single data point.
`y`	Training data output variable. Should be convertible to a `matrix` with a single column and the number of rows equal to the number of rows of `x`.
`config`	A configuration object created using `ConfigureIBHM`. Contains settings such as the stop criterion, optimization method parameters etc.

Value

The created model which is an object of class "IBHM".

References

Zawistowski, P. and Arabas, J.: "Benchmarking IBHM method using NN3 competition dataset." In *Proc. 6th int. conf. on Hybrid artificial intelligent systems* - Vol. 1, HAIS'11, pp 263–270, 2011. Springer-Verlag.

See Also

ConfigureIBHM,summary.IBHM,predict.IBHM

Examples

```
# Training data
x <- seq(-3,3,length.out=400)
y <- tanh(x)

# A held out validation set for the stop criterion
x.val <- runif(50,min=-6,max=6)
y.val <- tanh(x.val)


# Training the model using the validation set to prevent overfitting
m <- TrainIBHM(x,y,
          ConfigureIBHM(stop.criterion = ValidationSC(x.val, y.val))
     )

summary(m)
plot(y.val,predict(m,x.val),asp=1)
```

ValidationSC *ValidationSC*

Description

A validation set based stop criterion for TrainIBHM function creating IBHM approximation models. Should be passed to ConfigureIBHMwhile creating a configuration object.

Usage

```
ValidationSC(x, y)
```

Arguments

x	Validation set input arguments, should be convertible to a matrix.
y	Validation set predicted argument, should be convertible to a single column matrix.

Details

The criterion is checked after each iteration and the current model is used to predict values on the validation data set. When the error increases in comparison to the previous iteration, the construction process is stopped, and the changes in the model from the last iteration are undone.

See Also

IterationSC,ConfigureIBHM, TrainIBHM

Examples

```
# Training data
  x <- seq(-3,3,length.out=400)
  y <- tanh(x)

  # A held out validation set for the stop criterion
  x.val <- runif(50,min=-6,max=6)
  y.val <- tanh(x.val)


  # Training the model using the validation set to prevent overfitting
  m <- TrainIBHM(x,y,
            ConfigureIBHM(stop.criterion = ValidationSC(x.val, y.val))
       )

  summary(m)
  plot(y.val,predict(m,x.val),asp=1)
```

Index