Package: EnsemblePCReg (via r-universe)

August 28, 2024

Type Package

Title Extensible Package for Principal-Component-Regression-Based Heterogeneous Ensemble Meta-Learning

Version 1.1.4

Date 2022-04-18

Author Mansour T.A. Sharabiani, Alireza S. Mahani

Maintainer Alireza S. Mahani <alireza.s.mahani@gmail.com>

Description Extends the base classes and methods of 'EnsembleBase' package for Principal-Components-Regression-based (PCR) integration of base learners. Default implementation uses cross-validation error to choose the optimal number of PC components for the final predictor. The package takes advantage of the file method provided in 'EnsembleBase' package for writing estimation objects to disk in order to circumvent RAM bottleneck. Special save and load methods are provided to allow estimation objects to be saved to permanent files on disk, and to be loaded again into temporary files in a later R session. Users and developers can extend the package by extending the generic methods and classes provided in 'EnsembleBase' package as well as this package.

License GPL (>= 2)

Depends EnsembleBase, methods

Imports parallel

Suggests R.rsp

VignetteBuilder R.rsp

NeedsCompilation no

Date/Publication 2022-04-18 21:54:29 UTC

Repository https://asmahani.r-universe.dev

RemoteUrl https://github.com/cran/EnsemblePCReg

RemoteRef HEAD

RemoteSha 6bcd66eec44395b1256abc465410c155f87d36b9

Contents

`epcreg` *Principal-Components-Regression-Based (PCR) Integration of Base Learners for Ensemble Learning*

Description

This function applies PCR to predictions from regression base learners to produce an ensemble prediction. Number of PC's used in the PCR algorithm is determined by minimizing the cross-validation error. The data partition for the integration phase does not have to be the same as the partition(s) used to generate the base learners. Functions from **EnsembleBase** are used for training and prediction of base learners. Also, base classes and generic methods of the same package are extended to support PCR integration.

Usage

```
epcreg(formula, data
  , baselearner.control=epcreg.baselearner.control()
  , integrator.control=epcreg.integrator.control()
  , ncores=1, filemethod=FALSE, print.level=1
  , preschedule = TRUE
  , schedule.method = c("random", "as.is", "task.length")
  , task.length
)
```

Arguments

`formula`	Formula expressing response variable and covariates.
`data`	Data frame containing the response variable and covariates.

`baselearner.control`
: Control structure determining the base learners, their configurations, and data partitioning details. See `epcreg.baselearner.control`.

`integrator.control`
: Control structure governing integrator behavior. See `epcreg.integrator.control`.

`ncores`
: Number of cores used for parallel training of base learners.

`filemethod`
: Boolean flag indicating whether or not to save estimation objects to disk or not. Using `filemethod=T` reduces RAM pressure.

`print.level`
: Controlling verbosity level.

`preschedule`
: Boolean flag, indicating whether base learner training jobs must be scheduled statically (`TRUE`) or dynamically (`FALSE`).

`schedule.method`
: Method used for scheduling tasks on threads. In "as.is" tasks are assigned to threads in a round-robin fashion for static scheduling. In dynamic scheduling, tasks form a queue without any re-ordering. In "random", tasks are first randomly shuffled, and the rest is similar to "as.is". In "task.length", a heuristic algorithm is used in static scheduling for assigning tasks to threads to minimize load imbalance, i.e. make total task lengths in threads roughly equal. In dynamic scheduling, tasks are sorted in descending order of expected length to form the task queue.

`task.length`
: Vector of estimated task lengths, to be used in the "task.length" method of scheduling.

Value

An object of classes `epcreg` (if `filemethod==TRUE`, also has class of `epcreg.file`), a list with the following elements:

`call`
: Copy of function call.

`formula`
: Copy of formula argument in function call.

`instance.list`
: An object of class `Instance.List`, containing all permutations of base learner configurations and random data partitions generated in the body of the function.

`integrator.config`
: Copy of configuration object passed to the integrator. Object of class `Regression.Integrator.PCR.SelM`

`method`
: Integration method. Currently, only "default" is supported.

`est`
: A list with these elements: 1) `baselearner.cv.batch`, an object of class `Regression.CV.Batch.FitObj` containing the fit object from CV batch training of base learners; 2) `integrator`, an object of class `Regression.Integrator.PCR.SelMin.FitObj` containing the fit object returned by the integrator.

`y`
: Copy of response variable vector.

`pred`
: Within-sample prediction of the ensemble model.

`filemethod`
: Copy of passed-in `filemethod` argument.

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

See Also

epcreg.baselearner.control, epcreg.integrator.control, Instance.List, Regression.Integrator.PCR.SelMin
Regression.CV.Batch.FitObj, Regression.Batch.FitObj, Regression.Integrator.PCR.SelMin.FitObj

Examples

```
data(servo)
myformula <- class~motor+screw+pgain+vgain
perc.train <- 0.7
index.train <- sample(1:nrow(servo), size = round(perc.train*nrow(servo)))
data.train <- servo[index.train,]
data.predict <- servo[-index.train,]
## to run longer test using all 5 default regression base learners
## try: est <- epcreg(myformula, data.train, ncores=2)
est <- epcreg(myformula, data.train, ncores=2
  , baselearner.control=epcreg.baselearner.control(
    baselearners="knn"
    , baselearner.configs = make.configs("knn"
      , config.df = expand.grid(kernel = "rectangular"
        , k = c(5, 10)))))
newpred <- predict(est, data.predict)
```

---

epcreg.baselearner.control

Utility Functions for Configuring Regression Base Learners and Integrator in **EnsemblePCReg** *Package*

Description

Function epcreg.baselearner.control sets up the base learners used in the epcreg call. Function epcreg.integrator.control sets up the PCR integrator.

Usage

```
epcreg.baselearner.control(
  baselearners = c("nnet","rf","svm","gbm","knn")
  , baselearner.configs = make.configs(baselearners, type = "regression")
  , npart = 1, nfold = 5
)
epcreg.integrator.control(errfun=rmse.error, nfold=5, method=c("default"))
```

Arguments

baselearners	Names of base learners used. Currently, regression options available are Neural Network ("nnet"), Random Forest ("rf"), Support Vector Machine ("svm"), Gradient Boosting Machine ("gbm"), and K-Nearest Neighbors ("knn").

`baselearner.configs`
: List of base learner configurations. Default is to call `make.configs` from package **EnsembleBase**.

`npart`
: Number of partitions to train each base learner configuration in a CV scheme.

`nfold`
: Number of folds within each data partition.

`errfun`
: Error function used to compare performance of base learner configurations. Default is to use `rmse.error` from package **EnsembleBase**.

`method`
: Integrator method. Currently, only option is "default", where PCR is performed on all base learner instances, and CV error is used to find the optimal number of PC's. Same CV-based PCR output is used to make final prediction.

Value

Both functions return lists with same element names as function arguments.

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

See Also

`make.configs`, `rmse.error`

`epcreg.save` *Custom Functions for Disk I/O in* **EnsemblePCReg** *Package*

Description

These functions can be used whether `filemethod` flag is set to `TRUE` or `FALSE` during the `epcreg` call. Note that `epcreg.load` 'returns' the estimation object (in contrast to the standard `load` method).

Usage

```
epcreg.save(obj, file)
epcreg.load(file)
```

Arguments

`obj`
: Object of classes "`epcreg`" (and possibly "`epcreg.file`"), typically the output of call to function `epcreg`.

`file`
: Filepath to where `obj` must be saved to / loaded from.

Value

Function `epcreg.load` returns the saved `obj`, with estimation files automatically copied to R temporary directory, and filepaths inside the `obj` fields updated to point to these new filepaths (if `filemethod` was set to `TRUE` in the call to `epcreg`).

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

See Also

epcreg

Examples

```
data(servo)
myformula <- class~motor+screw+pgain+vgain
perc.train <- 0.7
index.train <- sample(1:nrow(servo), size = round(perc.train*nrow(servo)))
data.train <- servo[index.train,]
data.predict <- servo[-index.train,]
est <- epcreg(myformula, data.train, ncores=2
  , baselearner.control=epcreg.baselearner.control(
    baselearners="knn"
    , baselearner.configs = make.configs("knn"
      , config.df = expand.grid(kernel = "rectangular"
        , k = c(5, 10)))), filemethod = TRUE)
epcreg.save(est, "somefile")
rm(est) # alternatively, exit and re-launch R session
est.loaded <- epcreg.load("somefile")
newpred <- predict(est.loaded, data.predict)
file.remove("somefile")
```

---

plot.epcreg *Plot function for* epcreg *model*

Description

Function for generating diagnostics plot for epcreg trained model.

Usage

```
## S3 method for class 'epcreg'
plot(x, ...)
```

Arguments

x	Object of class "epcreg", typically the output of function epcreg.
...	Arguments passed to/from other methods.

Value

Function `plot.epcreg` creates two sub-plots in a figure: 1) a plot of base learner CV errors, with one data point per base learner configuration. The horizontal dotted line indicates the CV error corresponding to the chosen base learner configuration. For "default" method, this is the same as the minimum error of points on this plot; 2) plot of CV error as a function number of PC's used in PCR-based integration. The minimum point of this plot is chosen as the optimal number of PC's and subsequrnyl used for prediction.

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

`predict.epcreg` *Predict method for class* `"epcreg"`

Description

Obtain model predictions from training or new data for `epcreg` model.

Usage

```
## S3 method for class 'epcreg'
predict(object, newdata=NULL, ncores=1
  , preschedule = TRUE, ...)
```

Arguments

`object`	Object of class `"epcreg"`, typically the output of function `epcreg`.
`newdata`	New data frame to make predictions for. If `NULL`, prediction is made for training data.
`ncores`	Number of cores to use for parallel prediction.
`preschedule`	Boolean flag, indicating whether base learner training jobs must be scheduled statically (`TRUE`) or dynamically (`FALSE`).
`...`	Arguments passed to/from other methods.

Value

A vector of length `nrow(newdata)` (or of length of training data if `newdata==NULL`.)

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

`Regression.Integrator.PCR.SelMin.Config-class`
Class `"Regression.Integrator.PCR.SelMin.Config"`

Description

Configuration class for PCR-base integration, where the number of PC's is selected to minimize the cross-validation error of the integrator.

Objects from the Class

Objects can be created by calls of the form `new("Regression.Integrator.PCR.SelMin.Config", ...)`.

Slots

`partition`**:** Object of class `"integer"`, data partition to use for cross-validation selection of optimal PC's in PCR integration. This can be the output of `generate.partition`.

`errfun`**:** Object of class `"function"`, error function to use for selecting best number of PC's.

Extends

Class `"Regression.Integrator.Config"`, directly.

Methods

Regression.Integrator.Fit `signature(object = "Regression.Integrator.PCR.SelMin.Config")`: ...

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

See Also

`generate.partition`

`Regression.Integrator.PCR.SelMin.FitObj-class`
Class `"Regression.Integrator.PCR.SelMin.FitObj"`

Description

Class containing the output of fitting a PCR-based integrator with CV-error minimization method for selecting the number of PC's.

Objects from the Class

Objects can be created by calls of the form `new("Regression.Integrator.PCR.SelMin.FitObj", ...)`.

Slots

`config`: Object of class `"Regression.Integrator.Config"`, containing the error function and the partition to use for training the PCR integrator.

`est`: Object of class `"ANY"`, estimation object that is used for prediction.

`pred`: Object of class `"numeric"`, prediction for training set.

Extends

Class `"Regression.Integrator.FitObj"`, directly.

Methods

No methods defined with class "Regression.Integrator.PCR.SelMin.FitObj" in the signature.

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

See Also

`"Regression.Integrator.FitObj"`

`Regression.Sweep.CV.Fit`

Function for cross-validation based sweep operation.

Description

Perform the same sweep operation on data partitions and assemble the pieces into a complete set.

Usage

```
Regression.Sweep.CV.Fit(config, X, y, partition, print.level = 1)
```

Arguments

`config`	Object of class `Regression.Sweep.Config`, determining the configuration of the underlying sweep operations.
`X`	Matrix of predictors to perform PCR on.
`y`	Vector of response to use during PCR.
`partition`	Data partition used for CV sweep, typically the output of `generate.partition`
`print.level`	Determining verbosity level during function execution.

Value

An object of class `Regression.Sweep.CV.FitObj`.

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

See Also

`Regression.Sweep.CV.FitObj`

`Regression.Sweep.CV.FitObj-class`

Class `"Regression.Sweep.CV.FitObj"`

Description

Class containing output of `Regression.Sweep.CV.Fit` function.

Objects from the Class

Objects can be created by calls of the form `new("Regression.Sweep.CV.FitObj", ...)`.

Slots

`sweep.list`: Object of class `"list"`, list of length equal to number of folds in `partition`. Each element of list is contains the output of `Regression.Sweep.Fit` and has class `Regression.Sweep.FitObj`.

`pred`: Object of class `"matrix"`, containing the matrix of predictions from this operation.

`partition`: Object of class `"OptionalInteger"`, data partition used to perform CV sweep.

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

See Also

`Regression.Sweep.CV.Fit`

`Regression.Sweep.Fit-methods`

~~ Methods for Function `Regression.Sweep.Fit` *in Package* **EnsemblePCReg** *~~*

Description

~~ Methods for function `Regression.Sweep.Fit` in package **EnsemblePCReg** ~~

Methods

`signature(object = "Regression.Sweep.PCR.Config")`

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

`Regression.Sweep.PCR.Config-class`

Class `"Regression.Sweep.PCR.Config"`

Description

Configuration class for PCR sweep operation

Objects from the Class

Objects can be created by calls of the form `new("Regression.Sweep.PCR.Config", ...)`.

Slots

`n`: Object of class `"OptionalNumeric"`, indicating, in this derived class, the maximum number of PC's to perform the PCR sweep for.

Extends

Class `"Regression.Sweep.Config"`, directly.

Methods

Regression.Sweep.Fit `signature(object = "Regression.Sweep.PCR.Config")`: ...

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

`Regression.Sweep.PCR.FitObj-class`

Class `"Regression.Sweep.PCR.FitObj"`

Description

Class containing the output of performing - or fitting - of PCR sweep operation.

Objects from the Class

Objects can be created by calls of the form `new("Regression.Sweep.PCR.FitObj", ...)`.

Slots

`config`: Object of class `"Regression.Sweep.Config"` ~~

`est`: Object of class `"ANY"`, the estimation object needed for prediction.

`pred`: Object of class `"matrix"`, matrix of predictions for training data. Column `n` corresponds to the prediction using PC's from 1 to `n`.

Extends

Class `"Regression.Sweep.FitObj"`, directly.

Methods

No methods defined with class "Regression.Sweep.PCR.FitObj" in the signature.

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

See Also

"Regression.Sweep.FitObj"

---

summary.epcreg *Summary function for* epcreg *model*

Description

Summary function for epcreg model

Usage

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

Arguments

object	Object of class epcreg, usually the output of function epcreg.
...	Arguments passed to/from other functions.

Value

A list with the following elements:

n.instance	Number of base learner instances used in training the model.
maxpc	Maximum number of PC's considered in PCR-based integration of base learners.
index.min	Optimal number of PC's, i.e. what minimizes the CV error.
error.min	Minimum CV error in PCR-based integration, corresponding to index.min number of PC's.
tvec	Vector of task lengths for each base learner instance. This can be passed to task.length argument of epcreg for more efficient task scheduling in parallel training. Only available if epcreg was run in serial mode, i.e., with ncores = 1.

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

Index