Package 'EnsemblePenReg'

September 14, 2016

Type Package

Title Extensible Classes and Methods for Penalized-Regression-Based Integration of Base Learners

Version 0.7

Date 2016-09-13

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

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

Description Extending the base classes and methods of EnsembleBase package for Penalized-Regression-based (Ridge and Lasso) integration of base learners. Default implementation uses cross-validation error to choose the optimal lambda (shrinkage parameter) 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

Imports parallel,methods,glmnet

NeedsCompilation no

Repository CRAN

Date/Publication 2016-09-14 18:50:35

R topics documented:

`ecpreg` *Penalized-Regression-Based (PenReg) Integration of Base Learners for Ensemble Learning*

Description

This function applies Penalized Regression (Lasso and Ridge) to predictions from regression base learners to produce an ensemble prediction. Shrinkage parameter (`lambda`) 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 PenReg integration.

Usage

```
epenreg(formula, data
  , baselearner.control=epenreg.baselearner.control()
  , integrator.control=epenreg.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 `epenreg.baselearner.control`.
`integrator.control`	Control structure governing integrator behavior. See `epenreg.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 `epenreg` (if `filemethod==TRUE`, also has class of `epenreg.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.PenReg.S`

`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.PenReg.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

`epenreg.baselearner.control`, `epenreg.integrator.control`, `Instance.List`, `Regression.Integrator.PenReg.Se`
`Regression.CV.Batch.FitObj`, `Regression.Batch.FitObj`, `Regression.Integrator.PenReg.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 <- epenreg(myformula, data.train, ncores=2)
est <- epenreg(myformula, data.train, ncores=2
  , baselearner.control=epenreg.baselearner.control(baselearners="knn"))
newpred <- predict(est, data.predict)
```

`epenreg.baselearner.control` *Utility Functions for Configuring Regression Base Learners and Integrator in* **EnsemblePenReg** *Package*

Description

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

Usage

```
epenreg.baselearner.control(
  baselearners = c("nnet","rf","svm","gbm","knn")
  , baselearner.configs = make.configs(baselearners, type = "regression")
  , npart = 1, nfold = 5
)
epenreg.integrator.control(errfun=rmse.error, alpha=1.0
  , n=100, 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"), K-Nearest Neighbors ("knn"), Penalized Rergession ("penreg") and Bayesian Additive Regression Trees ("bart"). The last two learners are not include in the default list: "penreg" tends to produce highly correlated, and generally imprecise, predictions and skews the integration stage towards itself. "bart", on the other hand, is quite time- and memory-consuming to train, depsite generally having superior predictive performance. Users with more CPU and memory resources can add "bart" to achieve higher predictive accuracy.
`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**.

alpha	Determining L1 vs L2 penalty. alpha=1 leads to Lasso (L1) shrinkage, while alpha=0.0 leads to Ridge (L2) shrinkage. See glmnet help files for more.
n	Suggested number of lambda's in Penalized Regression. Actual number may be smaller than n, and is determined by the glmnet package.
method	Integrator method. Currently, only option is "default", where PenReg is performed on all base learner instances, and CV error is used to find the optimal shrinkage parameter. Same CV-based PenReg 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

---

epenreg.save *Custom Functions for Disk I/O in* **EnsemblePenReg** *Package*

Description

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

Usage

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

Arguments

obj	Object of classes "epenreg" (and possibly "epenreg.file"), usually the output of call to function epenreg.
file	Filepath to where obj must be saved to / loaded from.

Value

Function epenreg.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.

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

See Also

epenreg

Examples

```
## Not run:
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 <- epenreg(myformula, data.train, ncores=2, filemethod=TRUE
  , baselearner.control=epenreg.baselearner.control(baselearners="knn"))
epenreg.save(est, "somefile")
rm(est) # alternatively, exit and re-launch R session
est.loaded <- epenreg.load("somefile")
newpred <- predict(est.loaded, data.predict)

# can also be used with filemethod set to FALSE
est <- epenreg(myformula, data.train, ncores=2, filemethod=FALSE
  , baselearner.control=epenreg.baselearner.control(baselearners="knn"))
epenreg.save(est, "somefile")
rm(est) # alternatively, exit and re-launch R session
est.loaded <- epenreg.load("somefile")
newpred <- predict(est.loaded, data.predict)

## End(Not run)
```

---

plot.epenreg *Plot function for* epenreg *model*

Description

Function for generating diagnostics plot for epenreg trained model.

Usage

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

Arguments

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

Value

Function `plot.epenreg` 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 of the value of shrinkage parameter (x-axis in log scale). The minimum point of this plot is chosen as the optimal lambda and subsequently used for prediction.

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

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

Description

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

Usage

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

Arguments

`object`	Object of class `"epenreg"`, typically the output of function `epenreg`.
`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.
`...`	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.PenReg.SelMin.Config-class`
Class `"Regression.Integrator.PenReg.SelMin.Config"`

Description

Configuration class for PenReg-base integration, where optimal shrinkage parameter 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.PenReg.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`.

`n`**:** Object of class `"OptionalNumeric"`, indicating, in this derived class, the maximum number of values of `lambda`'s to produce predictions for.

`alpha`**:** Object of class `"numeric"`, indicating the relative strength of L1 (`alpha=1.0`) vs. L2 (`alpha=0.0`) penalty in penalized regression.

`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.PenReg.SelMin.Config")`: ...

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

See Also

`generate.partition`

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

Description

Class containing the output of fitting a PenReg-based integrator with CV-error minimization method for selecting the optimal shrinkage parameter.

Objects from the Class

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

Slots

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

Class `"Regression.Sweep.PenReg.Config"`

Description

Configuration class for PenReg sweep operation

Objects from the Class

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

Slots

`n`: Object of class `"OptionalNumeric"`, indicating, in this derived class, the maximum number of values of `lambda`'s to produce predictions for.

`alpha`: Object of class `"numeric"`, indicating the relative strength of L1 (`alpha=1.0`) vs. L2 (`alpha=0.0`) penalty in penalized regression.

`lambda`: Object of class `"numeric"`, containing the values of shrinkage parameter to generate predictions for. During CV sweep, this parameter is determined in the first fold and passed on to the remaining folds.

Extends

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

Methods

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

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

`Regression.Sweep.PenReg.FitObj-class`
Class `"Regression.Sweep.PenReg.FitObj"`

Description

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

Objects from the Class

Objects can be created by calls of the form `new("Regression.Sweep.PenReg.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.PenReg.FitObj" in the signature.

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

See Also

`"Regression.Sweep.FitObj"`

Index