Super Learner Prediction Function

**Description**

A Prediction Function for the Super Learner. The SuperLearner function takes a training set pair (X,Y) and returns the predicted values based on a validation set.

**Usage**

SuperLearner(Y, X, newX = NULL, family = gaussian(), SL.library,

method = "method.NNLS", id = NULL, verbose = FALSE,

control = list(), cvControl = list(), obsWeights = NULL)

**Arguments**

|  |  |
| --- | --- |
| Y | The outcome in the training data set. Must be a numeric vector. |
| X | The predictor variables in the training data set, usually a data.frame. |
| newX | The predictor variables in the validation data set. The structure should match X. If missing, uses X for newX. |
| SL.library | Either a character vector of prediction algorithms or a list containing character vectors. See details below for examples on the structure. A list of functions included in the SuperLearner package can be found with listWrappers(). |
| verbose | logical; TRUE for printing progress during the computation (helpful for debugging). |
| family | Currently allows gaussian or binomial to describe the error distribution. Link function information will be ignored and should be contained in the method argument below. |
| method | A list (or a function to create a list) containing details on estimating the coefficients for the super learner and the model to combine the individual algorithms in the library. See?method.template for details. Currently, the built in options are either "method.NNLS" (the default), "method.NNLS2", "method.NNloglik", "method.CC\_LS", "method.CC\_nloglik", or "method.AUC". NNLS and NNLS2 are non-negative least squares based on the Lawson-Hanson algorithm and the dual method of Goldfarb and Idnani, respectively. NNLS and NNLS2 will work for both gaussian and binomial outcomes. NNloglik is a non-negative binomial likelihood maximization using the BFGS quasi-Newton optimization method. NN\* methods are normalized so weights sum to one. CC\_LS uses Goldfarb and Idnani's quadratic programming algorithm to calculate the best convex combination of weights to minimize the squared error loss. CC\_nloglik calculates the convex combination of weights that minimize the negative binomial log likelihood on the logistic scale using the sequential quadratic programming algorithm. AUC, which only works for binary outcomes, uses the Nelder-Mead method via the optim function to minimize rank loss (equivalent to maximizing AUC). |
| id | Optional cluster identification variable. For the cross-validation splits, id forces observations in the same cluster to be in the same validation fold. id is passed to the prediction and screening algorithms in SL.library, but be sure to check the individual wrappers as many of them ignore the information. |
| obsWeights | Optional observation weights variable. As with id above, obsWeights is passed to the prediction and screening algorithms, but many of the built in wrappers ignore (or can't use) the information. If you are using observation weights, make sure the library you specify uses the information. |
| control | A list of parameters to control the estimation process. Parameters include saveFitLibrary  and  trimLogit. See [SuperLearner.control](http://127.0.0.1:20938/help/library/SuperLearner/help/SuperLearner.control) for details. |
| cvControl | A list of parameters to control the cross-validation process. Parameters include V,stratifyCV, shuffle and validRows. See [SuperLearner.CV.control](http://127.0.0.1:20938/help/library/SuperLearner/help/SuperLearner.CV.control) for details. |

**Details**

SuperLearner fits the super learner prediction algorithm. The weights for each algorithm in SL.library are estimated, along with the fit of each algorithm.

The prescreen algorithms. These algorithms first rank the variables in X based on either a univariate regression p-value of the randomForest variable importance. A subset of the variables in X is selected based on a pre-defined cut-off. With this subset of the X variables, the algorithms in SL.library are then fit.

The SuperLearner package contains a few prediction and screening algorithm wrappers. The full list of wrappers can be viewed with listWrappers(). The design of the SuperLearner package is such that the user can easily add their own wrappers. We also maintain a website with additional examples of wrapper functions at <https://github.com/ecpolley/SuperLearnerExtra>.

**Value**

|  |  |
| --- | --- |
| call | The matched call. |
| libraryNames | A character vector with the names of the algorithms in the library. The format is 'predictionAlgorithm\_screeningAlgorithm' with '\_All' used to denote the prediction algorithm run on all variables in X. |
| SL.library | Returns SL.library in the same format as the argument with the same name above. |
| SL.predict | The predicted values from the super learner for the rows in newX. |
| coef | Coefficients for the super learner. |
| library.predict | A matrix with the predicted values from each algorithm in SL.library for the rows in newX. |
| Z | The Z matrix (the cross-validated predicted values for each algorithm inSL.library). |
| cvRisk | A numeric vector with the V-fold cross-validated risk estimate for each algorithm in SL.library. Note that this does not contain the CV risk estimate for the SuperLearner, only the individual algorithms in the library. |
| family | Returns the family value from above |
| fitLibrary | A list with the fitted objects for each algorithm in SL.library on the full training data set. |
| varNames | A character vector with the names of the variables in X. |
| validRows | A list containing the row numbers for the V-fold cross-validation step. |
| method | A list with the method functions. |
| whichScreen | A logical matrix indicating which variables passed each screening algorithm. |
| control | The control list. |
| cvControl | The cvControl list. |
| errorsInCVLibrary | A logical vector indicating if any algorithms experienced an error within the CV step. |
| errorsInLibrary | A logical vector indicating if any algorithms experienced an error on the full data. |

**References**

van der Laan, M. J., Polley, E. C. and Hubbard, A. E. (2008) Super Learner, *Statistical Applications of Genetics and Molecular Biology*, **6**, article 25. <http://www.bepress.com/sagmb/vol6/iss1/art25>

**Function**

**CV.SuperLearner:** Function to get V-fold cross-validated risk estimate for super learner

**Description**

Function to get V-fold cross-validated risk estimate for super learner. This function simply splits the data into V folds and then calls SuperLearner. Most of the arguments are passed directly to SuperLearner.

**Usage**

CV.SuperLearner(Y, X, V = 20, family = gaussian(), SL.library,

method = "method.NNLS", id = NULL, verbose = FALSE,

control = list(saveFitLibrary = FALSE), cvControl = list(),

obsWeights = NULL, saveAll = TRUE, parallel = "seq")

**Arguments**

|  |  |
| --- | --- |
| Y | The outcome. |
| X | The covariates. |
| V | The number of folds for CV.SuperLearner. This is not the number of folds forSuperLearner. The number of folds for SuperLearner is controlled withcvControl. |
| family | Currently allows gaussian or binomial to describe the error distribution. Link function information will be ignored and should be contained in the method argument below. |
| SL.library | Either a character vector of prediction algorithms or a list containing character vectors. See details below for examples on the structure. A list of functions included in the SuperLearner package can be found with listWrappers(). |
| method | A list (or a function to create a list) containing details on estimating the coefficients for the super learner and the model to combine the individual algorithms in the library. See?method.template for details. Currently, the built in options are either "method.NNLS" (the default), "method.NNLS2", "method.NNloglik", "method.CC\_LS", "method.CC\_nloglik", or "method.AUC". NNLS and NNLS2 are non-negative least squares based on the Lawson-Hanson algorithm and the dual method of Goldfarb and Idnani, respectively. NNLS and NNLS2 will work for both gaussian and binomial outcomes. NNloglik is a non-negative binomial likelihood maximization using the BFGS quasi-Newton optimization method. NN\* methods are normalized so weights sum to one. CC\_LS uses Goldfarb and Idnani's quadratic programming algorithm to calculate the best convex combination of weights to minimize the squared error loss. CC\_nloglik calculates the convex combination of weights that minimize the negative binomial log likelihood on the logistic scale using the sequential quadratic programming algorithm. AUC, which only works for binary outcomes, uses the Nelder-Mead method via the optim function to minimize rank loss (equivalent to maximizing AUC). |
| id | Optional cluster identification variable. For the cross-validation splits, id forces observations in the same cluster to be in the same validation fold. id is passed to the prediction and screening algorithms in SL.library, but be sure to check the individual wrappers as many of them ignore the information. |
| verbose | Logical; TRUE for printing progress during the computation (helpful for debugging). |
| control | A list of parameters to control the estimation process. Parameters includesaveFitLibrary and trimLogit. See [SuperLearner.control](http://127.0.0.1:20938/help/library/SuperLearner/help/SuperLearner.control) for details. |
| cvControl | A list of parameters to control the cross-validation process. Parameters include V,stratifyCV, shuffle and validRows. See [SuperLearner.CV.control](http://127.0.0.1:20938/help/library/SuperLearner/help/SuperLearner.CV.control) for details. |
| obsWeights | Optional observation weights variable. As with id above, obsWeights is passed to the prediction and screening algorithms, but many of the built in wrappers ignore (or can't use) the information. If you are using observation weights, make sure the library you specify uses the information. |
| saveAll | Logical; Should the entire SuperLearner object be saved for each fold? |
| parallel | Options for parallel computation of the V-fold step. Use "seq" (the default) for sequential computation. parallel = 'multicore' to use mclapply for the V-fold step (but note that SuperLearner() will still be sequential). The default for mclapply is to check the mc.cores option, and if not set to default to 2 cores. Be sure to setoptions()$mc.cores to the desired number of cores if you don't want the default. Orparallel can be the name of a snow cluster and will use parLapply for the V-fold step. For both multicore and snow, the inner SuperLearner calls will be sequential. |

**Details**

The SuperLearner function builds a estimator, but does not contain an estimate on the performance of the estimator. Various methods exist for estimator performance evaluation. If you are familiar with the super learner algorithm, it should be no surprise we recommend using cross-validation to evaluate the honest performance of the super learner estimator. The function CV.SuperLearner computes the usual V-fold cross-validated risk estimate for the super learner (and all algorithms in SL.library for comparison).

**Value**

An object of class CV.SuperLearner (a list) with components:

|  |  |
| --- | --- |
| call | The matched call. |
| AllSL | If saveAll = TRUE, a list with output from each call to SuperLearner, otherwise NULL. |
| SL.predict | The predicted values from the super learner when each particular row was part of the validation fold. |
| discreteSL.predict | The traditional cross-validated selector. Picks the algorithm with the smallest cross-validated risk (in super learner terms, gives that algorithm coefficient 1 and all others 0). |
| whichDiscreteSL | A list of length V. The elements in the list are the algorithm that had the smallest cross-validated risk estimate for that fold. |
| library.predict | A matrix with the predicted values from each algorithm in SL.library. The columns are the algorithms in SL.library and the rows represent the predicted values when that particular row was in the validation fold (i.e. not used to fit that estimator). |
| coef | A matrix with the coefficients for the super learner on each fold. The columns are the algorithms in SL.library the rows are the folds. |
| folds | A list containing the row numbers for each validation fold. |
| V | Number of folds for CV.SuperLearner. |
| libraryNames | A character vector with the names of the algorithms in the library. The format is 'predictionAlgorithm\_screeningAlgorithm' with '\_All' used to denote the prediction algorithm run on all variables in X. |
| SL.library | Returns SL.library in the same format as the argument with the same name above. |
| method | A list with the method functions. |
| Y | The outcom |