Package ‘ensemble’

September 19, 2014

Type Package

Title What the package does (short line)

Version 1.0

Date 2014-09-18

Author Who wrote it

Maintainer Who to complain to `<yourfault@somewhere.net>`

Description More about what it does (maybe more than one line)

License What license is it under?

R topics documented:

`ensemble-package` *Fit ensemble models and optimize tuning parameters*

Description

This package provides functions for building models via cross-validation and combining those models via ensembling to generate a good prediction on test data. Additionally, the package provides a function for optimizing tuning parameters.

Details

Package:	ensemble
Type:	Package
Version:	1.0
Date:	2014-09-18

Author(s)

Maintainer: Josh Browning <rockclimber112358@gmail.com>

cvModel *Build an Cross-Validated Model*

Description

This function allows the user to specify a particular model (lm, randomForest, neuralnet, etc.) and easily build cross-validated models on the data. Additionally, predictions from all 10 models are averaged on a test dataset for prediction or as an input to an ensemble model.

Usage

```
cvModel(modelFunc, cvGroup, predFunc = predict, d = NULL, form = NULL, x = NULL, y = NULL, args = list(),
```

Arguments

`modelFunc`	Function which specifies what model will be fit to the data. Can be a built-in function or a custom, user-defined function; however, it must accept arguments d and form or x and y (depending on what is specified with this cvModel function) as well as any arguments passed via args).
`cvGroup`	A numeric vector of the same length as the number of rows of x and y or d. It specifies which observations should belong to which cross-validation groups. If data is supplied only for prediction purposes, than cvGroup should be -1 for those observations.
`predFunc`	A function which takes two arguments, fit and newdata, and returns a numeric vector or matrix of the same number of rows as newdata. If multiple predictions are desirable, this function should return a matrix with multiple columns and pred.cols should be set accordingly. Note that some functions default predict function will work here, but sometimes the user must write a wrapper to the predict function (usually adjusting argument names).
`d`	One of d and form OR x and y should be supplied. d is the data.frame which contains the data for modeling.
`form`	One of d and form OR x and y should be supplied. form is the formula which will be passed to modelFunc.

`x` One of d and form OR x and y should be supplied. x is the data.frame, matrix, or numeric vector of independent observations.

`y` One of d and form OR x and y should be supplied. y is the numeric vector of dependent observations to be modeled.

`args` A list of additional (named) arguments to be supplied to modelFunc.

`pred.cols` If predFunc returns a matrix with more than one column, then this should be set to the number of columns the user desires to save.

`saveMods` Binary value indicating whether or not models should be saved. The object returned from this function is a list, and one component is mods. If saveMods=T, a list of all the models fit will be returned in mods.

`seed` Random seed to set, if desirable.

Details

This function uses the cross-validation groups supplied by the user to partition the data into several groups. For each i in the cross-validation values, a model is built on all the data excluding i and -1 (the testing data). This model is then used to predict on the testing data as well as the data with cv group=i. Since multiple predictions will be made on the testing data, this function averages the predictions across all models.

Value

A list with the following components:

`ensemble` A matrix containing the predictions of the cross-validation models for all observations.

`models` If saveMods=T, then this element is a list of all the model objects fit to each of the cross-validation groups. With large datasets, this object can be extremely large.

`call` The function call

.

Author(s)

Josh Browning (rockclimber112358@gmail.com)

Examples

```
d = data.frame( x1=rnorm(1000), x2=rnorm(1000), y=rnorm(1000) )
out = cvModel( modelFunc=glm
    ,cvGroup=sample(1:10, size=1000, replace=TRUE)
    ,form="y ~ x1 + x2"
    ,d=d )
library(glmnet)
out = cvModel( modelFunc=glmnet
    ,cvGroup=sample(1:10, size=1000, replace=TRUE)
    ,x=d[,1:2]
    ,y=d$y
    ,predFunc=function(fit, newdata){predict(fit, newx=newdata, s=c(4,2,1,.5))}
    ,pred.cols=4)
```

`makeEnsem` *Combine multiple predictions into a final ensemble prediction.*

Description

This function accepts multiple models, fit on the training data and estimated on the testing data (typically via a function such as cvModel). It generates an ensemble prediction by combining these models in an intelligent way. Note that this function reads in all files in the Submissions directory of the form '_raw.csv', which should be the predictions from the ensemble models. These files should contain all the predictions to be ensembled together.

Usage

```
makeEnsem(actual, lossFunc = function(preds, actual) {
    mean((preds - actual)^2)
}, numIters = 100, topN = 10, prune = 0.7, baseLoss = 0.5)
```

Arguments

`actual` A numeric vector with the observed dependent variable values (for both the test and training data). This vector should have observations in the same order as all the prediction files.

`lossFunc` A loss function to measure the performance of the fit. Smaller values indicate better performance.

`numIters` What are the maximum number of iterations this algorithm should iterate through? At each iteration, an additional model may be averaged to the current predictions.

`topN` The initial predictions are formed using the topN best models (i.e. models with the lowest lossFunc value). The default value of 10 is a good choice, at least if you have many models to ensemble together.

`prune` How many models should be removed from consideration? The worst prune % of models are removed.

`baseLoss` What is the loss of a naive prediction (i.e. the mean for all observations)? This is used to throw out really bad models.

Details

This function generates an ensemble prediction by first taking the best topN models and combining them. Then, a new ensemble is considered by performing a weighted average of w times the current forecast and 1 times a new model (where w=topN). If any of these averages beat the current ensemble (in terms of loss evaluated on the cross-validated training sets), that ensemble is stored as the best. Either way, w is increased by 1. The algorithm proceeds until numIters is reached.

Value

`preds` The prediction of the ensemble model on all observations (cross-validated as well as testing observations).

`weights` Models selected at each step to build the ensemble.

Author(s)

Josh Browning (rockclimber112358@gmail.com)

See Also

cvModel

Examples

```
d = data.frame( x1=rnorm(1000), x2=rnorm(1000), y=rnorm(1000) )
#Assume last 100 rows are test data
cvGroup = c(sample(1:10,size=900,replace=TRUE), rep(-1,100))
d$y[901:1000] = NA
out = cvModel( modelFunc=glm
    ,cvGroup=cvGroup
    ,form="y ~ x1 + x2"
    ,d=d )
library(glmnet)
out2 = cvModel( modelFunc=glmnet
    ,cvGroup=cvGroup
    ,x=d[,1:2]
    ,y=d$y
    ,predFunc=function(fit, newdata){predict(fit, newx=newdata, s=c(4,2,1,.5))}
    ,pred.cols=4)
dir.create("Submissions")
write.csv(out$ensemble, file="Submissions/glm_raw.csv", row.names=FALSE)
write.csv(out2$ensemble, file="Submissions/glmnet_raw.csv", row.names=FALSE)
ensem = makeEnsem(actual=d$y, prune=.2)
```

`makeOutput` *Generate output in Kaggle format*

Description

This function provides an easy interface for taking predictions and formatting the output for Kaggle submissions. May need to be tweaked for different competitions.

Usage

```
makeOutput(preds, call)
```

Arguments

`preds` Predicted values from a model or ensemble.

`call` What was the call generating the model (for documentation purposes)?

Details

Creates a new file in the submissions directory and assigns it an id. Additionally, it adds an entry to the desc.csv file describing what model was fit.

Value

Nothing is returned, but files are created/updated. See details.

Author(s)

Josh Browning (rockclimber112358@gmail.com)

optParams *Optimize model parameters*

Description

This function is designed to optimize the tuning parameters to a particular data mining model by building many models. Note that it may be extremely slow, but should give good estimates for the optimal tuning parameters (by trying many combinations).

Usage

```
optParams(func, form = NULL, data = NULL, x = NULL, y = NULL, nTrain = c(100, 1000, 10000), nValid = nTrai
    mean((pred - actual)^2)
}, optArgs = list(), optVals = rep(5, length(optArgs)), optRed = rep(0.7, length(optArgs)), predFunc = p
```

Arguments

`func` The data mining function to be optimized.

`form` Either form and d OR x and y should be supplied. form supplies the formula to be used for fitting the model.

`data` Either form and d OR x and y should be supplied. d is a dataframe to be used for fitting the model.

`x` Either form and d OR x and y should be supplied. x is a matrix, dataframe, or numeric vector containing the independent variables for fitting.

`y` Either form and d OR x and y should be supplied. y is a numeric vector containing the dependent variable for fitting.

`nTrain` The number of observations to be randomly sampled at each iteration (to build training models).

nValid — The number of observations to be randomly sampled at each iteration (to measure error of the trained models).

replications — Specifies how many iterations should be performed at each optimization step. Typically 30 should be a good amount to ensure a good optimum is found, but decreasing this can help improve computation time.

optFunc — A function accepting two arguments: pred and actual. From these two numeric vectors, the optFunc should provide a performance measure to be minimized.

optArgs

optVals

optRed

predFunc

constArgs

coldStart

seed

---

randArgs *Helper function for optParams*

Description

Gnerates random initial parameters, typically for use in optParams.

Usage

```
randArgs(optArgs)
```

Arguments

optArgs — A list containing the arguments to be optimized. See optParams.

Value

A random set of parameters for use in optParams.

Note

Typically, this function will not be called directly by the user but rather via optParams.

Author(s)

Josh Browning (rockclimber112358@gmail.com)

Index