Package 'gbm2sas'

July 22, 2025

Type Package

Title Convert GBM Object Trees to SAS Code

Version 4.0

Date 2025-06-02

Author John R. Dixon [aut, cre]

Maintainer John R. Dixon <gbm2sas@gmail.com>

Description Writes SAS code to get predicted values from every tree of a gbm.object.

License GPL-3

Depends gbm, utils

NeedsCompilation no

Repository CRAN

Date/Publication 2025-06-02 13:00:10 UTC

Contents

gbm2sas-package 1
gbm2sas 2

Index **6**

`gbm2sas-package` *Convert GBM Object Trees to SAS Code*

Description

Writes SAS code to get predicted values from every tree of a gbm object.

Author(s)

John R. Dixon <gbm2sas@gmail.com>

Maintainer: John R Dixon <gbm2sas@gmail.com>

gbm2sas *Convert GBM Object Trees to SAS Code*

Description

Writes SAS code to get predicted values for every tree of a gbm object. The predicted values are as reported by the gbm package function pretty.gbm.tree.

Usage

```
gbm2sas(
gbmobject,
data=NULL,
sasfile=NULL,
ntrees=NULL,
mysasdata="mysasdata",
treeval="treeval",
prefix="do_"
)
```

Arguments

gbmobject	An object of type gbm.object.
data	Data used to fit gbm.object.
sasfile	Name of file to write the SAS code to.
ntrees	Optional number of trees to use in prediction. Default is the value n.trees from gbmobject.
mysasdata	Optional name of dataset operated on in the SAS code. Default is "mysasdata".
treeval	Optional name to use for the value from each gbm tree. These will have a number appended. Default variable names will be of the form "treeval1", "treeval2", etc. Set this to a different prefix to avoid overwriting SAS dataset values, if necessary.
prefix	Optional prefix to use for intermediate SAS program control variables. These will have a number appended. Default variable names will be of the form "do_1", "do_2", etc. Set this to a different prefix to avoid overwriting SAS dataset values, if necessary.

Value

sasfile	SAS code will be written to sasfile.

Author(s)

John R. Dixon <gbm2sas@gmail.com>

References

Package 'gbm'. Greg Ridgeway with contributions from others.

Examples

```
set.seed(18221)
# This example is taken from the gbm package documentation.
# create some data
N <- 1000
X1 <- runif(N)
X2 <- 2*runif(N)
X3 <- ordered(sample(letters[1:4],N,replace=TRUE),levels=letters[4:1])
X4 <- factor(sample(letters[1:6],N,replace=TRUE))
X5 <- factor(sample(letters[1:3],N,replace=TRUE))
X6 <- 3*runif(N)
mu <- c(-1,0,1,2)[as.numeric(X3)]

SNR <- 10 # signal-to-noise ratio
Y <- X1**1.5 + 2 * (X2**.5) + mu
sigma <- sqrt(var(Y)/SNR)
Y <- Y + rnorm(N,0,sigma)

# introduce some missing values
X1[sample(1:N,size=500)] <- NA
X4[sample(1:N,size=300)] <- NA

thedata <- data.frame(Y=Y,X1=X1,X2=X2,X3=X3,X4=X4,X5=X5,X6=X6)

# fit initial model
gbm1 <-
gbm(Y~X1+X2+X3+X4+X5+X6,
    data=thedata,
    var.monotone=c(0,0,0,0,0,0),
    distribution="gaussian",
    n.trees=1000,
    shrinkage=0.05,
    interaction.depth=3,
    bag.fraction = 0.5,
    train.fraction = 1,
    n.minobsinnode = 10,
    cv.folds = 3,
    keep.data=TRUE,
    verbose=FALSE,
    n.cores=1)

# We will pass a csv file to SAS, to demonstrate that the tree values from gbm2sas agree
# with R.  Note that if train.fraction were <1 above, we would need to pass the
# analogous subsample to sas via the csv file below. Since the fraction used was 1,
# we pass the entire dataset "thedata" below.
best.iter<-gbm.perf(gbm1,method="cv",plot.it=FALSE) # find a good number of trees
fit_from_r<-predict(gbm1, n.trees=best.iter) # get fitted values from R
avgresponse<-gbm1$initF # we will pass gbm1's intercept to SAS via the csv file
```

```
numtrees<-best.iter # we will pass the total number of trees to SAS via the csv file
fitdata<-cbind(thedata,avgresponse,numtrees,fit_from_r) # augment the training data
# Write the csv file.  We require SAS's missing() function and R agree on what values
# are missing.  Hence the "na" argument below, which assures SAS's proc import will
# assign missing values in agreement with what R considers a missing value.
write.table(fitdata,"checkdata.csv",
sep=",", quote=FALSE, row.names=FALSE, col.names=TRUE, na="")

# Now use gbm2sas
gbm2sas(
gbm1, # gbm object from above
data=thedata, # dataset used to fit model
sasfile="gbmforest.sas", # name to use for SAS code file
ntrees=best.iter, # number of trees
mysasdata="sasdataset", # name to use for dataset within SAS
treeval="treevalue", # name to use for value returned for each tree from pretty.gbm.tree
prefix="dobranch_" # variable name for controlling branching in SAS code
)

# SAS program to check R versus SAS fitted values
#proc import out=sasdataset
#     datafile= "checkdata.csv" /* file written by the R example code */
#     dbms=csv replace;
#     getnames=yes;
#run;
#/* Below we assume the SAS missing() function and R agree on what values are missing.
#The missing values were written by R to checkdata.csv such that proc import will
#correctly assign a missing value to them. */
#
#%include "gbmforest.sas"; /* SAS code written by gbm2sas */
#
#data sasdataset;
#set sasdataset;
#call symput('numtrees', numtrees); /* define macro variable holding ntrees */
#run;
#
#%macro checksascode(); /* macro to check SAS versus R */
#data sasdataset; set sasdataset;
#fit_from_sas=avgresponse; /* we need to start with the intercept from the gbm.object */
#%DO loop = 1 %TO &numtrees.;
#fit_from_sas=fit_from_sas+treevalue&loop.; /* add fitted value from each tree */
#%END;
#/* find the discrepancy between R and SAS fitted values for this observation */
#diff=abs(fit_from_sas-fit_from_r);
#run;
#%mend checksascode;
#
#%checksascode() /* call the checking macro */
#
#/* get worst discrepancy over all observations */
#proc sql; select max(diff) as max_discrepancy from sasdataset; quit;
#
# output from SAS
```

```
#max_discrepancy
#---------------
#       7.33E-15

file.remove("checkdata.csv")
file.remove("gbmforest.sas")
```

Index

∗ **gbm**
gbm2sas, 2
gbm2sas-package, 1
∗ **sas**
gbm2sas, 2
gbm2sas-package, 1

gbm2sas, 2
gbm2sas-package, 1