<div style="border:1px solid">

h2oml gbregress — Gradient boosting regression[+]

</div>

[+]This command includes features that are part of StataNow.

<table>
<tr><td>Description</td><td>Quick start</td><td>Menu</td><td>Syntax</td></tr>
<tr><td>Options</td><td>Remarks and examples</td><td>Stored results</td><td>Also see</td></tr>
</table>

Description

h2oml gbregress implements gradient boosting regression for continuous and count responses. You can choose from six loss functions, validate your model by using validation data or cross-validation, and tune hyperparameters and stop early to improve model performance on new data. This command provides only measures of performance. See [H2OML] **h2oml postestimation** for commands to compute and explain predictions, examine variable importance, and perform other postestimation analyses.

For an introduction to decision trees and the gradient boosting machine (GBM) method, see [H2OML] **Intro**.

Quick start

Before running the h2oml gbregress command, an H2O cluster must be initialized and data must be imported to an H2O frame; see [H2OML] **H2O setup** and *Prepare your data for H2O machine learning in Stata* in [H2OML] **h2oml**.

Perform gradient boosting regression of response y1 on predictors x1 through x100

```
h2oml gbregress y1 x1-x100
```

As above, but also report measures of fit for the validation frame named valid, and set an H2O random-number seed for reproducibility

```
h2oml gbregress y1 x1-x100, validframe(valid) h2orseed(123)
```

As above, but instead of a validation frame, use 3-fold cross-validation

```
h2oml gbregress y1 x1-x100, cv(3) h2orseed(123)
```

As above, but set the number of trees to 30, the maximum tree depth to 10, the learning rate to 0.01, and the predictor sampling rate to 0.6

```
h2oml gbregress y1 x1-x100, cv(3) h2orseed(123) ntrees(30)      ///
    maxdepth(10) lrate(0.01) predsamprate(0.6)
```

As above, but use the default exhaustive grid search to select the optimal number of trees and the maximum tree depth that minimize the mean squared error (MSE) metric

```
h2oml gbregress y1 x1-x100, cv(3) h2orseed(123) lrate(0.01)      ///
    predsamprate(0.6) ntrees(10(5)100) maxdepth(3(1)10)      ///
    tune(metric(mse))
```

As above, but use a random grid search, set an H2O random-number seed for this search, and limit the maximum search time to 200 seconds

```
h2oml gbregress y1 x1-x100, cv(3) h2orseed(123) lrate(0.01)      ///
    predsamprate(0.6) ntrees(10(5)100) maxdepth(3(1)10)      ///
    tune(metric(mse) grid(random, h2orseed(456)) maxtime(200))
```

As above, but specify a learning-rate decay of 0.9, and tune the number of bins for the categorical and continuous predictors

```
h2oml gbregress y1 x1-x100, cv(3) h2orseed(123) lrate(0.01)      ///
    lratedecay(0.9) predsamprate(0.6) ntrees(10(5)100)          ///
    maxdepth(3(1)10) binscont(15(5)50) binscat(500(50)1100)    ///
    tune(metric(mse) grid(random, h2orseed(456)) maxtime(200))
```

Run gradient boosting quantile regression by specifying the quantile loss function

```
h2oml gbregress y1 x1-x100, loss(quantile)
```

Menu

Statistics > H2O machine learning

Syntax

> h2oml gbregress *response_reg predictors* [, *options*]

response_reg and *predictors* correspond to column names of the current H2O frame.

options	Description
Model	
<u>loss</u>(*losstype*)	specify the loss function; default is loss(gaussian)
<u>valid</u>frame(*framename*)	specify the name of the H2O frame containing the validation dataset that will be used to evaluate the performance of the model
cv[(# [, *cvmethod*])]	specify the number of folds and method for cross-validation
cv(*colname*)	specify the name of the variable (H2O column) for cross-validation that identifies the fold to which each observation is assigned
<u>h2orseed</u>(#)	set H2O random-number seed for GBM
encode(*encode_type*)	specify H2O encoding type for categorical predictors; default is encode(enum)
stop[(# [, *stop_opts*])]	specify the number of training iterations and other criteria for stopping GBM training if the stopping metric does not improve
<u>maxt</u>ime(#)	specify the maximum run time in seconds for GBM; by default, no time restriction is imposed
<u>scoreevery</u>(#)	specify that metrics be scored after every # trees during training
<u>mono</u>tone(*predictors* [, *mon_opts*])	specify monotonicity constraints on the relationship between the response and the specified predictors
Hyperparameter	
<u>ntrees</u>(#|*numlist*)	specify the number of trees to build the GBM model; default is ntrees(50)
<u>lrate</u>(#|*numlist*)	specify the learning rate of each tree; default is lrate(0.1)
<u>lratedecay</u>(#|*numlist*)	specify the rate by which the learning rate specified in lrate() is decaying after adding each tree to the GBM; default is lratedecay(1)
<u>maxdepth</u>(#|*numlist*)	specify the maximum depth of each tree; default is maxdepth(5)
<u>minobsleaf</u>(#|*numlist*)	specify the minimum number of observations per child for splitting a leaf node; default is minobsleaf(10)
<u>predsamprate</u>(#|*numlist*)	specify the sampling rate for randomly selecting a fraction of predictors to build a tree; default is predsamprate(1)
<u>samprate</u>(#|*numlist*)	specify the sampling rate for randomly selecting a fraction of observations to build a tree; default is samprate(1)
<u>minsplitthreshold</u>(#|*numlist*)	specify the threshold for the minimum relative improvement needed for a node split; default is minsplitthreshold(1e-05)
<u>binscat</u>(#|*numlist*)	specify the number of bins to build the histogram for node splits for categorical predictors (enum columns in H2O); default is binscat(1024)

binsroot(#|*numlist*)	specify the number of bins to build the histogram for root node splits for continuous predictors (`real` and `int` columns in H2O); default is `binsroot(1024)`
binscont(#|*numlist*)	specify the number of bins to build the histogram for node splits for continuous predictors (`real` and `int` columns in H2O); default is `binscont(20)`

Tuning

tune(*tune_opts*)	specify hyperparameter tuning options for selecting the best-performing model

Only one of `validframe()` or `cv[()]` is allowed.

If neither `validframe()` nor `cv[()]` is specified, the evaluation metrics are reported for the training dataset.

`monotone()` can be specified only with `loss(gaussian)`, `loss(tweedie)`, or `loss(quantile)`.

When *numlist* is specified in one or more hyperparameter options, tuning is performed for those hyperparameters.

`collect` is allowed; see [U] 11.1.10 Prefix commands.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

losstype	Description
gaussian	Gaussian loss; the default
tweedie[, power(#)]	Tweedie loss; response must be nonnegative
poisson	Poisson loss; response must be nonnegative
laplace	Laplace loss
huber[, alpha(#)]	Huber loss
quantile[, alpha(#)]	quantile loss

cvmethod	Description
random	randomly split the training dataset into folds; the default
modulo	evenly split the training dataset into folds using the modulo operation
stratify	evenly distribute observations from the different classes of the response to all folds

stop_opts	Description
metric(*metric_option*)	specify the stopping metric for training or grid search
tolerance(#)	specify the tolerance value by which a model must improve before the training or grid search stops; default is `tolerance(1e-3)`

tune_opts	Description
metric(*metric_option*)	specify the metric for selecting the best-performing model
grid(*gridspec*)	specify whether to perform an exhaustive or random search for all hyperparameter combinations
maxmodels(#)	specify the maximum number of models considered in the grid search; default is all configurations
maxtime(#)	specify the maximum run time for the grid search in seconds; default is no time limit
stop[(# [, *stop_opts*])]	specify the number of iterations and other criteria for stopping GBM training if the stopping metric does not improve in the grid search
parallel(#)	specify the number of models to build in parallel during the grid search; default is parallel(1), sequential model building
nooutput	suppress the table summarizing hyperparameter tuning

If any of maxmodels(), maxtime(), or stop[()] is specified, then grid(random) is implied.

Options

> Model

loss(), validframe(), cv[()], h2orseed(), encode(), stop[()], maxtime(), scoreevery(), and monotone(); see [H2OML] *h2oml gbm*.

> Hyperparameter

ntrees(), lrate(), lratedecay(), maxdepth(), minobsleaf(), predsamprate(), samprate(), minsplitthreshold(), binscat(), binsroot(), and binscont(); see [H2OML] *h2oml gbm*.

> Tuning

tune(); see [H2OML] *h2oml gbm*.
stata.com

Remarks and examples

For examples, see *Remarks and examples* in [H2OML] *h2oml gbm*.

Stored results

h2oml gbregress stores the following in e():

Scalars
    e(N_train)   number of observations in the training frame
    e(N_valid)   number of observations in the validation frame (with option validframe())
    e(N_cv)   number of observations in the cross-validation (with option cv())
    e(n_cvfolds)   number of cross-validation folds (with option cv())
    e(k_predictors)   number of predictors
    e(n_trees)   number of trees
    e(n_trees_a)   actual number of trees used in GBM
    e(maxdepth)   maximum specified tree depth

e(depth_min_a)	achieved minimum tree depth
e(depth_avg_a)	achieved average depth among trees
e(depth_max_a)	achieved maximum tree depth
e(minobsleaf)	minimum specified number of observations for a child leaf
e(lrate)	learning rate
e(lratedecay)	learning rate decay
e(samprate)	observation sampling rate
e(predsamprate)	predictor sampling rate
e(minsplitthr)	minimum split improvement threshold
e(binscat)	number of bins for categorical predictors
e(binsroot)	number of bins for root node
e(binscont)	number of bins for continuous predictors
e(h2orseed)	H2O random-number seed
e(alpha)	top percentile of residuals if loss(huber); quantile if loss(quantile)
e(power)	variance power if loss(tweedie)
e(maxtime)	maximum run time
e(stop_iter)	maximum iterations before stopping training without metric improvement
e(stop_tol)	tolerance for metric improvement before training stops
e(scoreevery)	number of trees before scoring metrics during training
e(tune_h2orseed)	random-number seed for tuning (with option tune())
e(tune_stop_iter)	maximum iterations before stopping tuning without metric improvement (with option tune())
e(tune_stop_tol)	tolerance for metric improvement before tuning stops (with option tune())
e(tune_maxtime)	maximum run time for tuning grid search (with option tune())
e(tune_maxmodels)	maximum number of models considered in tuning grid search (with option tune())

Macros

e(cmd)	h2oml gbregress
e(cmdline)	command as typed
e(subcmd)	gbregress
e(method)	gbm
e(method_type)	regression
e(method_full_name)	Gradient boosting regression
e(response)	name of response
e(predictors)	names of predictors
e(title)	title in estimation output
e(loss)	name of the loss function
e(train_frame)	name of the training frame
e(valid_frame)	name of the validation frame (with option validframe())
e(cv_method)	fold assignment method (with option cv())
e(cv_varname)	name of variable identifying cross-validation folds (with option cv())
e(encode_type)	encoding type for categorical predictors
e(monotone_inc)	names of predictors with monotone increasing constraints
e(monotone_dec)	names of predictors with monotone decreasing constraints
e(stop_metric)	stopping metric for training
e(tune_grid)	grid search method used for tuning (with option tune())
e(tune_metric)	name of the tuning metric (with option tune())
e(tune_stop_metric)	stopping metric for tuning (with option tune())
e(properties)	nob noV
e(estat_cmd)	program used to implement h2omlestat
e(predict)	program used to implement h2omlpredict
e(marginsnotok)	predictions disallowed by margins

Matrices

e(metrics)	training, validation, and cross-validation metrics
e(hyperparam_table)	minimum, maximum, and selected hyperparameter values

Also see