h2omlgraph pdp — Produce partial dependence plot+

+This command includes features that are part of StataNow.

Description | Quick start | Menu | Syntax
Options | Remarks and examples | References | Also see

Description

h2ograph pdp produces the partial dependence plot (PDP) after h2oml *gbm* and h2oml *rf*. For regression, the PDP graphs the average prediction versus the values of a predictor of interest. For classification, PDP graphs average predicted probabilities versus values of a predictor of interest. Thus, PDP graphically depicts the average or partial effect of predictors on the response.

Quick start

Plot the PDP for the predictor x1

```
h2omlgraph pdp x1
```

As above, but plot for x1, x2, and x3, and combine the plots

```
h2omlgraph pdp x1 x2 x3, combine
```

As above, but show the standard deviations of the average response, and do not show the histogram

```
h2omlgraph pdp x1 x2 x3, combine sd nohistogram
```

Create a contour plot of the joint PDP for x1 and x2

```
h2omlgraph pdp x1 x2, pair
```

Menu

Statistics > H2O machine learning

Syntax

`h2omlgraph pdp` *predictors* [, *options*]

options	Description
Main	
* `target(`*classes*`)`	specify the target class(es) of the response variable for multiclass classification
`obs(#)`	specify the observation number for computing partial dependence
`savedata(`*filename*[, `replace`]`)`	save plot data to *filename*
Plot options	
`pair`	create a contour plot of the joint marginal predictions
`pairopts(`*contour_options*`)`	affect rendition of PDP contour plot
`lineopts(`*line_options*`)`	affect rendition of PDP line
`line#opts(`*line_options*`)`	affect rendition of PDP line for target class #
`sd`	display standard deviation band with PDP
`sdopts(`*area_options*`)`	affect rendition of the standard deviation band
`combine`	combine multiple PDP graphs
`combineopts(`*comb_opts*`)`	affect rendition of the combined graphs
`nohistogram`	do not plot histogram of the predictor
`histopts(`*bar_opts*`)`	affect rendition of the histogram
Y axis, X axis, Titles, Legend, Overall	
`name(`*namespec*[, `replace`]`)`	specify names of graphs
`saving(`*filespec*[, `replace`]`)`	save graphs in files
twoway_options	any options other than `by()` documented in [G-3] ***twoway_options***
`train`	specify that the partial dependence be reported using training results
`valid`	specify that the partial dependence be reported using validation results
`test`	specify that the partial dependence be computed using testing frame
`test(`*framename*`)`	specify that the partial dependence be computed using data in testing frame *framename*
`frame(`*framename*`)`	specify that the partial dependence be computed using data in H2O frame *framename*
`framelabel(`*string*`)`	label frame as *string* in the output

*`target()` is required after multiclass classification.

`train`, `valid`, `test`, `test()`, `frame()`, and `framelabel()` do not appear in the dialog box.

Options

Main

`target(`*classes*`)` specifies for which class or classes of the response variable the partial dependence should be plotted. `target()` is required after multiclass classification with `h2oml gbmulticlass` or `h2oml rfmulticlass`. `target()` is not allowed with `pair`.

`obs(#)` specifies the observation number for which partial dependence will be computed. The specified value should be a positive integer. If `obs()` is specified, the individual conditional expectation for `obs(#)` is computed; see [H2OML] **h2omlgraph ice**. `obs()` is not allowed with `sd`.

`savedata(`*filename*[`, replace`]`)` saves the plot data to a Stata data file (`.dta` file). `replace` specifies that *filename* be overwritten if it exists.

Plot options

`pair` specifies to create the contour plot of the joint marginal predictions of predictors. This option is valid only if two or more predictors are specified. `pair` is not allowed with any of `sd`, `target()`, `lineopts()`, `histopts()`, or `line#opts()`.

`pairopts(`*contour_options*`)` affects the rendition of the contour plot. See [G-2] **graph twoway contour**.

`lineopts(`*line_options*`)` affects the rendition of the PDP line. See [G-3] ***line_options***. `lineopts()` is not allowed with `pair`.

`line#opts(`*line_options*`)` affects the rendition of the PDP line for the target class #. See [G-3] ***line_options***. `line#opts()` is valid only if `target()` is specified. `line#opts()` is not allowed with `pair`.

`sd` specifies to plot a standard deviation band. For each observed value of the specified predictor, PDP estimates the mean response, and the standard deviation is estimated using those responses. `sd` is not allowed with `pair` or `obs()`.

`sdopts(`*area_options*`)` affects the rendition of the standard deviation band. See [G-3] ***area_options***.

`combine` specifies to combine the graphs of PDP for individual predictors when more than one predictor is specified.

`combineopts(`*comb_opts*`)` affects the rendition of the combined graphs. See [G-2] **graph combine**.

`nohistogram` removes the histogram of the predictor from the PDP. By default, the histogram is included.

`histopts(`*bar_opts*`)` affects the rendition of the histogram; see [G-2] **graph twoway bar**. `histopts()` is not allowed with `pair`.

Y axis, X axis, Titles, Legend, Overall

`name(`*namespec*[`, replace`]`)` specifies the name of the graph or multiple graphs. See [G-3] ***name_option*** for a single graph. If multiple graphs are produced, then the argument of `name()` is either a list of names or a *stub*, in which case graphs are named *stub*`1`, *stub*`2`, and so on. With multiple graphs, if `name()` is not specified and neither `sleep()` nor `wait` is specified, then `name(Graph__#, replace)` is assumed.

`replace` specifies to replace existing graphs with the specified name or names.

`saving(`*filespec*[`, replace`]`)` specifies the filename or filenames to use to save the graph or multiple graphs to disk. See [G-3] ***saving_option*** for a single graph. If multiple graphs are produced, then the argument of `saving()` is either a list of filenames or a *stub*, in which case graphs are saved with filenames *stub*`1`, *stub*`2`, and so on.

`replace` specifies to replace existing graphs with the specified name or names.

twoway_options are any of the options documented in [G-3] ***twoway_options***, excluding `by()`. These include options for titling the graph (see [G-3] ***title_options***) and options for saving the graph to disk (see [G-3] ***saving_option***).

The following options are available with h2omlgraph pdp but are not shown in the dialog box:

train, valid, test, test(), and frame() specify the H2O frame for which partial dependencies are reported. Only one of train, valid, test, test(), or frame() is allowed.

train specifies that partial dependencies be reported using training results. This is the default when validation is not performed during estimation and when a postestimation frame has not been set with h2omlpostestframe.

valid specifies that partial dependencies be reported using validation results. This is the default when validation is performed during estimation and when a postestimation frame has not been set with h2omlpostestframe. valid may be specified only when the validframe() option is specified with h2oml *gbm* or h2oml *rf*.

test specifies that partial dependencies be computed on the testing frame specified with h2omlpostestframe. This is the default when a testing frame is specified with h2omlpostestframe. test may be specified only after a testing frame is set by using h2omlpostestframe. test is necessary only when a subsequent h2omlpostestframe command is used to set a default postestimation frame other than the testing frame.

test(*framename*) specifies that partial dependencies be computed using data in testing frame *framename* and is rarely used. This option is most useful when running a single postestimation command on the named frame. If multiple postestimation commands are to be run on the same test frame, it is more computationally efficient and convenient to specify the testing frame by using h2omlpostestframe instead of specifying test(*framename*) with individual postestimation commands.

frame(*framename*) specifies that partial dependencies be computed using the data in H2O frame *framename*.

framelabel(*string*) specifies the label to be used for the frame in the output.
stata.com

Remarks and examples

We assume you have read the introduction to explainable machine learning in [H2OML] **Intro**.

Remarks are presented under the following headings:

Introduction
Examples of using PDP

Introduction

The partial dependence plot (PDP) is an intuitive tool to study the marginal effect of predictors on the response (Friedman 2001). The PDP allows you to easily visualize how the expected response changes across different values of a predictor. For regression, the PDP graphs the average prediction versus the values of a predictor of interest. For classification, the PDP graphs the average of the predicted probabilities versus the values of a predictor of interest.

In fact, to study the average predictions (or predictive margins) for a single predictor in regression or binary classification, the PDP is analogous to the plot of predictive margins we can obtain from marginsplot in Stata after fitting a model with regress or logit, respectively.

Formally, let $f(\mathbf{X}_S, \mathbf{X}_C)$ be our machine learning model, $\mathbf{X}_S$ be the predictors whose effect we wish to study, and $\mathbf{X}_C$ be all other predictors in our model. For $\mathbf{X}_S$ fixed at $\mathbf{x}_S$, the partial dependence is defined as

$$f_S(\mathbf{x}_S) = E_{\mathbf{X}_C}\{f(\mathbf{x}_S, \mathbf{X}_C)\} = \int f(\mathbf{x}_S, \mathbf{x}_C)dP(\mathbf{x}_C)$$

In words, partial dependence is an average (over the marginal distribution of $\mathbf{X}_C$) of the predictions our model makes when we fix $\mathbf{X}_S$ at some value $\mathbf{x}_S$. In the `h2omlgraph pdp` syntax, $\mathbf{X}_S$ corresponds to the input *predictors*. In a finite sample, for the jth observation, partial dependence is computed by averaging predictions computed at the observed values of predictors $\mathbf{x}_{C_i}$ for $i = 1, \ldots, n$.

$$\widehat{f}_S(\mathbf{x}_{Sj}) = \frac{1}{n}\sum_{i=1}^{n}\widehat{f}(\mathbf{x}_{Sj}, \mathbf{x}_{C_i})$$

The PDP is a plot of such average predictions over the support of $\mathbf{X}_S$, which allows us to investigate how average predicted values of the response (in regression) or average predicted probabilities (in classification) vary over the support of the predictors of interest.

In practice, PDP works well when the dependence between $\mathbf{X}_S$ and $\mathbf{X}_C$ is not strong. When the dependence is strong or the true model includes interactions, PDP is not reliable and the individual conditional expectation curve is recommended for postestimation analysis of partial effects.

Examples of using PDP

In this section, we demonstrate some uses of the `h2omlgraph pdp` command. The examples are presented under the following headings.

Example 1: PDP interpretation for regression
Example 2: Caution on PDP causal interpretation
Example 3: PDP with a monotonicity constraint
Example 4: Joint marginal predictions through PDP
Example 5: PDP interpretation for multiclass classification

▷ Example 1: PDP interpretation for regression

In this example, we plot and interpret the PDP for a random forest regression model.

We start by opening the 1978 automobile data (`auto.dta`) in Stata and then putting the data into an H2O frame. Recall that `h2o init` initiates an H2O cluster, `_h2oframe put` loads the current Stata dataset into an H2O frame, and `_h2oframe change` makes the specified frame the current H2O frame. For details, see *Prepare your data for H2O machine learning in Stata* in [H2OML] **h2oml** and see [H2OML] **H2O setup**.

```
. use https://www.stata-press.com/data/r18/auto
(1978 automobile data)

. h2o init
  (output omitted)

. _h2oframe put, into(auto)
Progress (%): 0 100

. _h2oframe change auto
```

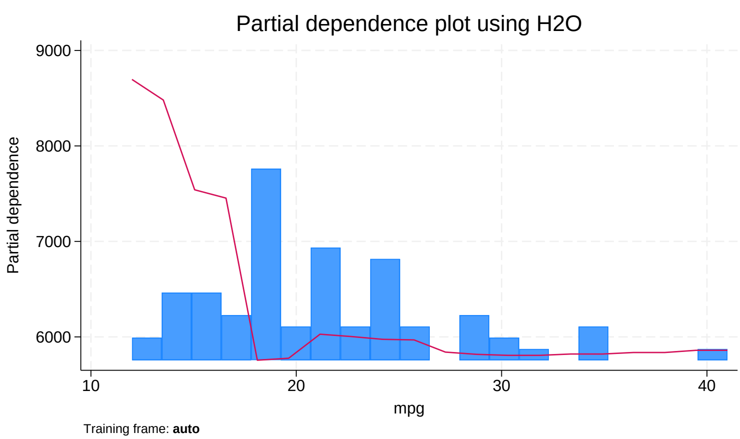
For simplicity, we save the predictor names in the global macro `predictors` in Stata. We then perform random forest regression with 100 trees and a maximum depth of 5.

```
. global predictors mpg trunk weight length
. h2oml rfregress price $predictors, h2orseed(19) ntrees(100) maxdepth(5)
Progress (%): 0 92.0 100
Random forest regression using H2O
Response: price
Frame:                                        Number of observations:
  Training: auto                                          Training =     74
Model parameters
Number of trees       = 100
              actual = 100
Tree depth:                                   Pred. sampling value =     -1
           Input max =   5                    Sampling rate        =   .632
                 min =   5                    No. of bins cat.     =  1,024
                 avg = 5.0                    No. of bins root     =  1,024
                 max =   5                    No. of bins cont.    =     20
Min. obs. leaf split =   1                    Min. split thresh.   = .00001
Metric summary
```

Metric	Training
Deviance	3760463
MSE	3760463
RMSE	1939.191
RMSLE	.2626369
MAE	1361.947
R-squared	.5618179

Finally, we use the `h2omlgraph pdp` command to show how the average predicted `price` changes across levels of the predictor `mpg`.

```
. h2omlgraph pdp mpg
```



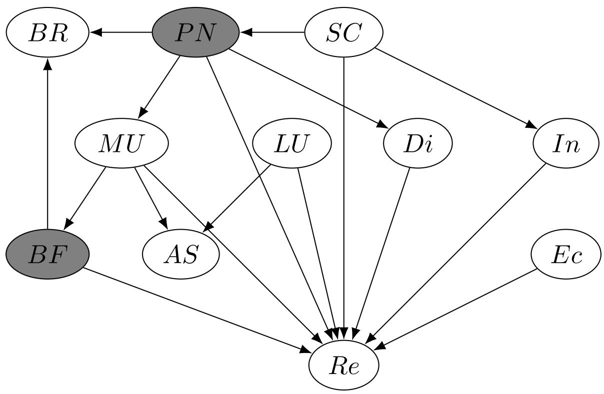
From the plot, we can see that the predicted price tends to decrease as the value of mpg increases. We also see a histogram of mpg, showing that only a few observations have mpg values over 30.

◁

▷ Example 2: Caution on PDP causal interpretation

In this example, we explore why it is important to exercise caution when using and interpreting machine learning explanation methods such as PDPs. See also example 2 of [H2OML] **h2omlgraph varimp** and examples in Krishna et al. (2022), Lakkaraju and Bastani (2020), and Slack et al. (2020).

The data-generating process and the discussion closely follow Lundberg (2021). Our goal is to understand how various predictors affect a subscriber's decision to renew their contract with a company, which is a causal question. We assume that our data are generated from the following causal directed acyclic graph (DAG).



See [CAUSAL] **Intro** for an introduction to DAGs. Here the abbreviations in the nodes correspond to the following predictors: MU is customer monthly usage, BF is the number of bugs faced, PN is product need, SC is the number of sales calls, Di is the customer discount, Ec is other macroeconomic activities, AS is the ad spending amount, LU is the last upgrade, Re is whether the customer renewed the contract, In is the number of interactions with a customer, and BR is bugs reported by a customer. The response is Re, whether the customer renewed the contract. The gray nodes represent unobserved confounders.

An important assumption to causally interpret PDP is that the model needs to satisfy the backdoor or unconfoundedness assumption (Zhao and Hastie 2021). In short, to identify the causal effect of one of these predictors on the response renewal, all other paths between the predictor and renewal must be blocked. Blocking the alternative paths involves "controlling for" or "conditioning on" a specific set of predictors. For definitions, see Pearl (2009) and Imbens and Rubin (2015).

We start by opening the `retention.dta` dataset in Stata and then putting it into an H2O frame.

```
. use https://www.stata-press.com/data/r18/retention
(Fictional retention data)

. h2o init
  (output omitted)

. _h2oframe put, into(retention)
Progress (%): 0 100

. _h2oframe change retention
```

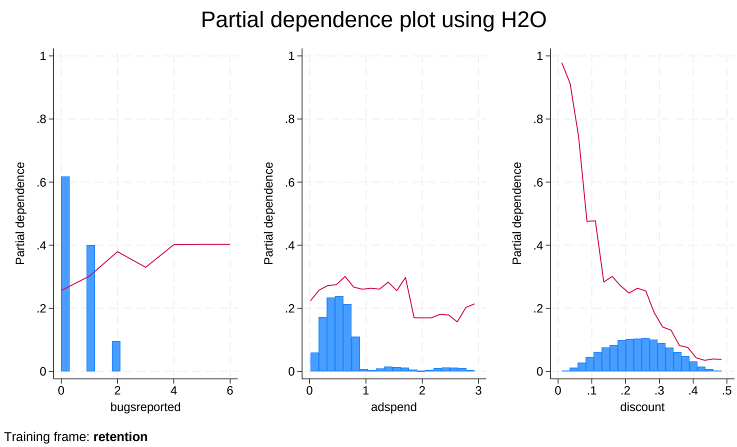
For convenience, we create a global macro `predictors` in Stata to store the names of the observed predictors. We then perform gradient boosting binary classification using these observed predictors.

```
. global predictors_obs salescalls interactions economy lastupgrade
> discount monthlyusage adspend bugsreported

. h2oml gbbinclass renew $predictors_obs, h2orseed(19) lrate(0.1)
> maxdepth(15) ntrees(300)

Progress (%): 0 0.6 5.3 13.6 23.0 41.6 63.6 86.0 95.9 98.3 100

Gradient boosting binary classification using H2O

Response: renew
Loss:     Bernoulli
Frame:                                  Number of observations:
  Training: retention                              Training = 10,000

Model parameters

Number of trees      =  300             Learning rate       =     .1
              actual =  300             Learning rate decay =      1
Tree depth:                             Pred. sampling rate =      1
           Input max =   15             Sampling rate       =      1
                 min =   15             No. of bins cat.    =  1,024
                 avg = 15.0             No. of bins root    =  1,024
                 max =   15             No. of bins cont.   =     20
Min. obs. leaf split =   10             Min. split thresh.  = .00001

Metric summary
```

Metric	Training
Log loss	.007453
Mean class error	0
AUC	1
AUCPR	1
Gini coefficient	1
MSE	.0000988
RMSE	.0099407

Next we use h2omlgraph pdp to plot the partial dependence for the predictors `bugsreported`, `adspend`, and `discount`. To combine the plots, we specify the `combine` option. We also specify the `combineopts()` option with the `cols(3)` suboption to request three columns, and we give the y axis a common scale by specifying the `ycommon` suboption.

```
. h2omlgraph pdp bugsreported adspend discount, combine
> combineopts(cols(3) ycommon)
```



The figure suggests counterintuitive results. Specifically, as the number of bugs reported increases, the probability of retention also increases, and as the discount increases, the probability of retention decreases.

A closer look at a causal DAG sheds more light on the source of these counterintuitive results. The `bugsreported` (BR) predictor is a collider (for definitions, see *Causal diagrams* in [CAUSAL] **Intro**), and by conditioning on a collider, we open a path between its parents, BF and PN, which are unobserved. This leads to an incorrect positive effect for BR, when there is no true effect. Similarly, conditioning on the predictor `adspend` (AS), we introduce a collider bias. Finally, the effect of `discount` (Di) suffers from the unobserved confounders. In causal DAG language, because PN and BF are unobserved, there are open backdoor paths between Di and Re.

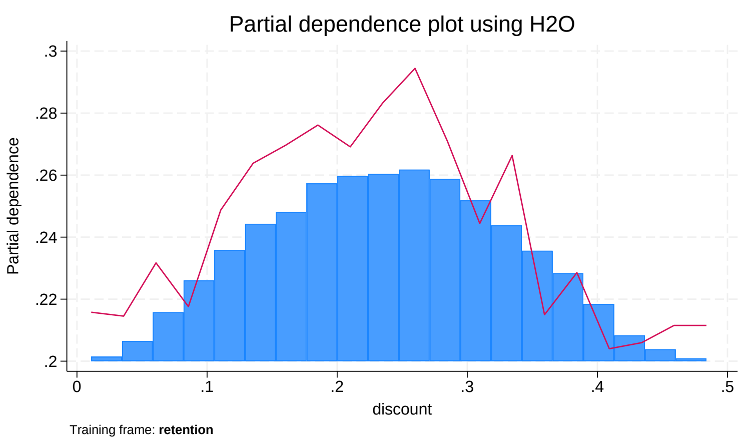
These results highlight the fundamental difference between prediction and causal inference. The same predictors can be good for predicting an outcome but may not be useful for causal inference. For details and more discussion, see Cinelli, Forney, and Pearl (2024).

Because the dataset is artificial, we can demonstrate the effect of controlling unobserved confounders on the average predicted probabilities. We now control for the number of bugs faced and product needed, and we omit BR and AS from our model. The new set of predictors is saved in the global macro `predictors` in Stata.

```
. global predictors salescalls interactions economy lastupgrade
> discount monthlyusage bugsfaced productneed

. h2oml gbbinclass renew $predictors, h2orseed(19) lrate(0.1)
> maxdepth(15) ntrees(300)

Progress (%): 0 6.3 15.6 25.3 35.6 56.9 77.6 97.6 100

Gradient boosting binary classification using H2O

Response: renew
Loss:     Bernoulli
Frame:                                    Number of observations:
  Training: retention                                Training = 10,000

Model parameters

Number of trees       =  300             Learning rate       =     .1
             actual =  300             Learning rate decay =      1
Tree depth:                              Pred. sampling rate =      1
           Input max =   15              Sampling rate       =      1
                 min =   15              No. of bins cat.    =  1,024
                 avg = 15.0              No. of bins root    =  1,024
                 max =   15              No. of bins cont.   =     20
Min. obs. leaf split =   10              Min. split thresh.  = .00001

Metric summary
```

Metric	Training
Log loss	.0022039
Mean class error	0
AUC	1
AUCPR	1
Gini coefficient	1
MSE	9.28e-06
RMSE	.0030459

```
. h2omlgraph pdp discount

Progress (%): 0 100
```



We can see that the interpretation of Di changed substantially. The partial dependence first grows with the discount, but then clearly decreases for discounts greater than 0.25.

◁

▷ Example 3: PDP with a monotonicity constraint

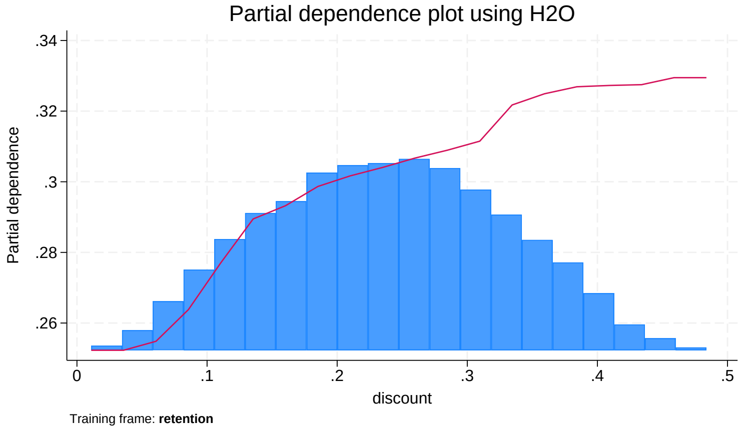
In some applications, it is reasonable to assume that the response is a monotone function of the predictor. For details, see [H2OML] **Intro**. In this example, we continue with example 2 and show a PDP after enforcing monotonicity constraints. Suppose we strongly believe that the effect of the predictor `discount` should be monotonic increasing. This information can be directly imposed on the gradient boosting machine model by using the `monotone()` option.

```
. h2oml gbbinclass renew $predictors, h2orseed(19) lrate(0.1)
> maxdepth(15) ntrees(300) monotone(discount)
Progress (%): 0 4.3 13.3 22.6 31.3 49.0 68.3 86.3 100
Gradient boosting binary classification using H2O
Response: renew
Loss:     Bernoulli
Frame:                                     Number of observations:
  Training: retention                                 Training = 10,000
Model parameters
Number of trees       =  300               Learning rate       =     .1
              actual  =  300               Learning rate decay =      1
Tree depth:                                Pred. sampling rate =      1
           Input max =   15                Sampling rate       =      1
                 min =   15                No. of bins cat.    =  1,024
                 avg = 15.0                No. of bins root    =  1,024
                 max =   15                No. of bins cont.   =     20
Min. obs. leaf split =   10                Min. split thresh.  = .00001
Metric summary
```

Metric	Training
Log loss	.0050499
Mean class error	0
AUC	1
AUCPR	1
Gini coefficient	1
MSE	.0000516
RMSE	.0071842

```
Monotone increasing: discount
```

```
. h2omlgraph pdp discount
Progress (%): 0 100
```



Compared with the PDP in example 2, the partial dependence of the predictor `discount` is monotonically increasing as the size of the discount increases.

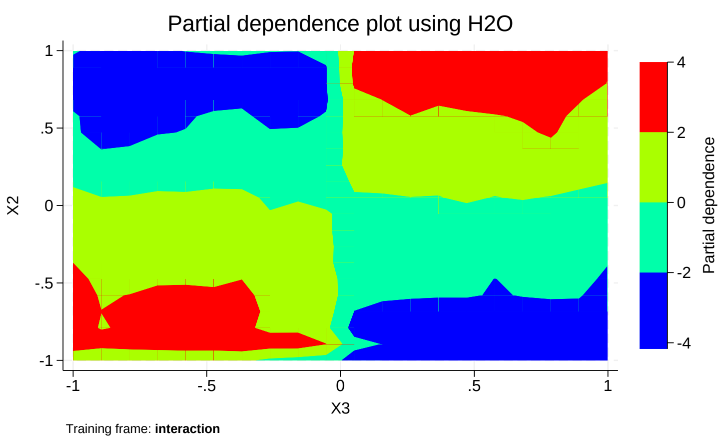
◁

▷ Example 4: Joint marginal predictions through PDP

In example 2 of [H2OML] **h2omlgraph ice**, we show that partial dependence curves are not useful for capturing an interaction effect and instead suggest to use ICE curves. In this example, we show how we might mitigate this issue by plotting the joint partial effect.

We start by restoring the `rf_inter` model by using the `h2omlest restore` command. The model was stored in example 1 of [H2OML] **h2omlgraph ice**.

```
. h2omlest restore rf_inter
(results rf_inter are active now)
. h2omlgraph pdp X2 X3, pair
```



We can see that the contour plot of the joint effect clearly captures the interaction, with the largest predictions in the regions $X_3 < 0$, $X_2 < -0.5$ and $X_3 > 0$, $X_2 > 0.5$.

◁

▷ Example 5: PDP interpretation for multiclass classification

In this example, we consider the well-known iris dataset, where the goal is to predict a class of iris plant. This dataset was used in Fisher (1936) and originally collected by Anderson (1935). We will demonstrate how to interpret the PDP for multiclass classification. For illustration purposes, we use random forest multiclass classification with 500 trees.

```
. use https://www.stata-press.com/data/r18/iris
(Iris data)
. h2o init
  (output omitted)
. _h2oframe put, into(iris)
Progress (%): 0 100
. _h2oframe change iris
. global predictors seplen sepwid petlen petwid
. h2oml rfmulticlass iris $predictors, h2orseed(19) ntrees(500)
Progress (%): 0 11.8 43.5 70.8 100
Random forest multiclass classification using H2O
Response: iris                           Number of classes     =      3
Frame:                                   Number of observations:
  Training: iris                                     Training =    150
Model parameters
Number of trees       = 500
              actual = 500
Tree depth:                              Pred. sampling value =     -1
           Input max =  20               Sampling rate        =   .632
                 min =   1               No. of bins cat.     =  1,024
                 avg = 3.7               No. of bins root     =  1,024
                 max =   9               No. of bins cont.    =     20
Min. obs. leaf split =   1               Min. split thresh.   = .00001
Metric summary
```

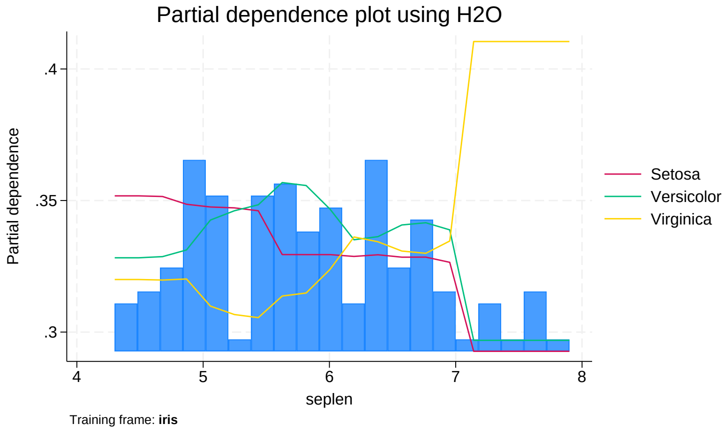
Metric	Training
Log loss	.118939
Mean class error	.0533333
MSE	.037385
RMSE	.1933519

To plot the partial dependence after multiclass classification, we need to specify the `target()` option in `h2omlgraph pdp`. In the `target()` option, we specify the names of the classes of the response `iris` for which we want to produce a PDP. We can list the classes of the response by typing

```
. _h2oframe levelsof iris
`"Setosa"' `"Versicolor"' `"Virginica"'
```

Next we plot the partial dependence of the predictor `seplen` on all three classes.

```
. h2omlgraph pdp seplen, target(Setosa Versicolor Virginica)
Progress (%): 0 100
```



On the plot, the red line corresponds to the PDP for the `Setosa` class. The plot shows how the average probability of predicting `Setosa` differs with the different values of the predictor `seplen`.

◁

References

Anderson, E. 1935. The irises of the Gaspé Peninsula. *Bulletin of the American Iris Society* 59: 2–5.

Cinelli, C., A. Forney, and J. Pearl. 2024. A crash course in good and bad controls. *Sociological Methods and Research* 53: 1071–1104. https://doi.org/10.1177/00491241221099552.

Fisher, R. A. 1936. The use of multiple measurements in taxonomic problems. *Annals of Eugenics* 7: 179–188. https://doi.org/10.1111/j.1469-1809.1936.tb02137.x.

Friedman, J. H. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* 29: 1189–1232. https://doi.org/10.1214/aos/1013203451.

Imbens, G. W., and D. B. Rubin. 2015. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. New York: Cambridge University Press. https://doi.org/10.1017/CBO9781139025751.

Krishna, S., T. Han, A. Gu, S. Wu, S. Jabbari, and H. Lakkaraju. 2022. The disagreement problem in explainable machine learning: A practitioner's perspective. arXiv:2202.01602 [cs.LG], https://doi.org/10.48550/arXiv.2202.01602.

Lakkaraju, H., and O. Bastani. 2020. ""How do I fool you?": Manipulating user trust via misleading black box explanations". In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 79–85. New York: Association for Computing Machinery. https://doi.org/10.1145/3375627.3375833.

Lundberg, S. M. 2021. Be careful when interpreting predictive models in search of causal insights. *Medium: Thoughts and Theory*. https://medium.com/towards-data-science/be-careful-when-interpreting-predictive-models-in-search-of-causal-insights-e68626e664b6.

Pearl, J. 2009. *Causality: Models, Reasoning, and Inference*. 2nd ed. Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9780511803161.

Slack, D., S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju. 2020. "Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods". In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 180–186. New York: Association for Computing Machinery. https://doi.org/10.1145/3375627.3375830.

Zhao, Q., and T. J. Hastie. 2021. Causal interpretations of black-box models. *Journal of Business and Economic Statistics* 39: 272–281. https://doi.org/10.1080/07350015.2019.1624293.

Also see

[H2OML] **h2oml** — Introduction to commands for Stata integration with H2O machine learning+

[H2OML] **h2omlgraph ice** — Produce individual conditional expectation plot+