hal9001

An R package for the practical application of the highly adaptive lasso (HAL)

Rachael V. Phillips

Center for Targeted Machine Learning and Causal Inference (CTML)
University of California, Berkeley

May 14, 2024

American Causal Inference Conference 2024 Short Course
HAL and Adaptive TMLE in Causal Inference

Basic `hal9001` Functionality

1. Load package and data:
   ```
   library(hal9001)
   data(mtcars)
   ```
2. Create numeric vector for dependent variable:
   ```
   Y <- mtcars[,"mpg"]
   ```
3. Create dataframe or matrix of predictors:
   ```
   X <- mtcars[,c("cyl", "disp", "hp", "wt")]
   ```
4. Fit HAL:
   ```
   hal_fit <- fit_hal(X=X, Y=Y, family="gaussian")
   ```

Supported `family` include "gaussian" for penalized linear model, "binomial" for penalized logistic regression, "poisson" for penalized Poisson regression, "cox" for a penalized proportional hazards model, and "mgaussian" for multivariate penalized linear model

Basic hal9001 Functionality

1. Load package and data:
```
library(hal9001)
data(mtcars)
```
2. Create numeric vector for dependent variable:
```
Y <- mtcars[,"mpg"]
```
3. Create dataframe or matrix of predictors:
```
X <- mtcars[,c("cyl", "disp", "hp", "wt")]
```
4. Fit HAL:
```
hal_fit <- fit_hal(X=X, Y=Y, family="gaussian")
```

Supported `family` include "gaussian" for penalized linear model, "binomial" for penalized logistic regression, "poisson" for penalized Poisson regression, "cox" for a penalized proportional hazards model, and "mgaussian" for multivariate penalized linear model

Basic hal9001 Functionality

1. Load package and data:
   ```
   library(hal9001)
   data(mtcars)
   ```
2. Create numeric vector for dependent variable:
   ```
   Y <- mtcars[,"mpg"]
   ```
3. Create dataframe or matrix of predictors:
   ```
   X <- mtcars[,c("cyl", "disp", "hp", "wt")]
   ```
4. Fit HAL:
   ```
   hal_fit <- fit_hal(X=X, Y=Y, family="gaussian")
   ```

Supported `family` include "gaussian" for penalized linear model, "binomial" for penalized logistic regression, "poisson" for penalized Poisson regression, "cox" for a penalized proportional hazards model, and "mgaussian" for multivariate penalized linear model

Basic hal9001 Functionality

1. Load package and data:
```
library(hal9001)
data(mtcars)
```
2. Create numeric vector for dependent variable:
```
Y <- mtcars[,"mpg"]
```
3. Create dataframe or matrix of predictors:
```
X <- mtcars[,c("cyl", "disp", "hp", "wt")]
```
4. Fit HAL:
```
hal_fit <- fit_hal(X=X, Y=Y, family="gaussian")
```

Supported `family` include "gaussian" for penalized linear model, "binomial" for penalized logistic regression, "poisson" for penalized Poisson regression, "cox" for a penalized proportional hazards model, and "mgaussian" for multivariate penalized linear model

Summary table of `hal9001` HAL fit


```
summary(hal_fit)$table
```

coef	term
35.4070	Intercept
-4.0801	I(disp >= 440)
-4.0118	I(disp >= 78.7)
-2.7170	I(wt >= 1.513)
-2.4454	I(wt >= 3.215)
-1.8184	I(disp >= 71.1)
-1.7208	I(hp >= 180)
-1.6830	I(disp >= 95.1)
-1.6039	I(hp >= 66)*I(wt >= 2.2)
-1.5623	I(wt >= 2.2)
1.3785	I(disp >= 351)
-1.2444	I(hp >= 175)
-1.1888	I(disp >= 301)
-0.9026	I(hp >= 123)
0.7336	I(hp >= 52)
-0.5810	I(disp >= 120.1)*I(hp >= 97)
-0.4395	I(disp >= 108)*I(hp >= 93)

Tuning HAL

HAL's computational cost is controlled by the number of basis functions, which can be as large as $n * 2^{d-1}$

Tuning HAL

HAL's computational cost is controlled by the number of basis functions, which can be as large as $n * 2^{d-1}$

Options for constraining functional form of target function:

- Enforce minimum proportion of 1's in basis functions with `reduce_basis`

Tuning HAL

HAL's computational cost is controlled by the number of basis functions, which can be as large as $n * 2^{d-1}$

Options for constraining functional form of target function:

- Enforce minimum proportion of 1's in basis functions with `reduce_basis`
- Enforce a maximal order of interaction with `max_degree`

Tuning HAL

HAL's computational cost is controlled by the number of basis functions, which can be as large as $n * 2^{d-1}$

Options for constraining functional form of target function:

- Enforce minimum proportion of 1's in basis functions with `reduce_basis`
- Enforce a maximal order of interaction with `max_degree`
- Specify particular additive model structure (e.g., $f(X_1, X_2, X_3) = f_1(X_1) + f_2(X_2, X_3)$) and/or enforce monotonicity for some of the basis functions with `formula`

Tuning HAL

HAL's computational cost is controlled by the number of basis functions, which can be as large as $n * 2^{d-1}$

Options for constraining functional form of target function:

- Enforce minimum proportion of 1's in basis functions with `reduce_basis`
- Enforce a maximal order of interaction with `max_degree`
- Specify particular additive model structure (e.g., $f(X_1, X_2, X_3) = f_1(X_1) + f_2(X_2, X_3)$) and/or enforce monotonicity for some of the basis functions with `formula`
- Enforce higher order splines and thereby more smoothness with `smoothness_orders`

Tuning HAL

HAL's computational cost is controlled by the number of basis functions, which can be as large as $n * 2^{d-1}$

Options for constraining functional form of target function:

- Enforce minimum proportion of 1's in basis functions with `reduce_basis`
- Enforce a maximal order of interaction with `max_degree`
- Specify particular additive model structure (e.g., $f(X_1, X_2, X_3) = f_1(X_1) + f_2(X_2, X_3)$) and/or enforce monotonicity for some of the basis functions with `formula`
- Enforce higher order splines and thereby more smoothness with `smoothness_orders`
- Discretize continuous covariates using fewer cut points with `num_knots`

Specifying HAL model formulas

Example: Observe $O = (W_1, W_2, A, Y) \sim P_0$

R code: `fit_hal(Y, X, family, formula, ...)`

Additive model `formula`:

`Y` $\sim$. or `Y` $\sim$ `h(W1) + h(W2) + h(A)`

Bi-additive model `formula`:

`Y` $\sim$.^2 or

`Y` $\sim$ `h(W1)+h(W2)+h(A)+h(W1,W2)+h(W1,A)+h(W2,A)`

Only interactions with A `formula`:

`Y` $\sim$ `h(.)` + `h(.,A)` or

`Y` $\sim$ `h(W1)+h(W2)+h(A)+h(W1,A)+h(W2,A)`

Monotone ↑ (i) ↓ (d) `formula` examples:

`Y` $\sim$ `i(.)` or `Y` $\sim$ `i(.)` + `i(.,.)` or `Y` $\sim$ `i(W1)+d(W2)+i(A)`

Possible HAL fits under various smoothness orders

Example: Observe $(W, A, Y) \sim P_0$

R code: `fit_hal(Y, X, family, smoothness_orders=...)`

Example of `smoothness_orders=0`:

Additive model:

$$Y = \mathbb{I}(W > 0.5) + \mathbb{I}(W > 0.3) + \mathbb{I}(A > 0)$$

Bi-additive model:

$$Y = \mathbb{I}(W > 0.5) + \mathbb{I}(A > 0) + \mathbb{I}(W > 0.5, A > 0)$$

Example of `smoothness_orders=1`:

Additive model:

$$Y = \mathbb{I}(W > 0.5)[W - 0.5] + \mathbb{I}(W > 0.3)[W - 0.3] + \mathbb{I}(A > 0)[A - 0]$$

Bi-additive model:

$$Y = \mathbb{I}(W > 0.5)[W - 0.5] + \mathbb{I}(A > 0)[A - 0] + \mathbb{I}(W > 0.5, A > 0)[W - 0.5][A - 0]$$

Fitting HAL under different discretization of continuous covariates

Reducing number of spline knot points. This can be done separately for 1-way, 2-way, 3-way basis functions.

Example: Observe $(W, A, Y) \sim P_0$, $n = 2000$, $d = 12$, and let's consider up to 3-way interactions among covariates

R code: `fit_hal(Y, X, family, max_degree=3, num_knots,`

Fitting HAL under different discretization of continuous covariates

Example: Observe $(W, A, Y) \sim P_0$, $n = 2000$, $d = 12$, and let's consider up to 3-way interactions among covariates. Let's examine the size of the regression matrix $(n \times p)$ in this example under different binning schemes:

[Binning] $p = 30,000$ when we consider 100, 25, and 5 knot points for 1-way, 2-way, and 3-way interactions, respectively:

```
fit_hal(Y=Y, X=X, num_knots = c(100, 25, 5), max_degree=3)
```

Fitting HAL under different discretization of continuous covariates

Example: Observe $(W, A, Y) \sim P_0$, $n = 2000$, $d = 12$, and let's consider up to 3-way interactions among covariates. Let's examine the size of the regression matrix $(n \times p)$ in this example under different binning schemes:

[Binning] $p = 30,000$ when we consider 100, 25, and 5 knot points for 1-way, 2-way, and 3-way interactions, respectively:

```
fit_hal(Y=Y, X=X, num_knots = c(100, 25, 5), max_degree=3)
```

[No Binning] $p = 600,000$ when we consider n knot points:

```
fit_hal(Y=Y, X=X, num_knots = length(Y), max_degree=3)
```

Fitting HAL under different discretization of continuous covariates

Example: Observe $(W, A, Y) \sim P_0$, $n = 2000$, $d = 12$, and let's consider up to 3-way interactions among covariates. Let's examine the size of the regression matrix $(n \times p)$ in this example under different binning schemes:

[Binning] $p = 30,000$ when we consider 100, 25, and 5 knot points for 1-way, 2-way, and 3-way interactions, respectively:

```
fit_hal(Y=Y, X=X, num_knots = c(100, 25, 5), max_degree=3)
```

[No Binning] $p = 600,000$ when we consider n knot points:

```
fit_hal(Y=Y, X=X, num_knots = length(Y), max_degree=3)
```

Note: More basis functions does not necessarily imply better performance, and the package documentation provides detailed discussion on recommendations.

Fitting HAL under different discretization of continuous covariates

Example: Observe $(W, A, Y) \sim P_0$, $n = 2000$, $d = 12$, and let's consider up to 3-way interactions among covariates. Let's examine the size of the regression matrix $(n \times p)$ in this example under different binning schemes:

[Binning] $p = 30,000$ when we consider 100, 25, and 5 knot points for 1-way, 2-way, and 3-way interactions, respectively:

```
fit_hal(Y=Y, X=X, num_knots = c(100, 25, 5), max_degree=3)
```

[No Binning] $p = 600,000$ when we consider n knot points:

```
fit_hal(Y=Y, X=X, num_knots = length(Y), max_degree=3)
```

Additional Control of HAL Fits


```
fit_hal(
  X,
  Y,
  formula = NULL,
  X_unpenalized = NULL,
  max_degree = ifelse(ncol(X) >= 20, 2, 3),
  smoothness_orders = 1,
  num_knots = num_knots_generator(max_degree = max_degree, smoothness_orders =
    smoothness_orders, base_num_knots_0 = 200, base_num_knots_1 = 50),
  reduce_basis = NULL,
  family = c("gaussian", "binomial", "poisson", "cox", "mgaussian"),
  lambda = NULL,
  id = NULL,
  weights = NULL,
  offset = NULL,
  fit_control = list(cv_select = TRUE, use_min = TRUE, lambda.min.ratio = 1e-04,
    prediction_bounds = "default"),
  basis_list = NULL,
  return_lasso = TRUE,
  return_x_basis = FALSE,
  yolo = FALSE
)
```

Super Learner (SL) incorporating HAL

- By varying HAL tuning parameters, one can include many HAL estimators as candidates in the SL library
- The SL will perform as well as the oracle choice among all these HAL estimators and thereby achieves at minimal rate of convergence $n^{-1/3}(\log n)^{d/2}$
- One can still include other machine learning algorithms and parametric models as candidates in the SL
- Standard SL R packages support HAL candidates in the library:
  - `SuperLearner` R package: `SL.hal9001`
  - `sl3` R package: `Lrnr_hal9001`.

Meta-learning with HAL

The Super Learner (SL) is defined by the the library of candidate estimators, cross-validation scheme, loss function, and meta-learning algorithm.

Procedure for the Meta-HAL Super Learner

1. Perform meta-learning with HAL under specified L_1-norm.
2. Define a discrete SL that includes as candidates the L_1-norm specific meta-HAL SLs, in order to optimally select the L_1-norm of the HAL meta-learner.

This implementation guarantees the final selected meta-HAL will perform as well as the optimally tuned meta-HAL SL.

Wang, Zhang, and van der Laan. "Super Ensemble Learning Using the Highly-Adaptive-Lasso." *arXiv preprint arXiv:2312.16953* (2023).

Inferential bottleneck in realistic models

In models small enough so MLE is well behaved, nonparametric bootstrap outperforms estimating the sampling distribution with a normal distribution, *if asymptotics hasn't kicked in*

Efficient estimation of a pathwise differentiable estimand in realistic models warrants ML for nuisance function estimation

- Nonparametric bootstrap represents a generally inconsistent method (e.g., CV selector behaves very differently under sampling from the empirical versus true data distribution)
- IC-based inference using a normal limit distribution can be off-centered or less spread out than the actual sampling distribution of the estimator **in finite samples**
- Model-based bootstrap will be asymptotically valid as long as the density estimator is consistent...

HAL-MLE provides a solution to this bottleneck

HAL-MLE robust behavior under minimal, realistic assumptions

- The HAL-MLE of the nuisance parameter is an actual MLE, minimizing empirical risk over infinite-dimensional parameter space (depending on the model), where it's assumed nuisance parameter's sectional variation norm is universally bounded.
- The HAL-MLE is still well behaved by being consistent at a rate that is in the worst case still faster than $n^{-1/4}$.
- Smooth enough function of the data (while not compactly differentiable at all) that it's equally well behaved under sampling from empirical distribution

Nonparametric bootstrap provides asymptotically consistent estimation of the HAL-TMLE sampling distribution for the estimand

---

Cai and van der Laan. "Nonparametric bootstrap inference for the targeted highly adaptive least absolute shrinkage and selection operator (LASSO) estimator." *The International Journal of Biostatistics* (2020).

Concluding Remarks

- HAL is the first general nonparametric MLE
- It converges at fast rate
- TMLE with HAL-based initial estimators is guaranteed asymptotically efficient
- It represents a class of HAL estimators via various restrictions of function space
- It has many applications, such as meta-HAL super-learner for multi-modal data, outcome-adaptive HAL

Additional Resources

- SL Chapter of TLverse Handbook for HAL SL Examples: https://tlverse.org/tlverse-handbook/sl3.html
- Mark's FDA Webinar "Highly Adaptive Lasso (HAL) in Causal Inference": https://www.youtube.com/watch?v=YnXVAtzF4ss
- My contact information:
  - Email: rachaelvphillips@berkeley.edu
  - LinkedIn: https://www.linkedin.com/in/rachaelvp/
- Comprehensive suite of video lectures on Targeted Learning will be available on YouTube soon, following the addition of captions. Follow on LinkedIn to receive notification and/or email us for early access.