Technical Report: Partial Dependence without Model Predictions through Stratification

Terence Parr
University of San Francisco
parrt@cs.usfca.edu

James D. Wilson
University of San Francisco
jdwilson4@usfca.edu

Abstract

1 Introduction

Partial dependence, the isolated effect of a specific variable or variables on the response variable, y, is important to researchers and practitioners in many disparate fields such as medicine, business, and the social sciences. For example, in medicine, researchers are interested in the relationship between an individual's demographics or clinical features and their susceptibility to illness. Business analysts at a car manufacturer might need to know how changes in their supply chain are affecting defect rates. Climate scientists are interested in how different atmospheric carbon levels affect temperature.

For an explanatory matrix, $\mathbf{X}$, with a single variable, x_1, a plot of the y against x_1 visualizes the marginal effect of feature x_1 on y exactly. Given two or more features, one can similarly plot the marginal effects of each feature separately, however, the analysis is complicated by the interactions of the variables. Variable interactions, codependencies between features, result in marginal plots that do not isolate the specific contribution of a feature of interest to the target. For example, a marginal plot of sex (male/female) against body weight would likely show that, on average, men are heavier than women. While true, men are also taller than women on average, which likely accounts for most of the difference in average weight. It is unlikely that two "identical" people, differing only in sex, would be appreciably different in weight.

Rather than looking directly at the data, there are several partial dependence techniques that interrogate fitted models provided by the user: Friedman's original partial dependence (which we will denote FPD) Friedman [2000], Individual Conditional Expectations (ICE) Goldstein et al. [2015], Accumulated Local Effects (ALE) Apley [2016], and most recently SHAP Lundberg and Lee [2017]. Model-based techniques dominate the partial dependence research literature because interpreting the output of a fitted model has several advantages. Models have a tendency to smooth over noise. Models act like analysis preprocessing steps, potentially reducing the computational burden on model-based partial dependence techniques; e.g., ALE is $O(n)$ for the n records of $\mathbf{X}$. Model-based techniques are typically model-agnostic, though for efficiency, some provide model-specific optimizations, as SHAP does. Partial dependence techniques that interrogate models also provide insight into the models themselves; i.e., how variables affect model behavior. It is also true that, in some cases, a predictive model is the primary goal so creating a suitable model is not an extra burden.

Model-based techniques do have some significant disadvantages, however. As we demonstrate in Section 4 using synthetic and real data sets, model-based techniques vary in their ability

Preprint. Under review.

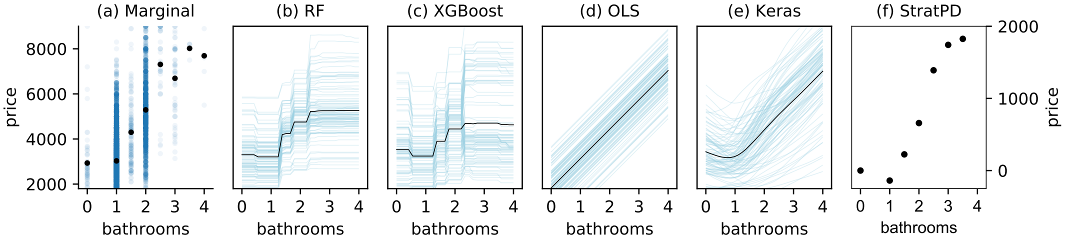

Figure 1: Plots of bathrooms versus rent price using New York City apartment rent data. (a) marginal plot, (b) PD/ICE plot derived from random forest, (c) PD/ICE plot derived from gradient boosted machine, and (d) PD/ICE plot derived from ordinary least squares regression; sample size is 10,000 observations of ~50k. The PD/ICE plots are different for the same data set, depending on the chosen user model. X['bathrooms'].unique() shows (array([0. , 1. , 1.5, 2. , 2.5, 3. , 3.5, 4.]), array([54, 8151, 140, 1539, 39, 67, 3, 7])). StratPD has missing last value, not enough data. what are R^2 values. how tuned all but last share y with left

to tease apart the effect of codependent features on the response. Also, recall that there are vast armies of business analysts and scientists at work that need to analyze data, in a manner akin to exploratory data analysis (EDA), that have no intention of creating a predictive model. Either they have no need, perhaps needing only partial dependence plots, or they do not have the expertise to choose, tune, and assess models (or write software at all).

Even in the case where a machine learning practitioner is available to create a fitted model for the analyst, hazards exist. First, if a fitted model is unable to accurately capture the relationship between features and y accurately, for whatever reason, then partial dependence does not provide any useful information to the user. To make interpretation more challenging, there is no definition of "accurate enough." Second, given an accurate fitted model, business analysts and scientists are still peering at the data through the lens of the model, which can distort partial dependence curves. Separating visual artifacts of the model from real effects present in the data requires expertise in model behavior (and optimally in the implementation of model fitting algorithms).

Consider the combined FPD/ICE plots shown in Figure 1 derived from several models (random forest, gradient boosting, linear regression, deep learning) fitted to the same New York City rent data set Kaggle [2017]. The subplots in Figure 1(b)-(e) present starkly different partial dependence relationships and it is unclear which, if any, is correct. The marginal plot, (a), drawn directly from the data shows a roughly linear growth in price for a rise in the number of bathrooms, but this relationship is biased because of the dependence of bathrooms on other variables, such as the number of bedrooms. (e.g., five bathroom, one bedroom apartments are unlikely.) For real data sets with codependent features, the true relationship is unknown so it is hard to evaluate the correctness of the plots. (Humans are unreliable estimators, which is why we need data analysis algorithms in the first place.) Nonetheless, having the same algorithm, operating on the same data, give meaningfully different partial dependences is undesirable and makes one question their validity.

Experts are often able to quickly recognize model artifacts, such as the stairstep phenomenon inherent to the decision tree-based methods in Figure 1(b) and (c). In this case, though, the stairstep is more accurate than the linear relationship in (d) and (e) because the number of bathrooms is discrete (except for "half baths"). The point is that interpreting model-based partial dependence plots can be misleading, even for experts.

An accurate mechanism to compute partial dependences that did not peer through fitted models would be most welcome. Such partial dependence curves would be accessible to users, like business analysts, who lack the expertise to create suitable models and would also reduce the chance of plot misinterpretation due to model artifacts. The curves could also help machine learning practitioners to choose appropriate models based upon relationships exposed in the data.

In this paper, we propose a strategy, called STRATified Partial Dependence (StratPD), that (*i*) computes partial dependences directly from training data $(\mathbf{X}, \mathbf{y})$, rather than through the predictions of a fitted model, and (*ii*) does not presume mutually-independent features. As an example, Figure 1(f) shows the partial dependence plot computed by StratPD. The technique depends on the notion of an idealized partial dependence: integration over the partial derivative of y with respect to the variable of interest for the smooth function that generated $(\mathbf{X}, \mathbf{y})$. As that function is unknown, we estimate the partial derivatives from the data non-parametrically. Colloquially, the approach examines changes in y across x_j while holding $\mathbf{X}_{\backslash j}$ constant or nearly constant ($\mathbf{X}_{\backslash j}$ denotes all variables except x_j). A similar stratification approach works for categorical variables (CatStratPD). Both StratPD and CatStratPD are quadratic in n, in the worst case (like FPD), but StratPD behaves linearly on real data sets. Our prototype is currently limited to regression, isolates only single-variable partial dependence, and cannot identify interaction effects (as ICE can). The software is available via Python package `stratx` with source code at `github`, including the code to regenerate images in this paper.

We begin by describing and providing algorithms for the proposed stratification approach in Section 2 then compare StratPD to related (model-based) work in Section 3. In Section 4, we present partial dependence curves generated by StratPD and CatStratPD on real data sets, contrast the plots with those of existing methods, and use synthetic data to highlight biases in some model-based methods.

2 Partial dependence without model predictions

In special circumstances, we know the precise effect of each feature x_j on y. Assume we are given training data pair $(\mathbf{X}, \mathbf{y})$ where $\mathbf{X} = [x^{(1)}, \dots, x^{(n)}]$ is an $n \times p$ matrix whose p columns represent observed features and $\mathbf{y}$ is the $n \times 1$ vector of responses. For any smooth function $f : \mathbb{R}^p \rightarrow \mathbb{R}$ that precisely maps each $x^{(i)}$ to $y^{(i)}$, $y^{(i)} = f(x^{(i)})$, the partial derivative of y with respect to x_j gives the change in y holding all other variables constant. Integrating the partial derivative then gives the *idealized partial dependence* of y on x_j, the isolated contribution of x_j to y:

Definition 1 The *idealized partial dependence* of y on feature x_j for smooth generator function $f : \mathbb{R}^p \rightarrow \mathbb{R}$ evaluated at $x_j = z$ is the cumulative sum up to z:

$$PD_j(z) = \int_{min(x_j)}^{z} \frac{\partial y}{\partial x_j} dx_j \tag{1}$$

$PD_j(z)$ is the value contributed to y by x_j at $x_j = z$ and $PD_j(min(x_j)) = 0$. (We will denote Friedman's original definition as FPD to distinguish it from this definition.) The advantages of this definition are that it does not depend on predictions from a fitted model and is insensitive to codependent features. Although the underlining generator function is unknown, we can estimate its partial derivatives.

The key idea is to stratify $\mathbf{X}_{\backslash j}$ feature space into disjoint regions of observations where all $\mathbf{X}_{\backslash j}$ variables are approximately matched across the observations in that region. Within each $\mathbf{X}_{\backslash j}$ region, any fluctuations in the response variable are likely due to the variable of interest, x_j. Estimates of the partial derivative within a region are estimated discretely as the changes in y values between unique and ordered x_j positions: $(y^{(i+1)} - y^{(i)})/(x_j^{(i+1)} - x_j^{(i)})$ for all i in a region. This amounts to performing piecewise linear regression through the leaf observations, one model per unique pair of x_j values, and collecting the model β_1 coefficients to estimate partial derivatives. The overall partial derivative at $x_j = z$ is the average of all slopes, found in any region, whose x_j range spans z. Stratification occurs through the use of a decision tree fit to $(\mathbf{X}_{\backslash j}, \mathbf{y})$, whose leaves aggregate observations with equal or similar $\mathbf{X}_{\backslash j}$ features. StratPD only uses the tree for the purpose of partitioning feature space and never uses predictions from any model. See Algorithm 1 for more details.

For this approach to work, decision tree leaves must satisfactorily stratify $\mathbf{X}_{\backslash j}$. If the $\mathbf{X}_{\backslash j}$ observations in each region are not similar enough, the relationship between x_j and y is less accurate. Regions can also become so small that even the x_j values become equal, leaving a single unique x_j observation in a leaf. Without a change in x_j, no partial derivative estimate is possible and these nonsupporting observations must be ignored. A degenerate case occurs when identical or nearly identical x_j and x'_j variables exist. Flattening x_j as part of $\mathbf{X}_{\setminus j}$ would also flatten x'_j, leading to both exhibiting flat curves, as if the decision tree were trained on $(\mathbf{X}, \mathbf{y})$ not $(\mathbf{X}_{\setminus j}, \mathbf{y})$. Our experiments show that using the collection of leaves from a random forest, which restricts the number of variables available during node splitting, prevents partitioning from relying too heavily on either x_j or x'_j. Some leaves have observations that vary in x_j or x'_j and partial derivatives can still be estimated. ***TODO:*** maybe talk about how PD/ICE on RF underestimates the curve.

StratPD uses hyper parameter `min_samples_leaf` to control the minimum number of observations in each decision tree leaf. Generally speaking, smaller values lead to more confidence that fluctuations in y are due solely to x_j, but more observations per leaf allow StratPD to capture more nonlinearities and make it less susceptible to noise. As the leaf size grows, however, one risks introducing contributions from $\mathbf{X}_{\setminus j}$ into the relationship between x_j and y. At the extreme, the decision tree would consist of a single leaf node containing all observations, leading to a marginal not partial dependence curve.

StratPD uses another hyper parameter called `min_slopes_per_x` to ignore any partial derivatives estimated with too few observations. Dropping uncertain partial derivatives greatly improves accuracy and stability. Partial dependences computed by integrating over local partial derivatives are highly sensitive to partial derivatives computed at the left edge of any x_j's range because imprecision at the left edge affects the entire curve. This presents a problem when there are few samples with x_j values at the extreme left (see, for example, the x_j histogram of Figure 2(d)). Fortunately, sensible defaults for StratPD (10 observations and 5 slopes) work well in most cases and were used to generate all plots in this paper.

For categorical variables, CatStratPD uses the same stratification approach, but cannot apply regression of y to non-ordinal, categorical x_j. Instead, CatStratPD groups leaf observations by category and computes the average $\bar{y}$ per category. Then, within each leaf, it chooses a random reference category and subtracts that category's $\bar{y}$ value from the leaf $\bar{\mathbf{y}}$ vector to get a vector of relative deltas between categories: $\Delta\mathbf{y} = \bar{\mathbf{y}} - \bar{\mathbf{y}}_{refcat}$. The $\Delta\mathbf{y}$ vectors from all leaves are then merged via averaging, weighted by the number of observations per category, to get the overall effect of each category on y. The delta vectors for two leaves, $\Delta\mathbf{y}$ and $\Delta\mathbf{y}'$, can only be merged if there is at least one category in common. CatStratPD initializes a running average vector to the first leaf's $\Delta\mathbf{y}$ and then makes passes over the remaining vectors, merging any vectors with a category in common with the running average vector. Observations associated with any remaining, unmerged leaves must be ignored.

Stratification of high-cardinality categorical variables tends to create small groups of category subsets, which complicates the averaging process across groups. (Such $\Delta\mathbf{y}$ vectors are sparse and *NaN* represents missing values.) If both groups have the same reference category, merging is a simple matter of averaging the two delta vectors, where $mean(z{,}NaN) = z$. For delta vectors with different reference categories and at least one category in common, one vector is adjusted to use a randomly-selected reference category, c in common: $\Delta\mathbf{y}' = \Delta\mathbf{y}' - \Delta\mathbf{y}'_c + \Delta\mathbf{y}_c$. That equation adjusts $\Delta\mathbf{y}'$ so $\Delta\mathbf{y}'_c = 0$ then adds the corresponding value from $\Delta\mathbf{y}$ so $\Delta\mathbf{y}'_c = \Delta\mathbf{y}_c$, which renders the average of $\Delta\mathbf{y}$ and $\Delta\mathbf{y}'$ meaningful. CatStratPD uses a single hyper parameter `min_samples_leaf` to control stratification. See Algorithm 2 for more details.

StratPD and CatStratPD both have theoretical worst-case time complexity of $O(n^2)$ for n observations. For StratPD, the cost to compute y deltas for all observations among the leaves is linear in n and the cost to average slopes across unique x_j ranges is on the order of $|unique(\mathbf{X}_j)| \times n$ or n^2 when all $\mathbf{X}_j$ are unique in the worst case. StratPD is, thus, $O(n^2)$ in the worst case. CatStratPD computes category deltas linearly in n but must make (at most two) passes of size $|T|$ (number of leaves) over the leaves to average all possible leaf category delta vectors. Averaging two vectors costs n operations, so each pass requires $|T| \times n$. Since the number of leaves is roughly $|T| \approx n/$`min_samples_leaf`, each pass requires $n/$`min_samples_leaf` $\times n = n^2/$`min_samples_leaf` or $O(n^2)$. The number of passes is either 1 or 2, depending on data set.

3 Related work

FPD

ICE

ALE

SHAP

cold start, counting execution time and number of hyper parameters. particularly deep learning

The techniques differ in algorithm simplicity, performance, and ability to isolate codependent variables. a nonparametric technique could also inform which machine learning model to use if a model is desired.

SHAP is mean centered FPD for independent variables, proof in supplemental material.

4 Experimental results

what if X,y relationship is very weak? models would get low accuracy. what happens to us? I think we would simply show low partial dependence curves.

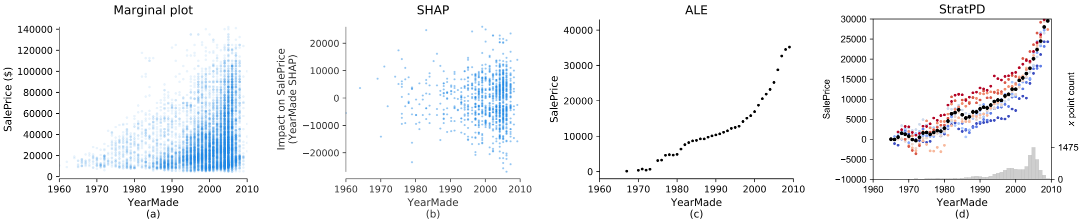

Figure 2: (a) Marginal plot of bulldozer `YearMade` versus `SalePrice` using subsample of 20k observations, (b) partial dependence drawn by SHAP interrogating an RF with 40 trees and explaining 1000 values with 100 observations as background data, (c) StratPD partial dependence.

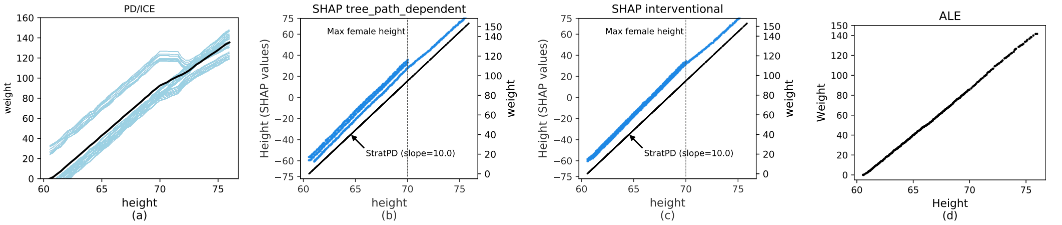

Figure 3: SHAP partial dependence plots of response body weight on feature `height` using 2000 synthetic observations from Equation (**??**). SHAP interrogated an RF with 40 trees and explained all 2000 samples; the interventional case used 100 observations as background data.

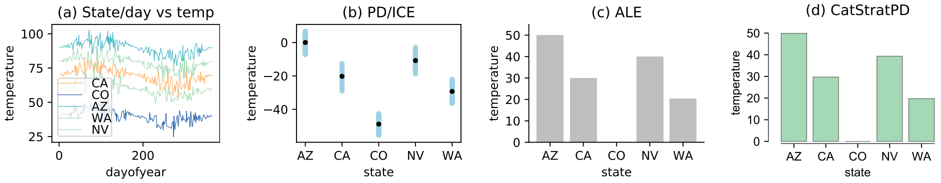

Figure 4: foo.

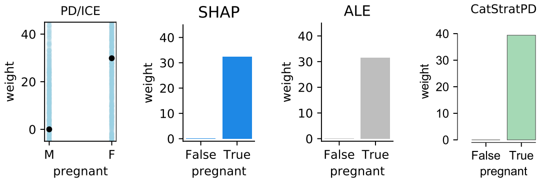

Figure 5: foo.

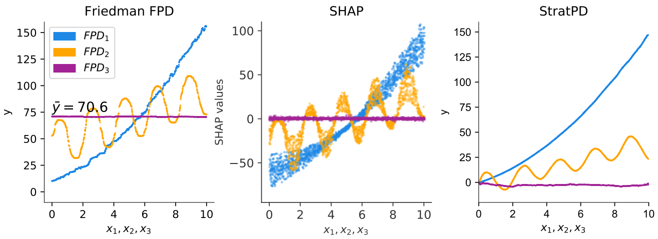

Figure 6: $y = x_1^2 + x_1x_2 + 5x_1 sin(3x_2) + 10$ where $x_1, x_2, x_3 \sim U(0, 10)$ and x_3 does not affect y. No noise added.

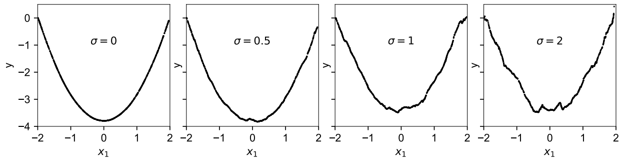

Figure 7: $y = x_1^2 + x_1 + 10 + N(0, \sigma)$ where $x_1, x_2 \sim U(-2, 2)$ and $\sigma \in [0, 0.5, 1, 2]$.

5 Discussion and future work

6 Appendix

Algorithm 1: $StratPD(\mathbf{X},\ \mathbf{y},\ j,\ min\_samples\_leaf,\ min\_slopes\_per\_x)$

```
$T$ := Decision tree regressor fit to $(\mathbf{X}_{\setminus j}, \mathbf{y})$ with hyper parameter: $min\_samples\_leaf$
for each leaf $L \in T$ do
    $(\mathbf{x}_L, \mathbf{y}_L) := \{(x_j^{(i)}, y^{(i)})\}_{i \in L}$    ▷ Get leaf observations
    $\mathbf{ux}, \bar{\mathbf{y}}$ := Group and sort $(\mathbf{x}_L, \mathbf{y}_L)$ by $x_j$ value, computing $\bar{y}$ per unique $x_j$ value
    $\mathbf{dx} := \mathbf{ux}^{(i+1)} - \mathbf{ux}^{(i)}$ for $i = 1..|\mathbf{ux}| - 1$    ▷ Discrete difference between adjacent
    $\mathbf{dy} := \bar{\mathbf{y}}^{(i+1)} - \bar{\mathbf{y}}^{(i)}$ for $i = 1..|\mathbf{ux}| - 1$
    Add tuples $(\mathbf{ux}^{(i)}, \mathbf{ux}^{(i+1)},\ \mathbf{dy}^{(i)}/\mathbf{dx}^{(i)})$ to list $\mathbf{D}$ for $i = 1..|\mathbf{ux}| - 1$
$\mathbf{ux} := unique(\mathbf{X}_j)$
for each $x \in \mathbf{ux}$ do    ▷ Count slopes and compute average slope per unique $x_j$ value
    $\mathbf{slopes} := [slope$ for $(a, b, slope) \in \mathbf{D}$ if $x \geq a$ and $x < b]$
    $\mathbf{c}_x := |\mathbf{slopes}|$
    $\mathbf{dydx}_x := \overline{\mathbf{slopes}}$
$\mathbf{dydx} := \mathbf{dydx}[\mathbf{c} \geq min\_slopes\_per\_x]$    ▷ Drop missing slopes, those computed with too few
$\mathbf{ux} := \mathbf{ux}[\mathbf{c} \geq min\_slopes\_per\_x]$
$\mathbf{pdx} := \mathbf{ux}^{(i+1)} - \mathbf{ux}^{(i)}$ for $i = 1..|\mathbf{ux}| - 1$
$\mathbf{pdy} := [0] +$ cumulative_sum$(\mathbf{dydx} * \mathbf{pdx})$    ▷ integrate, inserting 0 for leftmost $x_j$
return $\mathbf{pdx}, \mathbf{pdy}$
```

Algorithm 2: $CatStratPD(\mathbf{X}, \mathbf{y}, j, min\_samples\_leaf)$

```
$T$ := Decision tree regressor fit to $(\mathbf{X}_{\setminus j}, \mathbf{y})$ with hyper parameter: $min\_samples\_leaf$
Let $\Delta Y$ be a matrix whose columns are vectors of leaf category deltas
Let $C$ be a matrix whose columns are vectors for leaf category counts
for each leaf $L \in T$ do    ▷ Get average $y$ delta relative to random ref category for obs. in leaves
    $(\mathbf{x}_L, \mathbf{y}_L) := \{(x_j^{(i)}, y^{(i)})\}_{i \in L}$    ▷ Get leaf observations
    $\mathbf{ucats}$, $C_L := unique(\mathbf{x}_L)$    ▷ Get unique categories, counts from leaf observations
    $\bar{\mathbf{y}}$ := Group leaf records $(\mathbf{x}_L, \mathbf{y}_L)$ by category, computing $\bar{y}$ per unique category
    $refcat$ := random category from $\mathbf{x}_L$
    $\Delta Y_L = \bar{\mathbf{y}} - \bar{\mathbf{y}}_{refcat}$
$\Delta\mathbf{y}$, $\mathbf{c} := \Delta Y_1$, $C_1$    ▷ $\Delta\mathbf{y}$ is running average vector mapping category to average $y$ delta
Repeat twice
    for each leaf $L$ in work do    ▷ 2 passes is enough to merge all $\Delta Y_L$ into $\Delta\mathbf{y}$
        if $\Delta Y_L$ has category in common with $\Delta\mathbf{y}$ then
            $c$ = random category in intersection
            $\Delta\mathbf{y}_L := \Delta Y_L - \Delta Y_{L,c} + \Delta\mathbf{y}_c$    ▷ Adjust so $\Delta Y_{L,c} = 0$, add corresponding $\Delta\mathbf{y}_c$ value
            $\Delta\mathbf{y} := (\mathbf{c} \times \Delta\mathbf{y} + C_L \times \Delta\mathbf{y}_L)/(\mathbf{c} + C_L)$ where $z + NaN{=}z$    ▷ weighted average
            $\mathbf{c} := \mathbf{c} + C_L$    ▷ update running weight
return $\Delta\mathbf{y}$
```

References

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

Alex Goldstein, Adam Kapelner, Justin Bleich, and Emil Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, 24(1):44–65, 2015. doi: 10.1080/10618600.2014.907095. URL `https://doi.org/10.1080/10618600.2014.907095`.

Daniel W Apley. Visualizing the effects of predictor variables in black box supervised learning models. *arXiv preprint arXiv:1612.08468*, 2016.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf`.

Kaggle. Two sigma connect: Rental listing inquiries. `https://www.kaggle.com/c/two-sigma-connect-rental-listing-inquiries`, 2017. Accessed: 2019-06-15.