TreeIG:
Exact Integrated Gradients for Tree-Based Models

Ludger Hentschel

May 21, 2026

Abstract

Integrated Gradients, Accumulated Local Effects, and related attribution methods evaluate path integrals of local feature effects. Standard gradient-based implementations, using numerical integration, are naturally suited to smooth models with well-defined local gradients.

Tree-based models produce piecewise-constant scalar prediction functions whose local gradients are undefined at split boundaries and zero elsewhere. We show that, along any straight-line interpolation path, the distributional derivative is a finite sum of Dirac impulses at the split boundaries the path crosses. The path integral therefore reduces exactly to a finite crossing sum of prediction jumps. The resulting crossing-sum formulation extends integrated gradients and accumulated local effects to tree-based models, yields exact featurewise attribution up to floating-point precision, satisfies completeness without quadrature error, and replaces repeated high-dimensional numerical gradient estimation with sparse event detection and exact summation of prediction jumps.

The framework applies naturally to regression trees and to additive classification models in score or logit space.

www.ludgerhentschel.com ludger@ludgerhentschel.com

Contents

1 Introduction

Path integrals of gradients along interpolation paths arise throughout machine learning. Methods such as Integrated Gradients (Sundararajan et al., 2017) define feature attributions of the form

$$\phi_j(x; x_0) = (x_j - x_{0j}) \int_0^1 \frac{\partial f(x(t))}{\partial x_j}\, dt, \qquad j = 1, \ldots, p, \tag{1}$$

where $x(t)$ is the straight-line interpolation path from a baseline input x_0 to an observed input x. Related constructions appear in Expected Gradients (Erion et al., 2021), Accumulated Local Effects (Apley and Zhu, 2020), and other attribution frameworks based on integrating local feature effects along a path.

Because existing implementations rely on ordinary gradients evaluated along the interpolation path, they are generally limited to smooth models such as neural networks. Current implementations numerically approximate these path integrals using Riemann sums over M equally spaced interpolation points, with M often ranging from 20 to 1,000. We can check accuracy post hoc by verifying that the sum of attributed values approximately matches the realized change in the target quantity.

Although Integrated Gradients is often viewed as fundamentally tied to smooth models, we show that tree ensembles admit an exact path-integral representation in which the attribution reduces to a finite sum of prediction jumps at crossed split boundaries.

Tree-based models (Breiman et al., 1984; Breiman, 2001; Friedman, 2001; Ke et al., 2017) generate piecewise constant step functions that raise immediate concerns about the existence of gradients $\nabla_x f(x(t))$ and the ability to integrate them numerically. We confirm that those concerns are justified. More importantly, we show that, for tree-based models, integrated gradients reduce to a sum of steps along the path from the baseline x_0 to the observed input x. The key observation is that tree-based prediction functions are piecewise constant: their ordinary gradients vanish almost everywhere, while their distributional gradients concentrate at split boundaries. Integrating those boundary contributions yields exactly the jump in the prediction function at each crossed split. This sum of discrete crossing contributions is the exact evaluation of the path integral in equation (1) for tree-based models, not a numerical approximation. We call this TreeIG. For classification trees with additive score representations, TreeIG applies to the raw class score or logit, rather than to the post-link probability.

An open-source implementation is available in the TreeIG Python package (Hentschel, 2026b).

We can interpret TreeIG as the exact Integrated Gradients calculation for a piecewise-constant approximation to a smooth function. As we refine the tree approximation, the discrete crossing measure converges to the ordinary gradient density of the smooth function along the path, so the TreeIG attributions converge to the corresponding smooth-model Integrated Gradients. Appendix B confirms this intuition with an example.

Existing tree-specific attribution methods such as TreeSHAP (Lundberg et al., 2018) address a different Shapley-value (Shapley, 1953) attribution problem. The crossing-sum formulation extends the original Integrated Gradients path-integral framework from smooth models to tree-based machine learning models. Since the sum replaces numerical gradients and numerical integration with sparse event detection and summation of known jump contributions, it can also be computationally efficient, even for large ensembles of complex trees.

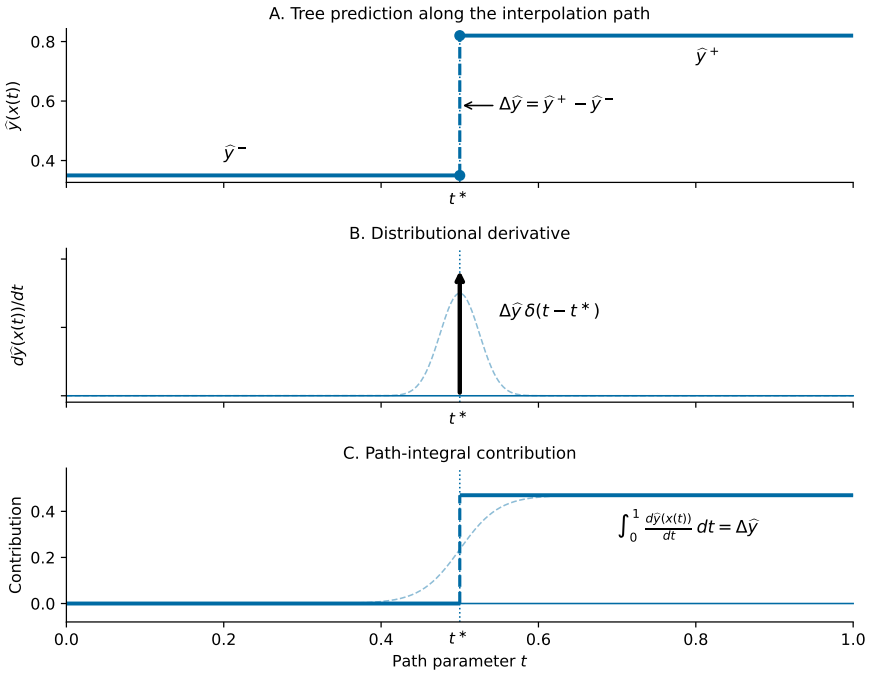
The total path integral equals the endpoint difference through the generalized fundamental theorem of calculus; this scalar identity continues to hold for piecewise-constant models under distributional differentiation. The substantive result is that the *featurewise* attribution is also exactly computable: each crossing contribution is naturally assigned to the feature whose split boundary is crossed, eliminating quadrature error and yielding completeness up to floating-point precision. As with standard Integrated Gradients, the resulting attribution depends on the choice of baseline and interpolation path.

The derivation applies not only to prediction functions but also to losses, fit measures, and other scalar functionals evaluated along the path. This note develops the crossing-sum formulation rigorously using distributional derivatives, establishes the resulting attribution formula for scalar tree outputs, including regression predictions and classification scores, and illustrates its numerical and computational properties relative to standard Riemann approximations for tree ensembles.

2 Path Integrals for Piecewise-Constant Models

The standard path-integral framework assumes that the target scalar function is differentiable along the interpolation path. For regression trees this scalar is the prediction function; for additive classifiers it is the class score before probability normalization.

Figure 1: Integral of Distributional Gradient



The figure illustrates the distributional derivative and path integral around a tree branch crossing indexed by the path parameter t^*.

The top panel shows the prediction function $\widehat{y}(x(t))$; the middle panel shows its distributional derivative, $d\widehat{y}(x(t))/dt = \Delta\widehat{y}\delta(t - t^*)$, where $\delta(t - t^*)$ is the Dirac delta function evaluated at t^*; and the bottom panel shows the accumulated path-integral contribution around t^*.

In Panels B and C, faint dashed curves illustrate smooth approximations converging to the exact distributional representations of the derivative and integral.

At first glance, Integrated Gradients may appear fundamentally mismatched with piecewise-constant tree models because classical derivatives vanish almost everywhere and are undefined at split boundaries. The path-integral formulation resolves this difficulty naturally. Rather than introducing numerical approximation error through quadrature, the tree structure permits an exact finite decomposition in which the attribution reduces to the sum of prediction jumps at split boundaries crossed along the integration path.

2.1 Distributional Derivatives for Step Functions

Figure 1 illustrates why the integrated gradients of a step function reduce to localized jump contributions at split crossings.

Consider the one-dimensional step function

$$f(x) = a + (b - a)\mathbf{1}_{x \geq x^*}, \tag{2}$$

representing a tree split at threshold x^* that changes the prediction from a to b.

The ordinary derivative of f is zero away from the split boundary and undefined at x^*. In the distributional sense (Friedlander and Joshi, 1998),

$$\frac{d}{dx}\mathbf{1}_{x\geq x^*} = \delta(x - x^*), \tag{3}$$

where $\delta(x - x^*)$ denotes a Dirac impulse concentrated at the point $x = x^*$. The distributional derivative of the step function is

$$f'(x) = (b - a)\delta(x - x^*). \tag{4}$$

The distributional derivative localizes entirely at the split boundary, with total mass equal to the jump size $(b - a)$.

This representation also arises naturally as the limit of smooth approximations. If the discontinuity is replaced by a smooth transition with bandwidth ε, the ordinary derivative becomes increasingly concentrated near x^* as $\varepsilon \to 0$. In the distributional limit, the ordinary derivatives converge to $(b-a)\delta(x-x^*)$. The Dirac impulse is the canonical distributional derivative of a tree split.

2.2 Path Integrals Localize at Split Crossings

Now consider the straight-line interpolation path

$$x(t) = x_0 + t(x - x_0), \qquad t \in [0, 1]. \tag{5}$$

Suppose the path crosses a split boundary on feature j at threshold x_j^*. When $x_j \neq x_{0j}$, the crossing occurs at

$$t^* = \frac{x_j^* - x_{0j}}{x_j - x_{0j}}, \tag{6}$$

provided $t^* \in [0, 1]$. Crossings at $t^* = 0$ or $t^* = 1$ correspond to cases where the baseline or realized input lies exactly on a split threshold. In practice, tree implementations must apply the corresponding branch convention at such endpoint crossings. When $x_j = x_{0j}$, the path does not move in dimension j and no split on feature j can be crossed.

Along the interpolation path, the tree prediction $\hat{y}(x(t))$ is piecewise constant with a jump

$$\Delta\hat{y} = \hat{y}^+ - \hat{y}^-, \tag{7}$$

where $\hat{y}^-$ and $\hat{y}^+$ denote the prediction values immediately before and after the crossing.

Differentiating with respect to the path parameter t in the distributional sense yields

$$\frac{d\hat{y}(x(t))}{dt} = \Delta\hat{y}\,\delta(t - t^*). \tag{8}$$

The derivative with respect to the path parameter is zero almost everywhere and concentrated entirely at the split crossing location. The path integral localizes at the crossing point

$$\int_0^1 \frac{d\hat{y}(x(t))}{dt}\,dt = \Delta\hat{y}. \tag{9}$$

The gradient contribution of a tree split is not distributed smoothly along the interpolation path. It is concentrated entirely at the split crossing. Numerical Integrated Gradients approximations implicitly attempt to recover this localized impulse using dense interpolation grids and numerical gradient approximations.

Although the straight-line path passes through intermediate values for discrete features, this is less problematic for tree models than for smooth models. A tree prediction is constant between split thresholds, so intermediate path values have no effect except when they determine whether a split boundary is crossed. For a binary feature, any threshold between 0 and 1 defines the same split on observed data; TreeIG attributes the resulting prediction jump when the path moves from 0 to 1 or from 1 to 0. The attribution therefore matches the fitted tree exactly, even though the crossing location along the continuous path is not substantively meaningful.

TreeIG evaluates integrated gradients exactly for tree-based models using a distributional derivative representation; it does not construct locally smooth approximations to the prediction surface or linear approximations to the gradients (Wycoff, 2025). Although the prediction function is discontinuous and the distributional derivatives are singular at split boundaries, their path integrals remain finite and yield exact additive contributions.

2.3 Feature Attribution via Crossing Sums

For a tree with multiple splits, or an ensemble with many trees, the path may cross Q split boundaries at times

$$0 \le t_1 \le \cdots \le t_Q \le 1, \tag{10}$$

with associated prediction jumps $\Delta\hat{y}_1, \ldots, \Delta\hat{y}_Q$ on features $j_1, \ldots, j_Q$. For an ensemble, the crossing sequence is pooled across all trees: each tree contributes its own set of crossings, and the same feature may appear in splits across multiple trees.

By linearity of the distributional derivative,

$$\frac{d\hat{y}(x(t))}{dt} = \sum_{q=1}^{Q} \Delta\hat{y}_q \, \delta(t - t_q). \tag{11}$$

Integrating over the interpolation path yields

$$\hat{y}(x) - \hat{y}(x_0) = \sum_{q=1}^{Q} \Delta\hat{y}_q. \tag{12}$$

Each jump contribution is naturally attributed to the feature whose split boundary is crossed. The attribution for feature j is

$$\phi_j = \sum_{q:\, j_q = j} \Delta\hat{y}_q. \tag{13}$$

This is the natural distributional extension of the coordinatewise Integrated Gradients formula to piecewise-constant functions. For a split crossing on feature j, the distributional partial derivative contributes

$$\frac{\Delta\hat{y}_q}{x_j - x_{0j}} \tag{14}$$

at the crossing point. Multiplying by the Integrated Gradients scaling factor $(x_j - x_{0j})$ recovers the jump contribution $\Delta\hat{y}_q$. The crossing-sum formulation is not an approximation to Integrated Gradients but its exact evaluation for piecewise-constant models.

The resulting attribution satisfies completeness,

$$\sum_{j=1}^{p} \phi_j = \hat{y}(x) - \hat{y}(x_0), \tag{15}$$

because the prediction jumps telescope along the interpolation path:

$$\sum_{j=1}^{p} \phi_j = \sum_{q=1}^{Q} \Delta\hat{y}_q = \sum_{q=1}^{Q} \left(\hat{y}_q^{+} - \hat{y}_q^{-}\right) = \hat{y}(x) - \hat{y}(x_0). \tag{16}$$

The telescoping argument depends only on the sequential structure of the prediction jumps. The stronger result established above is that TreeIG is also the exact evaluation of the Integrated Gradients path integral for piecewise-constant models.

TreeIG provides an exact path-integral attribution method for tree-based models, with no quadrature error and no dependence on interpolation step count.

Because the crossing-sum formulation is the exact evaluation of the Integrated Gradients path integral for piecewise-constant models, TreeIG inherits the standard axiomatic properties of Integrated Gradients (Sundararajan et al., 2017), including completeness, linearity, and sensitivity. The method does not introduce a new attribution functional specialized to trees; rather, it extends the original Integrated Gradients construction to nonsmooth prediction functions through distributional derivatives.

Although integrated gradients applications to tree models require generalized distributional derivatives, the corresponding path integrals are remarkably simple: they reduce exactly to the sum of prediction changes at split crossings along the interpolation path.

In the generally rare case where the interpolation path crosses multiple split boundaries simultaneously, there is no obvious allocation rule. Simple rules allocate the entire change to one feature or equally split the step across all affected features. Alternatively, we could allocate the jump by averaging over all infinitesimal orderings of the tied crossings. Starting just before the tied crossing, we can consider all axis-aligned paths that cross the tied split features one at a time and average the resulting marginal contributions. This local Shapley-style allocation uses the nearby step geometry while preserving symmetry across the tied features. In the absence of interaction among the tied splits, this reduces to an equal split.

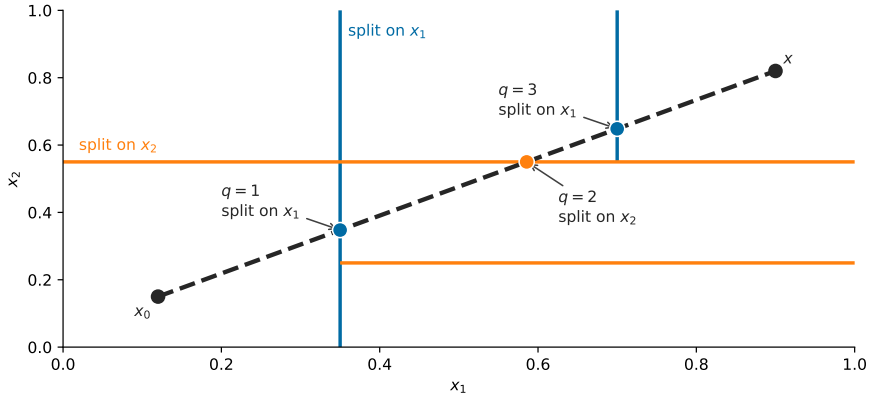
2.4 Enumerating Split Crossings

TreeIG is computationally practical because the set of split crossings along a straight-line interpolation path is finite and can be enumerated directly from the tree structure. Figure 2 illustrates this.

Once again, consider the interpolation path from the baseline x_0 to the actual input x in equation (5). Suppose a tree contains a split on feature j at threshold c. A crossing occurs when the interpolation path satisfies $x_j(t) = c$. Substituting the interpolation path yields

$$x_{0j} + t(x_j - x_{0j}) = c, \tag{17}$$

Figure 2: Boundary Crossing on a Two-Dimensional Tree



Boundary crossings along a two-feature interpolation path. The tree partitions the feature space into axis-aligned regions. Vertical segments correspond to splits on x_1 and horizontal segments correspond to splits on x_2. Only active split segments are drawn; deeper splits apply only within the regions defined by their parent nodes. The prediction function $\widehat{y}$ is piecewise constant within each region and changes by discrete jumps at split boundaries.

The straight-line interpolation path from x_0 to x crosses a finite sequence of split boundaries. Each realized prediction jump is assigned to the feature whose split boundary is crossed, yielding the crossing-sum attribution.

so the candidate crossing time is

$$t = \frac{c - x_{0j}}{x_j - x_{0j}}. \tag{18}$$

A candidate crossing contributes to the path integral only if

1. $t \in [0, 1]$, so the threshold lies on the interpolation segment between the baseline and realized feature values; and
2. the interpolation path passes through the corresponding branch of the tree.

Each split contributes at most one candidate crossing time along the interpolation path. Algorithm 1 states the enumeration procedure.

A split on feature j at threshold c is active along the interpolation path at candidate time t^* if and only if the interpolation path enters the parent node of that split immediately before the crossing under the tree implementation's routing convention. For interior crossings $t^* \in (0, 1)$, we evaluate this using the point $x(t^* - \epsilon)$ for infinitesimally small $\epsilon > 0$. For endpoint crossings at $t^* = 0$ or $t^* = 1$, the corresponding endpoint input and the implementation's branch convention at equality determine the active branch. Practically, we verify activity by evaluating the tree routing immediately before the crossing event: if traversal reaches the candidate node, the crossing is valid; otherwise the split lies in an unreachable branch and is discarded.

Algorithm 1 Enumerate split crossings along a straight-line path

Require: Tree structure $\mathcal{T}$, baseline x_0, observed point x
Ensure: Sorted crossing times $t_1 < \cdots < t_Q$ with associated features $j_1, \ldots, j_Q$ and jump sizes $\Delta\hat{y}_1, \ldots, \Delta\hat{y}_Q$

$C \leftarrow \emptyset$
for each active split on feature j at threshold c in $\mathcal{T}$ **do**
  if $x_j \neq x_{0j}$ **then**
    $t \leftarrow (c - x_{0j}) \,/\, (x_j - x_{0j})$
    if $t \in [0, 1]$ **and** path passes through this branch **then**
      Evaluate $\hat{y}^- = \hat{y}(x(t^-))$ and $\hat{y}^+ = \hat{y}(x(t^+))$
      $C \leftarrow C \cup \{(t,\ j,\ \hat{y}^+ - \hat{y}^-)\}$
    end if
  end if
end for
Sort C by t in increasing order
return C

For an ensemble, we apply algorithm 1 to each tree and pool the results. After sorting, we obtain the prediction jump at each crossing by evaluating the ensemble prediction immediately before and after the crossing, with endpoint crossings handled according to the tree implementation's branch convention at split thresholds. The resulting crossing sequence completely determines the path-integral attribution.

For shallow and moderately deep trees, the number of realized crossings Q is typically much smaller than the number of interpolation points required by Riemann approximations for highly-nonlinear functions. TreeIG replaces dense numerical quadrature with sums over sparse events. Moreover, the crossing-sum is a simple sum at Q points whereas numerical integration evaluates high-dimensional gradients at M quadrature nodes. Even if Q is larger than M, the computations are much simpler.

Appendix A extends this approach to scalar functions that are useful in attribution methods focused on model fit instead of predictions.

2.5 Classification Models and Logit Attribution

TreeIG also applies naturally to additive tree classifiers. Boosted classification models represent the prediction function as an additive ensemble of tree contributions in score or logit space,

$$g(x) = \sum_{m=1}^{M} T_m(x), \tag{19}$$

with class probabilities obtained only after application of a nonlinear link function such as the logistic or softmax transformation. We can apply TreeIG

to the raw class score rather than to the final probability, following the standard convention in Integrated Gradients attribution for classification models. The crossing-sum decomposition remains exact because the score function is additive across trees and piecewise constant within each tree partition.

For multiclass models, we can apply TreeIG separately to the score function for each class. The resulting attribution satisfies completeness relative to the corresponding class score difference,

$$\sum_j \phi_j(x; x_0) = g_k(x) - g_k(x_0), \tag{20}$$

where $g_k(x)$ denotes the score for class k.

TreeIG is designed for tree ensembles with additive score representations. Some classification ensembles, including random forests and extremely randomized trees, instead aggregate class probabilities or vote shares. These models do not possess an intrinsic additive score decomposition prior to probability normalization. As a result, exact Integrated-Gradients-style path attribution is less natural in such settings, and TreeIG does not consider these probability-averaging classifiers.

2.6 Application to Expected Gradients

Expected Gradients (Erion et al., 2021) extends Integrated Gradients by averaging path-integral attributions across random baseline points drawn from a reference distribution. For tree-based models, each individual Integrated Gradients computation again reduces exactly to a crossing sum. For tree-based models, Expected Gradients are expectations of crossing-sum attributions over the baseline distribution, eliminating the numerical quadrature error present in standard implementations.

2.7 Application to Accumulated Local Effects

Accumulated Local Effects (Apley and Zhu, 2020) characterize the marginal effect of a feature x_j on model predictions by integrating the local partial derivative over the feature's range. Formally, the first-order ALE for feature j evaluated at z is

$$\mathrm{ALE}_j(z) = \int_{z_0}^{z} \mathbb{E}_{X_{-j}|X_j=u}\left[\frac{\partial f(u, X_{-j})}{\partial u}\right] du, \tag{21}$$

where the expectation averages over the conditional distribution of the remaining features X_{-j}. For tree-based models, the partial derivative with

respect to feature j is distributionally concentrated at split thresholds on feature j. Accordingly, the ALE integral also reduces to a crossing-sum construction. Integrating over u accumulates one jump contribution each time u crosses a split threshold on feature j.

For tree-based models, we can compute the path integral component of the ALE construction exactly via crossing sums, once again eliminating the numerical integration error associated with finite differencing or interpolation along the feature dimension.

The first-order local-effect interpretation is especially natural for tree models because prediction changes occur through finite jumps at split boundaries. Higher-order ALE extensions are less straightforward. While we can still represent interaction effects through finite differences or distributional constructions, they depend on the geometry of interacting partition boundaries rather than simple jump discontinuities alone.

2.8 Relation to Existing Approximations

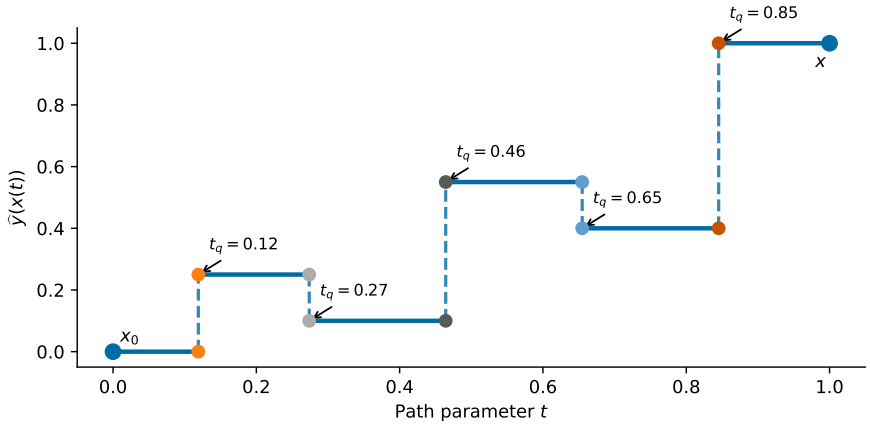
Numerical integrated gradients approximate the path integral using local finite-difference gradients evaluated near interpolation nodes. For tree models, the prediction function is discontinuous and the classical gradient is zero almost everywhere. If a finite-difference window does not straddle a threshold crossing, the estimated gradient is zero. If the window straddles a threshold, the estimated gradient becomes large over the associated quadrature interval. Consequently, the approximation error depends sensitively on the placement and spacing of the interpolation grid. Contribution estimates are often too small because the gradient calculations miss steps in the prediction function but can also be too large if a few gradient calculations straddle steps and infer large gradients over a wide range.

TreeIG identifies all realized split crossings along the interpolation path, accurately sums the contributions without numerical approximations, and requires no tuning parameters, like the number of nodes for numerical integration or the step size for numerical differentiation.

2.9 Relation to TreeSHAP

Integrated gradients and TreeIG differ fundamentally from TreeSHAP (Lundberg et al., 2018) and other Shapley-based attribution methods. TreeSHAP computes feature contributions from conditional expectation counterfactuals averaged over feature subsets. TreeIG instead computes a path-integral decomposition along a specified interpolation path that is exact, up to the numerical precision of sums.

Figure 3: Single Tree Example



The figure illustrates a simple tree to demonstrate the numerical accuracy of the crossing sum compared to the accuracy of numerical integration using quadrature. Table 1 shows the associated accuracy results.

The two approaches answer different attribution questions. TreeSHAP is a Shapley-value attribution method; TreeIG is a path-integral attribution method specialized to piecewise-constant models.

3 Numerical Illustration

We present two examples. Section 3.1 uses a synthetic single-tree model to illustrate the crossing-sum construction transparently. Section 3.2 uses a gradient-boosted ensemble on a standard tabular regression dataset to evaluate completeness accuracy and computational cost relative to numerical Integrated Gradients approximations. We also show a small comparison to TreeSHAP feature importance for this data set.

The examples are intentionally brief. An open-source implementation of TreeIG, including support for scikit-learn, XGBoost, and LightGBM tree models, is available in the TreeIG Python package (Hentschel, 2026b).

3.1 Single Regression Tree

We first consider a one-dimensional regression tree with 5 split thresholds, baseline value x_0, and realized value x. The tree is illustrated in figure 3. Along the straight-line interpolation path $x(t) = x_0 + t(x - x_0)$ for $t \in [0, 1]$, the prediction function is piecewise constant and changes only when the path crosses a tree split.

We compute the exact path integral using the crossing-sum formula in (13), which evaluates only the realized prediction jumps at the crossing

Table 1: Relative Accuracy of Integrated Gradients

M	Riemann approximation	Exact crossing sum	Riemann error
5	0.000	1.000	1.000
20	1.155	1.000	0.155
100	1.155	1.000	0.155
1,000	1.008	1.000	0.008

The table compares the accuracy of numerical integration using conventional Riemann sums and the exact crossing sum we develop for tree-based models. Figure 3 shows the example tree for this example.

The columns show the number of quadrature nodes M, the numerical Riemann estimate of the forecast change, the true forecast change, and the error of the estimate.

points. We compare this result to standard Riemann approximations using $M \in \{5, 20, 100, 1000\}$ equally spaced interpolation points.

Table 1 summarizes the results. For each value of M, we report the absolute approximation error relative to the crossing-sum result. TreeIG is exact up to floating-point precision because it evaluates the realized jump events directly. The Riemann approximation converges only gradually as the interpolation grid becomes sufficiently dense to localize the split crossings. At $M = 5$, all of the quadrature nodes happen to be so far from the split points that the local gradient calculations are all zero. For this example, reducing the approximation error below 1% requires roughly $M = 1{,}000$ interpolation nodes.

3.2 Gradient-Boosted Ensemble

We next consider a high-dimensional gradient-boosted tree ensemble (Friedman, 2001) fit to a standard tabular dataset. We use the Ames housing dataset (De Cock, 2011) with the log sale price as the outcome. There are 2,930 total observations and 80 base features, including a mix of numerical variables and categorical attributes (e.g., neighborhood, quality ratings). We preprocess all variables using median imputation, standardization, and one-hot encoding. After one-hot encoding the categorical features, there are 292 features. When reporting feature importance, we group the contributions from all the dummies corresponding to a single categorical feature.

We train the gradient boosting model on the full sample and then compute feature importance in the same sample. We use standard, default hyperparameters.

We use the empirical mean of the evaluation sample inputs in the transformed feature space as the baseline. This choice provides a neutral anchor for the path decomposition and centers the attribution on deviations from the evaluation distribution. The baseline is not intended to represent a feasible

Table 2: Relative Accuracy of Integrated Gradients: GBM

Method	M	$\lvert\widetilde{\varepsilon}_i\rvert$ median	q_{95}	max	Time
TreeIG		0.00	0.00	0.00	0.0088
Riemann sum	20	1.00	8.46	45.09	54
	200	4.45	22.66	73.34	549
	1,000	0.86	6.34	12.43	2,764

The table compares the accuracy of numerical integration using conventional Riemann sums and the exact crossing sum we develop for gradient boosted tree models.

The columns show the number of quadrature nodes M, the median relative error, the 95th percentile of the relative error, the maximum relative error, and the run time in seconds.

The statistics are for the absolute values of errors as a fraction of the total. The statistics are based on individual residuals for each observation i from a total sample of 2,930 observations in the Ames housing data. The crossing sum calculations consider all observations in the data set. To speed computations, the Riemann sums consider only 100 observations.

observation or to reproduce the joint dependence structure of the inputs; it serves only as the reference point from which realized feature variation is measured.

For each observation, we compute feature attributions using both the exact crossing-sum formulation and standard Integrated Gradients approximations based on Riemann quadrature with $M \in \{20, 200, 1000\}$ interpolation points.

TreeIG evaluates the realized split crossings encountered along the interpolation path and accumulates the associated prediction jumps. The numerical Integrated Gradients approximation instead evaluates finite-difference gradient approximations across all features at each quadrature point.

For each method we compute relative completeness residuals. For observation i, the completeness residual is

$$\varepsilon_i = \left| \sum_{j=1}^{p} \hat{\phi}_{ij} - \Big(f(x_i) - f(x_0) \Big) \right|. \tag{22}$$

To normalize for the scale of the prediction change, we focus on the proportional completeness residual

$$\widetilde{\varepsilon}_i = \frac{\left| \sum_{j=1}^{p} \hat{\phi}_{ij} - \Big(f(x_i) - f(x_0) \Big) \right|}{|f(x_i) - f(x_0)|}. \tag{23}$$

To avoid unstable ratios for observations whose endpoint prediction change is essentially zero, we compute proportional residuals only for observations

satisfying

$$|f(x_i) - f(x_0)| > \tau, \qquad \tau = 10^{-6}\,\text{median}_i\,|f(x_i) - f(x_0)|. \tag{24}$$

Because TreeIG evaluates the realized jump events directly, the completeness residual is zero up to floating-point precision. For finite Riemann approximations, the completeness residual is generally nonzero because the discontinuous prediction jumps are only approximately localized by the interpolation grid.

We apply the finite-difference IG approximation to the full ensemble prediction, not to the individual trees separately. This gives the numerical method the smoothing benefit of aggregation across trees. TreeIG, by contrast, uses linearity to compute exact tree-level crossing sums and aggregate them.

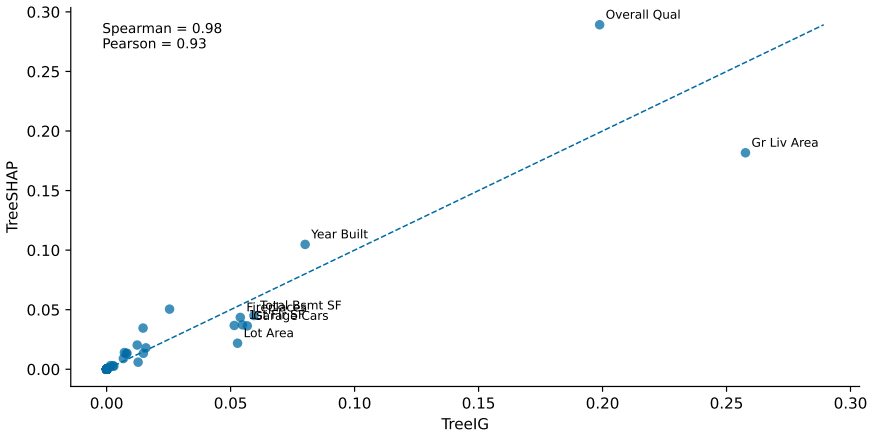
We report TreeIG computations for the full dataset, whereas we restrict the numerical IG calculations to 100 observations because of computational cost. Despite this favorable setup for numerical IG, TreeIG remains substantially faster while achieving exact completeness. The compute time for TreeIG includes the time required to parse the trees, find the crossing points along the path, and compute the sums.

Table 2 shows that Riemann approximations converge poorly for the tree ensemble fitted to the Ames housing data. Median, proportional completeness residuals remain large even at $M = 1{,}000$, which already requires roughly 2,700 times the wall time of TreeIG on this dataset. These results suggest that conventional numerical integration can be computationally inefficient and numerically unstable for integrated-gradient calculations on realistic tree ensembles. The absence of mainstream IG implementations for tree-based models suggests others have reached the same conclusion. TreeIG resolves this by design rather than by using finer grids and longer run times.

While we cannot compare our integrated gradient results to meaningful results based on conventional numerical integration, we can compare the integrated gradients to other feature importance measures. Figure 4 compares the feature importance assigned by the integrated gradients approach to TreeSHAP values for the same gradient boosting regression.

Since TreeSHAP and integrated gradients compute different notions of feature importance, these measures need not align perfectly, even when applied to the same model and data. TreeSHAP evaluates the marginal contribution of a feature across different subsets of observed features, averaging over many counterfactual feature combinations. Integrated gradients instead measure the accumulated change in the prediction along a specified interpolation path

Figure 4: Comparing Feature Importance for TreeSHAP and TreeIG



The figure compares proportional feature importance from TreeSHAP on the vertical axis and proportional feature importance from TreeIG on the horizontal axis. Both methods evaluate feature importance for the same gradient boosting regression model fitted to the Ames housing data.

from the baseline input to the realized observation. The two methods answer different attribution questions and need not assign the same importance to correlated or interacting predictors.

Nevertheless, the resulting feature importance rankings are similar in this example. Features identified as important by TreeSHAP, such as overall quality ("Overall Qual"), gross living area ("Gr Liv Area"), and year built ("Year Built"), also receive large integrated gradients attributions. The Spearman rank correlation between the proportional feature importance measures is 0.98, indicating that TreeIG produces feature attributions that are broadly consistent with established tree-specific attribution methods while preserving the path-integral interpretation of integrated gradients.

Table 3 shows run times for our integrated gradients computations and for TreeSHAP. The table shows that, for this example, TreeIG, implemented in Python, takes a similar amount of time as the compiled implementation of TreeSHAP.

The compute time for TreeIG depends on the complexity of the trees. The time scales roughly as $O(NTC)$, where N is the number of evaluated observations, T is the number of trees, and C is the average cost of parsing the trees. C rises with tree depth, number of active crossings, and node count. For highly complicated tree models, the calculations may be slower. In these situations, the method may run faster if we sample a subset of the observations or a subset of the tree ensemble. Running attribution over a subset of the observations is easily controlled by the user. Sampling a subset

Table 3: Example Run Times

Method	Time	Relative Time
TreeIG	0.0088	1.0
TreeSHAP	0.034	3.9

The table reports relative timings of the feature importance calculations for the Ames housing regression models. The model uses 80 raw features, which expand to 292 features after one-hot encoding categorial variables. We compute contributions across 2,930 observations.

All values are rounded to two significant figures and are intended to convey approximate computational cost rather than precise measurements.

of the trees generally requires

4 Summary

Path-integral attribution does not fundamentally require smoothness. For piecewise-constant models such as trees and boosted ensembles, distributional derivatives localize the path integral at split crossings, yielding an exact representation as a sum over discrete changes.

The resulting formulation produces exact feature attributions with exact completeness and no quadrature error. It replaces dense numerical integration with sparse crossing detection and provides an exact path-integral alternative to standard numerical Integrated Gradients implementations for tree-based models. The same construction applies to Accumulated Local Effects and to arbitrary scalar functionals of the prediction, including the loss-based attributions of Hentschel (2026a).

The method applies broadly to tree ensembles whose prediction rules can be represented as piecewise-constant partitions with explicit split boundaries and leaf predictions. This includes random forests and standard gradient-boosted tree frameworks such as sklearn trees, XGBoost, and LightGBM. Extensions to more specialized architectures, such as CatBoost (Prokhorenkova et al., 2018) with native categorical transformations, may require additional handling of internal feature encodings and split representations.

5 References

Apley, Daniel W., and Jingyu Zhu, 2020, Visualizing the effects of predictor variables in black box supervised learning models, *Journal of the Royal Statistical Society, Series B* 82 (4), 1059–1086.

Breiman, Leo, 2001, Random forests, *Machine Learning* 45, 5–32.

Breiman, Leo, Jerome Friedman, Richard Olshen, and Charles Stone, 1984, *Classification and Regression Trees* (Wadsworth, Belmont, CA).

De Cock, Dean, 2011, Ames, Iowa: Alternative to the Boston housing data as an end of semester regression project, *Journal of Statistics Education* 19 (3), 1–15.

Erion, Gabriel, Joseph D. Janizek, Pascal Sturmfels, Scott M. Lundberg, and Su-In Lee, 2021, Improving performance of deep learning models with axiomatic attribution priors and expected gradients, *Nature Machine Intelligence* 3 (7), 620–631.

Friedlander, F. Gerard, and Mark Joshi, 1998, *Introduction to the Theory of Distributions*, second edition (Cambridge University Press, Cambridge, England).

Friedman, Jerome H., 2001, Greedy function approximation: A gradient boosting machine, *The Annals of Statistics* 29 (5), 1189–1232.

Hentschel, Ludger, 2026a, Feature importance for predictive accuracy: An Euler decomposition, Working paper, Versor Investments, New York, NY.

Hentschel, Ludger, 2026b, TreeIG: Exact integrated gradients for tree-based models (Python software package, doi:10.5281/zenodo.20312653).

Ke, Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu, 2017, LightGBM: A highly efficient gradient boosting decision tree, in I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., *Advances in Neural Information Processing Systems*, volume 30 (Curran Associates, Inc.).

Lundberg, Scott M, Gabriel G Erion, and Su-In Lee, 2018, Consistent individualized feature attribution for tree ensembles, Working paper, University of Washington, Seattle, WA.

Prokhorenkova, Liudmila, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin, 2018, CatBoost: unbiased boosting with categorical features, in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, 6639–6649 (Curran Associates Inc., Red Hook, NY, USA).

Shapley, Lloyd S., 1953, A value for n-person games, *Contributions to the Theory of Games* 2, 307–317.

Sundararajan, Mukund, Ankur Taly, and Qiqi Yan, 2017, Axiomatic attribution for deep networks, in Doina Precup, and Yee Whye Teh, eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 3319–3328 (PMLR).

Wycoff, Nathan, 2025, Regression trees know calculus, Working paper, University of Massachusetts, Amherst, MA.

A Extension to General Scalar Functionals

The same attribution the main text derives for predictions applies more generally to any scalar functional evaluated along the interpolation path. This is valuable for feature attribution methods that focus on model fit.

Let $g(\hat{y})$ denote a scalar function of the prediction, such as a loss, likelihood, or fit measure.

At a split crossing, the contribution becomes

$$\Delta g = g(\hat{y}^+) - g(\hat{y}^-). \tag{25}$$

Summing over crossings yields

$$g(\hat{y}(x)) - g(\hat{y}(x_0)) = \sum_{q=1}^{Q} \Big(g(\hat{y}_q^+) - g(\hat{y}_q^-) \Big), \tag{26}$$

where $\hat{y}_q^-$ and $\hat{y}_q^+$ denote the prediction values immediately before and after the qth crossing. Equation (26) extends the crossing-sum formulation from prediction attribution to arbitrary scalar path functionals.

A.1 Loss Functions and Explained Fit

Hentschel (2026a) introduces the Euler Decomposition of Explained Fit (EDEF) for machine learning models. The attribution allocates realized model fit across features by integrating changes in a loss function along an interpolation path from a baseline feature vector to the observed feature vector, observation by observation.

For smooth models, these integrals are evaluated using gradients of the loss along the path. For tree-based models, the prediction is piecewise constant, so the loss is also piecewise constant along the path. The same crossing-sum formula therefore gives the exact loss contribution at each split crossing.

The canonical loss functions are squared error loss for regression and log loss for binary classification. In both cases, the contribution of a split crossing is the finite change in loss induced by the prediction jump at that crossing. The contribution to explained fit is the negative of this loss change.

Squared Error Loss

For squared error loss,

$$\ell(y, \hat{y}) = (y - \hat{y})^2, \tag{27}$$

the crossing contribution to loss is

$$\Delta\ell = (y - \hat{y}^+)^2 - (y - \hat{y}^-)^2 \tag{28}$$
$$= -2(y - \hat{y}^-)\Delta\hat{y} + (\Delta\hat{y})^2, \tag{29}$$

where $\Delta\hat{y} = \hat{y}^+ - \hat{y}^-$. The corresponding contribution to explained fit is

$$-\Delta\ell = (y - \hat{y}^-)^2 - (y - \hat{y}^+)^2 \tag{30}$$
$$= 2(y - \hat{y}^-)\Delta\hat{y} - (\Delta\hat{y})^2. \tag{31}$$

Summing these crossing contributions over all split crossings gives the total change in squared error loss, or equivalently the total change in explained fit, along the interpolation path.

Log Loss

For binary classification with a model that outputs predicted probabilities $\hat{y} \in (0, 1)$, as is standard for gradient-boosted classifiers, the log loss is

$$\ell(y, \hat{y}) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y}), \qquad y \in \{0, 1\}. \tag{32}$$

At a split crossing, the predicted probability changes from $\hat{y}^-$ to $\hat{y}^+$. The crossing contribution to log loss is

$$\Delta\ell = \ell(y, \hat{y}^+) - \ell(y, \hat{y}^-) \tag{33}$$
$$= -y \log\left(\frac{\hat{y}^+}{\hat{y}^-}\right) - (1 - y) \log\left(\frac{1 - \hat{y}^+}{1 - \hat{y}^-}\right). \tag{34}$$

The corresponding contribution to explained fit is

$$-\Delta\ell = \ell(y, \hat{y}^-) - \ell(y, \hat{y}^+) \tag{35}$$
$$= y \log\left(\frac{\hat{y}^+}{\hat{y}^-}\right) + (1 - y) \log\left(\frac{1 - \hat{y}^+}{1 - \hat{y}^-}\right). \tag{36}$$

As with squared error loss, summing over split crossings gives the total change in log loss and the exact contribution to explained fit. No linearization of the loss is required; the contribution is the finite change in loss across the prediction jump.

B Convergence to Smooth Integrated Gradients

This appendix illustrates the relationship between TreeIG and ordinary Integrated Gradients for smooth prediction functions. We show that when a

tree ensemble approximates a smooth nonlinear function, the corresponding TreeIG attributions converge to the Integrated Gradients of the underlying smooth model.

B.1 Simulation Design

We consider the smooth nonlinear prediction function

$$f(x_1, x_2) = \sin(x_1) + \frac{1}{2}x_2^2 + \frac{3}{4}x_1x_2. \tag{37}$$

We draw observations

$$(x_{i1}, x_{i2}) \sim N(0, I_2), \tag{38}$$

and define the response variable as

$$y_i = f(x_{i1}, x_{i2}). \tag{39}$$

The prediction function is smooth and continuously differentiable. We use the baseline input $x_0 = (0, 0)$.

For the smooth function in equation (37), the Integrated Gradients attributions are available in closed form,

$$\phi_1(x; x_0) = x_1 \int_0^1 \left[\cos(tx_1) + \frac{3}{4}tx_2\right] dt = \sin(x_1) + \frac{3}{8}x_1x_2, \tag{40}$$

and

$$\phi_2(x; x_0) = x_2 \int_0^1 \left[tx_2 + \frac{3}{4}tx_1\right] dt = \frac{1}{2}x_2^2 + \frac{3}{8}x_1x_2. \tag{41}$$

We fit gradient-boosted regression trees of increasing complexity to the simulated data. For each fitted tree ensemble, we compute TreeIG attributions using the exact crossing-sum formulation developed in the main text. We then compare the resulting TreeIG attributions to the exact Integrated Gradients of the underlying smooth function.

B.2 Simulation Results

Table 4 shows that, as the tree ensemble becomes more flexible, the fitted piecewise-constant prediction function converges toward the smooth target function in equation (37). Correspondingly, the TreeIG attributions converge toward the exact smooth-model Integrated Gradients.

Table 4: TreeIG convergence to IG

Model	Pred. RMSE	RMSE(ϕ_1)	RMSE(ϕ_2)	Corr(ϕ_1)	Corr(ϕ_2)	Mean Crossings
D1- 5	1.03	0.56	0.85	0.81	0.70	1.3
D1- 25	0.92	0.48	0.72	0.84	0.81	4.9
D2- 25	0.53	0.35	0.43	0.91	0.94	8.7
D2- 50	0.41	0.29	0.31	0.93	0.96	15.1
D2-100	0.35	0.26	0.26	0.95	0.97	29.0
D3- 50	0.33	0.27	0.24	0.94	0.97	25.1
D3-100	0.29	0.24	0.21	0.95	0.97	48.1
D3-200	0.25	0.22	0.19	0.96	0.98	89.9
D4-200	0.12	0.12	0.14	0.99	0.99	140.0

The table reports the correspondence between TreeIG from gradient-boosted tree approximations and analytical integrated gradients for the underlying smooth function. Each row corresponds to a different gradient boosting specification with varying tree depth and number of trees.

The columns report the root mean squared errors between the smooth prediction function and the tree approximation and between the smooth and tree-based attributions for feature 1 and 2, respectively; the correlations between the smooth and tree-based attributions for feature 1 and 2, respectively; and the average number of split crossing TreeIG encounters along the integration paths.

The text provides additional details on the simulation design and attribution calculations.

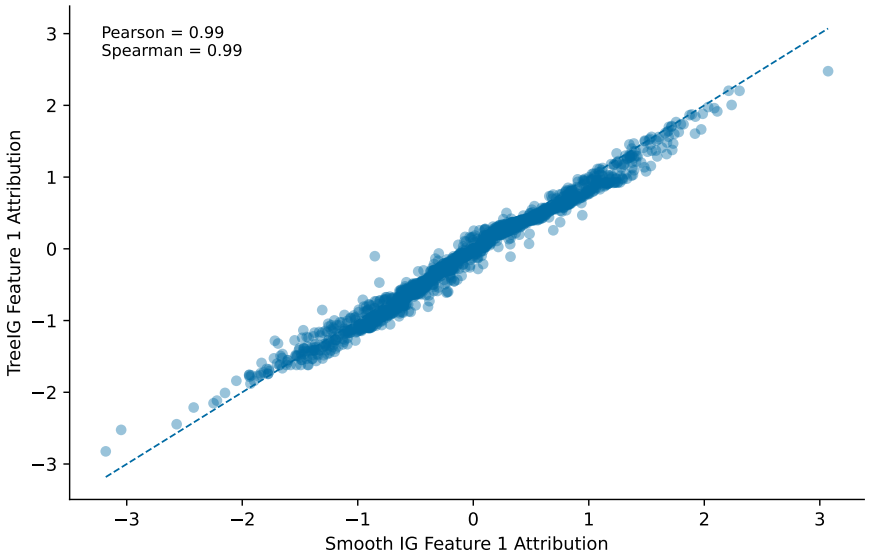
The rows contain results for gradient boosting models with different depths and different numbers of trees. The notation "D3-100" indicates a model with depth of 3 and 100 trees.

The first column shows the root mean squared prediction error between the fitted tree ensemble and the smooth target function in equation (37). The second and third columns show the root mean square error of the TreeIG attribution and the true integrated gradient attribution for features x_1 and x_2, respectively. The fourth and fifth columns show the correlations between the TreeIG and true integrated gradient attributions for features x_1 and x_2, respectively. The last column shows the average number of tree boundaries crossed by TreeIG.

The convergence is intuitive from the distributional representation of TreeIG. For a tree ensemble, the path derivative consists of a finite collection of impulses located at split crossings. As the partition becomes finer, the discrete crossing impulses increasingly approximate the continuous gradient density of the smooth function along the interpolation path.

Even relatively coarse tree approximations produce TreeIG attributions that are qualitatively similar to the smooth-model Integrated Gradients. The largest attribution discrepancies occur near regions where the smooth function varies rapidly and the tree approximation requires many local splits

Figure 5: Comparing Feature Attribution for TreeIG and IG



The figure compares TreeIG attributions to analytical integrated gradients for a smooth nonlinear function. We compute the integrated gradients analytically from the smooth function and compute TreeIG from a gradient-boosting approximation to that function. The gradient boosting model has depth 4 and 200 trees.

The underlying smooth function is

$$f(x_1, x_2) = \sin(x_1) + \frac{1}{2}x_2^2 + \frac{3}{4}x_1x_2.$$

The graph shows observation-level attributions for feature x_1 from 1,000 randomly generated evaluation samples.

to capture the curvature accurately.

Figure 5 compares the feature attribution for x_1 from a gradient boosting model with depth of 4 and 200 trees to the true feature attribution. The vertical axis shows TreeIG attribution for feature x_1; the horizontal axis shows the analytical Integrated Gradients attribution. For this case, there is excellent alignment between TreeIG and the true integrated gradient attribution for the smooth function. Even moderately sized tree ensembles produce TreeIG attributions that are very close to the smooth-model Integrated Gradients in this example.

In the limit of increasingly fine tree partitions, the TreeIG attribution converges to the smooth Integrated Gradients attribution along the interpolation path. This illustrates that TreeIG is not a fundamentally different attribution object from Integrated Gradients. Rather, TreeIG is the exact evaluation of the Integrated Gradients construction for a piecewise-constant approximation to the underlying prediction function.

Under increasingly fine tree partitions, the piecewise-constant tree ap-

proximation converges pointwise to the smooth target function, while the corresponding distributional derivatives converge weakly to the ordinary gradient field of the smooth function.