Fast ALE

Zizheng Zhang

November 2025

Abstract

We formalize the best-split-selection for Accumulated Local Effects (ALE) implemented by `search_best_split_point_ale` and its driver `search_best_split_ale`. The method evaluates candidate split positions induced by a chosen split feature, minimizing a global objective defined as the sum of interval-wise risks (SSE from sufficient statistics) over all features of interest. An $O(1)$ incremental update per rank step yields an overall $O(pn)$ sweep per split feature (plus sorting). For numeric split features, we apply a boundary stabilizer to reduce edge variance. For categorical split features, levels are ordered by the current split feature's sample-level signal (CART-style).

1 Problem Setting and Motivation

We consider a predictive model $f : \mathbb{R}^p \to \mathbb{R}$ and a set of features of interest $\mathcal{S} = \{1, \ldots, p\}$. GADGET constructs an interpretable tree by recursively partitioning the input space such that, within each node, the local feature effects are as homogeneous as possible across all $j \in \mathcal{S}$. This section recalls the ALE estimation, describes the representation used by GADGET, explains why a boundary stabilizer is needed, and motivates a fast algorithm by analyzing the computational cost of the original procedure.

1.1 ALE Estimation

For a given feature j, the first-order ALE function estimates its marginal effect on the prediction function $\hat{f}$ by (i) computing local prediction changes via finite differences (as an approximation to $\frac{\partial \hat{f}(\mathbf{x}_j, \mathbf{x}_{-j})}{\partial \mathbf{x}_j}$), (ii) averaging these local effects over the conditional distribution $P(\mathbf{x}_{-j} \mid \mathbf{x}_j)$, and (iii) accumulating these averaged local effects along the empirical range $\mathcal{X}_j$. Formally, the uncentered ALE function evaluated at $z \in \mathcal{X}_j$ is

$$\tilde{f}_j^{\mathrm{ALE}}(z) = \int_{z_{0j}}^{z} \mathbb{E}_{\mathbf{x}_{-j} | \mathbf{x}_j = z_j} \left[\frac{\partial \hat{f}(z_j, \mathbf{x}_{-j})}{\partial z_j} \right] dz_j.$$

Step 1: Interval construction. To approximate the integral in a data-supported manner, the range of $\mathbf{x}_j$ is partitioned into K_j intervals

$$[z_{0j}, z_{1j}],\ [z_{1j}, z_{2j}],\ \ldots,\ [z_{K_j-1,j}, z_{K_j,j}],$$

typically chosen as empirical quantiles so that each interval contains a comparable number of samples. Each observation i is mapped to exactly one interval,

$$\phi_j(i) = k \quad \Longleftrightarrow \quad x_{ij} \in [z_{k-1,j}, z_{k,j}],$$

which ensures that all subsequent finite-difference and averaging steps are computed only in regions with observed data.

Step 2: Local (finite-difference) effects. Within each interval k, ALE approximates the local effect of $\mathbf{x}_j$ by replacing the original feature value x_{ij} with the left and right interval boundaries. For each sample i with $\phi_j(i) = k$, the local prediction change is computed as

$$d_{ij} = \hat{f}(z_{k,j},\, \mathbf{x}_{i,-j}) \;-\; \hat{f}(z_{k-1,j},\, \mathbf{x}_{i,-j}),$$

where $\mathbf{x}_{i,-j}$ denotes all other features of sample i held fixed. This finite-difference quantity d_{ij} serves as a data-supported approximation to the partial derivative $\partial \hat{f}/\partial x_j$, evaluated over the interval $[z_{k-1,j},\, z_{k,j}]$.

Step 3: Interval-wise average effects. Within each interval k, the average local effect is

$$\frac{1}{n_{jk}} \sum_{i:\phi_j(i)=k} d_{ij}, \quad n_{jk} = \#\{i : \phi_j(i) = k\}.$$

Step 4: Average effects accumulation. The estimated uncentered ALE function for feature j is obtained by accumulating these interval means:

$$\hat{\tilde{f}}_j^{\mathrm{ALE}}(z) = \sum_{k=1}^{\phi_j(z)} \frac{1}{n_{jk}} \sum_{i:\phi_j(i)=k} d_{ij}.$$

GADGET does not use the ALE functions $\hat{\tilde{f}}_j^{\mathrm{ALE}}$ directly for splitting. Instead, it relies on the local effects d_{ij} and corresponding interval-wise sufficient statistics, because the splitting objective is formulated as a sum of interval-wise heterogeneity measures (SSE) across all features. For each feature $j \in \mathcal{S}$, GADGET stores:

- Per-sample local effects d_{ij};
- Interval indices $\phi_j(i) \in \{1, \ldots, K_j\}$ specifying to which ALE interval sample i belongs.

1.2 Boundary Instability in ALE-Based Local Effects

For continuous features, the ALE local effects can become highly unstable in regions where the true underlying effect exhibits abrupt interactions or "jumps." Smooth prediction models (e.g., neural networks or SVMs) typically do not place a sharp derivative exactly at the jump point; instead, they spread the change across a neighborhood around the jump. Consequently, even though a split at the true interaction point should, in principle, remove most of this heterogeneity, the finite-difference estimates d_{ij} in the jump neighborhood may still take unusually large values when the model fails to represent the abrupt change precisely.

This localized variance spike causes the heterogeneity objective to be dominated by a small number of jump-adjacent observations. Even if a split aligns with the conceptual jump, these unstable local effects can produce an erratic risk profile and lead to noise-driven split decisions. The resulting non-reducible heterogeneity cannot be handled by partitioning alone and therefore requires explicit stabilization.

1.3 Variance-Based Stabilization and Need for Efficiency

To prevent boundary-induced noise from driving split decisions, GADGET introduces a heuristic: around each candidate split, a small near window is compared to a far region on the same side. If the near-window variance exceeds the far-region variance by a substantial factor (e.g., a ratio ≥ 4), the near-window effects are treated as unreliable. Their SSE contribution

$$\text{risk}(n, s, q) = q - s^2/n$$

is replaced by the smoother value $n_{S,\text{near}}\hat{\sigma}^2_{S,\text{far}}$, using the far-region variance as a robust background estimate. This prevents local variance spikes from distorting the objective while preserving the global effect shape.

However, a naïve implementation recomputes near/far-window variances for all $O(n_j)$ candidate splits, each requiring $O(n_j)$ work, resulting in

$$O(n_j^2) \quad \text{per numeric feature.}$$

Combined with the interval-wise statistics across all features, this quadratic bottleneck makes ALE-based splitting impractical for large n_j.

The *Fast ALE* algorithm addresses this by maintaining all interval statistics via $O(1)$ incremental updates along the sorted order and by precomputing cumulative sums so that near/far-window variances are obtained in constant time. This reduces the full stabilized split search to

$$O(n_j \log n_j + pn_j),$$

making ALE-based splitting statistically robust and computationally scalable.

2 Inputs

Let $i = 1, \ldots, n$ index samples, $\mathcal{S} = \{1, \ldots, p\}$ be the set of *features of interest*. ALE effects are provided for the set $\mathcal{S}$ via a list `effect`: for each $j \in \mathcal{S}$ we have:

- Per-sample ALE effects d_{ij} (`dL`).
- Per-sample interval map $\phi_j(i) = k = 1, \ldots, K_j$ (`interval.index`).
- Per-interval sufficient statistics (n_{jk}, s_{jk}, q_{jk}) = (`int_n`, `int_s1`, `int_s2`) where n_{jk} is the count, $s_{jk} = \sum d_{ij}$ and $q_{jk} = \sum d_{ij}^2$ over samples within interval (j, k).

Risk from sufficient statistics. The interval-wise SSE is computed from sufficient statistics:

$$\text{risk}(n_{jk}, s_{jk}, q_{jk}) = \begin{cases} 0, & n_{jk} \leq 1, \\ q_{jk} - \dfrac{s_{jk}^2}{n_{jk}}, & n_{jk} \geq 2\ . \end{cases} \tag{1}$$

This matches `risk_from_stats` in code.

3 Candidate Order and Positions

We consider a candidate *split feature* j with observed values $\mathbf{z} = (z_i)_{i=1}^n$ over n samples, where $\mathbf{z}$ can be numeric or categorical. Define n_j as the number of non-missing entries of $\mathbf{z}$. We sort z ascending (or reorder categorical levels) and obtain a row order $\mathrm{ord}(1), \ldots, \mathrm{ord}(n_j)$.

We evaluate candidate split *ranks* t along the order induced by $\mathbf{z}$. The objective at rank t is a sum of *interval-wise* SSE risks over all features of interest, with a boundary stabilizer applied only if feature j is numeric.

3.1 Numeric split feature

Remove NA, sort values $\mathbf{z}_1 \le \cdots \le \mathbf{z}_{n_j}$ with row order $\mathrm{ord}(1), \ldots, \mathrm{ord}(n_j)$. Build a set of thresholds

$$C = \begin{cases} \{\text{unique quantiles of } \mathbf{z}\}, & \text{if } \texttt{n.quantiles} \text{ is given and } |\texttt{unique(z)}| \ge \texttt{n.quantiles}, \\ \texttt{unique(z)}, & \text{otherwise.} \end{cases}$$

Map each $c \in C$ to the rank of sorted feature values

$$t = \max\{u : z_{(u)} \le c\}.$$

Keep $t \in \{1, \ldots, n_j - 1\}$ and remove duplicates. Mark these positions as candidates $T \subseteq \{1, \ldots, n_j - 1\}$ (`is_cand`).

3.2 Categorical split feature (CART-like)

Let $L(i) \in \{1, \ldots, L\}$ denote the level of sample i after dropping unused levels. For each level ℓ, compute its average effect of the current split feature j:

$$\bar{d}_{\ell j} \;=\; \frac{1}{|\{i : L(i) = \ell\}|} \sum_{i : L(i) = \ell} d_{ij}.$$

Order levels by $\bar{d}_{\ell j}$ ascending, concatenate row blocks accordingly to get $\mathrm{ord}(1{:}n_j)$. Let c_r denote the cumulative sample size up to the rth level; then the candidate split positions are the block boundaries

$$T = \{c_1, \ldots, c_{L-1}\},$$

i.e. only between adjacent level blocks, excluding the final boundary that would empty the right side.

4 Per-feature Interval Statistics (Vector Representation)

Let $M = \sum_{j=1}^p K_j$ be the total number of intervals across all features of interest. We want to concatenate the interval information of all features (i.e. statistics n_{jk}, s_{jk}, q_{jk}) into a single long vector, arranging them sequentially within a unified one-dimensional index system $\{1, \cdots, M\}$.

To determine the position of the kth interval of the jth feature within this long vector, we first define the offsets of jth feature as the total number of intervals for the first $(j-1)$ features:

$$\mathbf{offsets}_j = \sum_{l=1}^{j-1} K_l, \quad \mathbf{offsets}_1 = 0.$$

Here the features are indexed simply in the original column order of the input data, with no reordering applied. Then the position of the kth interval of the jth feature is

$$m(j,k) = \mathbf{offsets}_j + k.$$

For each feature, per-interval total statistics are stored as constants within three vectors of length M.

$$\begin{aligned}\mathbf{tot\_n}_m &= n_{jk}\\ \mathbf{tot\_s}_m &= s_{jk}\\ \mathbf{tot\_q}_m &= q_{jk}.\end{aligned}$$

Create another three vectors of length M and initialize the right-side statistics to totals (all samples initially on the right):

$$\begin{aligned}\mathbf{r\_n}_m^{[0]} &= \mathbf{tot\_n}_m,\\ \mathbf{r\_s}_m^{[0]} &= \mathbf{tot\_s}_m,\\ \mathbf{r\_q}_m^{[0]} &= \mathbf{tot\_q}_m.\end{aligned}$$

The initial right-side per-feature risk is

$$\mathbf{R}_j^{[0]} = \sum_{m \in Range_j} \mathrm{risk}\Big(\mathbf{r\_n}_m^{[0]}, \mathbf{r\_s}_m^{[0]}, \mathbf{r\_q}_m^{[0]}\Big),$$

where $Range_j$ is determined via $[pos_j : pos_j + K_j - 1]$ with pos_1 being 1. Left-side stats are always complements:

$$\begin{aligned}\mathbf{l\_n}_m^{[0]} &= \mathbf{tot\_n}_m - \mathbf{r\_n}_m^{[0]},\\ \mathbf{l\_s}_m^{[0]} &= \mathbf{tot\_s}_m - \mathbf{r\_s}_m^{[0]},\\ \mathbf{l\_q}_m^{[0]} &= \mathbf{tot\_q}_m - \mathbf{r\_q}_m^{[0]}.\end{aligned}$$

Maintain two matrices: $\mathbf{DL}[i,j] = d_{ij}$ stores the per-sample ALE effect for feature j, and $\mathbf{IDX}[j,i] = \phi_j(i)$ stores the interval index to which sample i belongs for feature j. For the boundary variance stabilizer, precompute sufficient statistics along the order of samples according to current splitting feature:

$$\mathbf{S}[n,j] = \sum_{i=1}^{n} \mathbf{DL}[\mathrm{ord}(i), j], \qquad \mathbf{Q}[n,j] = \sum_{i=1}^{n} \mathbf{DL}^2[\mathrm{ord}(i), j].$$

5 One-pass Incremental Update

We now describe the main sweep that incrementally updates interval-wise risks while scanning possible split ranks $t = 1, \ldots, n_j - 1$ along the sorted order $\mathrm{ord}(1), \ldots, \mathrm{ord}(n_j)$. At each step, the sample $\mathrm{ord}(t)$ is moved from the right node to the left node, and the sufficient statistics of all features are updated accordingly.

5.1 Per-feature and Per-interval Update

Let $i^\star = \mathrm{ord}(t)$ be the sample index that moves from right to left at step t. For each feature $j \in \mathcal{S}$, define the interval index $k^\star = \phi_j(i^\star)$ and the effect value $d^\star = \mathbf{DL}[i^\star, j]$. We identify its global position in the concatenated vector as $m^\star = m(j, k^\star)$.

Right-side update. We decrease the right-side statistics of this interval by the contribution of sample $i^\star$:

$$\begin{aligned}
\mathbf{r__n}_{m^\star}^{[t]} &\leftarrow \mathbf{r__n}_{m^\star}^{[t-1]} - 1, \\
\mathbf{r__s}_{m^\star}^{[t]} &\leftarrow \mathbf{r__s}_{m^\star}^{t-1]} - d^\star, \\
\mathbf{r__q}_{m^\star}^{[t]} &\leftarrow \mathbf{r__q}_{m^\star}^{t-1]} - (d^\star)^2.
\end{aligned}$$

This reduces the number of samples and their cumulative sums in the corresponding interval on the right side.

Left-side update (implicit). Because left-side statistics are complements of totals,

$$\begin{aligned}
\mathbf{l__n}_{m^\star}^{[t]} &= \mathbf{tot__n}_{m^\star} - \mathbf{r__n}_{m^\star}^{[t]}, \\
\mathbf{l__s}_{m^\star}^{[t]} &= \mathbf{tot__s}_{m^\star} - \mathbf{r__s}_{m^\star}^{[t]}, \\
\mathbf{l__q}_{m^\star}^{[t]} &= \mathbf{tot__q}_{m^\star} - \mathbf{r__q}_{m^\star}^{[t]}.
\end{aligned}$$

No separate bookkeeping is needed; the left side automatically reflects the updated complement.

5.2 Incremental Risk Update

The risk contribution of feature j only changes in interval $k^\star$. Before the update, the right and left risks for that interval are:

$$\begin{aligned}
\mathbf{R}_{j,k^\star}^{[t-1]} &= \mathrm{risk}(\mathbf{r__n}_{m^\star}^{[t-1]}, \mathbf{r__s}_{m^\star}^{[t-1]}, \mathbf{r__q}_{m^\star}^{[t-1]}), \\
\mathbf{L}_{j,k^\star}^{[t-1]} &= \mathrm{risk}(\mathbf{l__n}_{m^\star}^{[t-1]}, \mathbf{l__s}_{m^\star}^{[t-1]}, \mathbf{l__q}_{m^\star}^{[t-1]}).
\end{aligned}$$

After the move, they become

$$\begin{aligned}
\mathbf{R}_{j,k^\star}^{[t]} &= \mathrm{risk}(\mathbf{r__n}_{m^\star}^{[t]}, \mathbf{r__s}_{m^\star}^{[t]}, \mathbf{r__q}_{m^\star}^{[t]}), \\
\mathbf{L}_{j,k^\star}^{[t]} &= \mathrm{risk}(\mathbf{l__n}_{m^\star}^{[t]}, \mathbf{l__s}_{m^\star}^{[t]}, \mathbf{l__q}_{m^\star}^{[t]}).
\end{aligned}$$

For boundary stabilizer, the per-feature risk is updated as

$$\begin{aligned}
\mathbf{R}_j^{[t]} &= \mathbf{R}_j^{[t-1]} - \mathbf{R}_{j,k^\star}^{[t-1]} + \mathbf{R}_{j,k^\star}^{[t]}, \\
\mathbf{L}_j^{[t]} &= \mathbf{L}_j^{[t-1]} - \mathbf{L}_{j,k^\star}^{[t-1]} + \mathbf{L}_{j,k^\star}^{[t]}.
\end{aligned}$$

The per-feature risk difference at step t is

$$\boldsymbol{\Delta}_j(t) = \left[\mathbf{L}_j^{[t]} - \mathbf{L}_j^{[t-1]}\right] + \left[\mathbf{R}_j^{[t]} - \mathbf{R}_j^{[t-1]}\right].$$

We accumulate the sum over all features:

$$\Delta(t) = \sum_{j=1}^{p} \boldsymbol{\Delta}_j(t).$$

The running total objective at rank t can therefore be updated in constant time:

$$Risk\_sum(t) = Risk\_sum(t-1) + \Delta(t),$$

where $Risk\_sum(t)$ denotes the global sum of risks across all features at rank t, and

$$Risk\_sum(0) = \sum_{j=1}^{p} \mathbf{R}_j^{[0]}.$$

Each step requires $O(p)$ operations (one interval update per feature), so the entire sweep is $O(pn_j)$ after sorting.

6 Boundary Stabilizer (Numeric Features Only)

For numeric split features, the order induced by $\mathbf{z}$ may yield unstable risk changes near boundaries due to large local variance. To mitigate this, we apply a local variance replacement for the current split feature j only.

6.1 Local window variance estimation

We describe the stabilizer for current numeric split feature j at a given candidate rank t, based on the sorted order $\mathrm{ord}(1), \ldots, \mathrm{ord}(n_j)$ induced by $\mathbf{z}$. For cumulative sums along the ordered samples, we have

$$\mathbf{S}[t, j] = \sum_{r=1}^{t} \mathbf{DL}[\mathrm{ord}(r), j],$$
$$\mathbf{Q}[t, j] = \sum_{r=1}^{t} \mathbf{DL}^2[\mathrm{ord}(r), j].$$

The left node at rank t contains the first t ordered samples, the right node contains the remaining $n_j - t$. We choose a base window size

$$w_\circ = \max(10, \lfloor 0.1 n_j \rfloor),$$

and side-specific window lengths

$$w_L = \min(w_\circ,\, t),$$
$$w_R = \min(w_\circ,\, n_j - t).$$

Left node. The *near* region is the most recent w_L samples moved into the left node:

$$s^L_{\mathrm{near}} = \mathbf{S}[t, j] - \mathbf{S}[t - w_L, j],$$
$$q^L_{\mathrm{near}} = \mathbf{Q}[t, j] - \mathbf{Q}[t - w_L, j].$$

The total of the left node is

$$s^L_{\mathrm{tot}} = \mathbf{S}[t, j], \qquad q^L_{\mathrm{tot}} = \mathbf{Q}[t, j].$$

This follows directly from the definition of the cumulative sums: since $\mathbf{S}[t, j]$ and $\mathbf{Q}[t, j]$ accumulate the first t ordered samples, they exactly represent the total effect and squared effect of all observations contained in the left node.

The *far* region is the remainder of the left node:

$$s^L_{\text{far}} = s^L_{\text{tot}} - s^L_{\text{near}},$$
$$q^L_{\text{far}} = q^L_{\text{tot}} - q^L_{\text{near}},$$

with size $n^L_{\text{far}} = t - w_L$.

Right node. Symmetrically, the *near* region is the first w_R samples remaining on the right:

$$s^R_{\text{near}} = (\mathbf{S}[t + w_R, j] - \mathbf{S}[t, j]),$$
$$q^R_{\text{near}} = (\mathbf{Q}[t + w_R, j] - \mathbf{Q}[t, j]),$$

and the total of the right node is

$$s^R_{\text{tot}} = \mathbf{S}[n, j] - \mathbf{S}[t, j],$$
$$q^R_{\text{tot}} = \mathbf{Q}[n, j] - \mathbf{Q}[t, j].$$

The *far* region of the right node excludes the *near* region:

$$s^R_{\text{far}} = s^R_{\text{tot}} - s^R_{\text{near}},$$
$$q^R_{\text{far}} = q^R_{\text{tot}} - q^R_{\text{near}},$$

with size $n^R_{\text{far}} = (n_j - t) - w_R$.

Variance estimates. For each side $S \in \{L, R\}$, define unbiased variance estimates (when corresponding sizes ≥ 2):

$$\sigma^2_{S,\text{near}} = \frac{q^S_{\text{near}} - (s^S_{\text{near}})^2/w_S}{\max(w_S - 1, 1)},$$
$$\sigma^2_{S,\text{far}} = \frac{q^S_{\text{far}} - (s^S_{\text{far}})^2/n^S_{\text{far}}}{\max(n^S_{\text{far}} - 1, 1)}.$$

If $n^S_{\text{far}} \leq 1$, the far variance is undefined and no stabilization is applied on that side.

6.2 Replacement criterion

For the split feature j and each side $S \in \{L, R\}$, the stabilizer tests whether the local near window exhibits excessive variance relative to its far region: if

$$\sigma^2_{S,\text{far}} > 0 \quad \text{and} \quad \sigma^2_{S,\text{near}} \geq 4\,\sigma^2_{S,\text{far}},$$

then the near-region risk term of this side is replaced by a stabilized version based on the far-region mean and variance. Let

$$\mu_{S,\text{far}} = \frac{s^S_{\text{far}}}{n^S_{\text{far}}}.$$

The original near-region risk contribution is

$$\text{risk}(w_S,\, s^S_{\text{near}},\, q^S_{\text{near}}).$$

We replace it by the risk implied if the near window were drawn from the far-region distribution:

$$\mathrm{risk}\Big(w_S,\, w_S\,\mu_{S,\mathrm{far}},\, w_S\Big(\sigma^2_{S,\mathrm{far}} + \mu^2_{S,\mathrm{far}}\Big)\Big) = w_S\,\sigma^2_{S,\mathrm{far}}.$$

Therefore, for current numeric split feature j at rank t:

$$\begin{aligned}
\mathbf{R}^{[t]}_{j,stab} &\leftarrow \mathbf{R}^{[t]}_j - \mathrm{risk}(w_R,\, s^R_{\mathrm{near}},\, q^R_{\mathrm{near}}) + w_R\,\sigma^2_{R,\mathrm{far}},\ \text{or}\\
\mathbf{L}^{[t]}_{j,stab} &\leftarrow \mathbf{L}^{[t]}_j - \mathrm{risk}(w_L,\, s^L_{\mathrm{near}},\, q^L_{\mathrm{near}}) + w_L\,\sigma^2_{L,\mathrm{far}}.
\end{aligned}$$

If the condition fails (e.g. $\sigma^2_{S,\mathrm{far}} \le 0$ or insufficient far samples), no modification is applied. This ensures symmetric boundary stabilization on both sides while preserving global risk continuity.

7 Objective and Split Selection

The global objective at rank t is the total risk across all features after the left/right partition induced by the current split feature j:

$$\mathcal{R}^{[t]} = \sum_{j\in\mathcal{S}} \left(\mathbf{L}^{[t]}_j + \mathbf{R}^{[t]}_j\right) = Risk\_sum(t).$$

For numeric split features, $\mathbf{L}^{[t]}_j$ and $\mathbf{R}^{[t]}_j$ are adjusted by the stabilizer (Section 5)

$$\mathcal{R}^{[t]} = Risk\_sum(t) - \mathbf{L}^{[t]}_j - \mathbf{R}^{[t]}_j + \mathbf{L}^{[t]}_{j,stab} + \mathbf{R}^{[t]}_{j,stab}$$

For categorical split features, no stabilization is applied. We evaluate only valid candidate ranks $t \in T$ satisfying the node size constraint

$$t \ge \texttt{min.node.size}, \quad n_j - t \ge \texttt{min.node.size}.$$

We search for the rank $t^\star$ that minimizes the global objective:

$$t^\star = \arg\min_{t\in T}\ \mathcal{R}^{[t]}.$$

The corresponding split point is then defined as

$$z^\star = \begin{cases} \dfrac{\max(z_{(1:t^\star)}) + \min(z_{(t^\star+1:n_j)})}{2}, & \text{if } \mathbf{z} \text{ is numeric,}\\[2ex] \text{level separating the first } r^\star \text{ ordered levels,} & \text{if } \mathbf{z} \text{ is categorical,}\end{cases}$$

where $r^\star$ satisfies the cumulative count condition

$$c_{r^\star} = \sum_{\ell=1}^{r^\star} n_\ell = t^\star,$$

with n_ℓ denoting the number of samples in level ℓ after the CART-style ordering. Finally, the per-feature contributions at the best split are stored as

$$\begin{aligned}
\mathbf{Objective.value}_j &= \mathbf{L}^{[t^\star]}_j + \mathbf{R}^{[t^\star]}_j,\\
\mathbf{Left.value}_j &= \mathbf{L}^{[t^\star]}_j,\\
\mathbf{Right.value}_j &= \mathbf{R}^{[t^\star]}_j.
\end{aligned}$$

8 Algorithm

The following pseudocode summarizes the entire best-split search procedure for one split feature j.

Algorithm 1 Best Split Selection for Feature j (`search_best_split_point_d`)

Input: feature values $\mathbf{z}$, ALE effects `effect`, feature index j, minimum node size m_{min}, quantile control `n.quantiles`
Output: best split point $z^\star$, objective $\mathcal{R}^{[t^\star]}$, per-feature risk vectors
Build candidate order and positions (`build_order_and_candidates`)
Build per-feature interval statistics vectors (`build_stats`)
Initialize right risks $\mathbf{R}_j^{[0]}$ and set $Risk\_sum(0) = \sum_j \mathbf{R}_j^{[0]}$
for $t = 1$ **to** $n_j - 1$ **do**
    Move sample ord(t) from right to left and update all intervals (`move_row_all`)
    Update $\mathbf{L}_j^{[t]}, \mathbf{R}_j^{[t]}$ and $Risk\_sum(t) = Risk\_sum(t-1) + \Delta(t)$
    if $t \in T$ **and** $t \geq m_{min}$ **and** $(n_j - t) \geq m_{min}$ **then**
        if $\mathbf{z}$ numeric **then**
            Apply boundary stabilizer to feature j
        end if
        $\mathcal{R}^{[t]} = \sum_j (\mathbf{L}_j^{[t]} + \mathbf{R}_j^{[t]})$
        if $\mathcal{R}^{[t]} < \mathcal{R}^{[t^*]}$ **then**
            Store $t^\star \leftarrow t$, $\mathcal{R}^{[t^*]} \leftarrow \mathcal{R}^{[t]}$, record per-feature vectors
        end if
    end if
end for
Determine $z^\star$ from $t^\star$ (numeric midpoint or categorical boundary)
return $\{z^\star, \mathcal{R}^{[t^*]}, \textbf{Objective.value}, \textbf{Left.value}, \textbf{Right.value}\}$

9 Complexity Analysis

Let n_j denote the number of valid (non-missing) observations in feature j, and $p = |\mathcal{S}|$ the number of features of interest.

Per-step complexity. Each iteration over rank t updates interval statistics and risks for all features. Since each feature has at most one affected interval, the update is $O(p)$ per step.

Total cost per feature. The one-pass sweep over all ranks is thus $O(pn_j)$ after sorting. Including sorting of $\mathbf{z}$ adds $O(n_j \log n_j)$, yielding:

$$O(n_j \log n_j + pn_j).$$

If the number of candidate ranks $|T|$ is reduced by quantile subsampling, the effective runtime scales with $|T|$ instead of n_j.

Overall complexity. For all split features $j = 1, \ldots, p$ considered in `search_best_split_d`, the total runtime is

$$O\Big(\sum_{j=1}^{p}(n_j \log n_j + p n_j)\Big),$$

which in the common case of balanced $n_j \approx n$ simplifies to $O(p(n \log n + pn))$.