GRADIENT BOOSTING WITH LEARNABLE AND INCREMENTALLY REFINED TREES

Anonymous authors
Paper under double-blind review

ABSTRACT

Gradient boosting builds an ensemble by sequentially adding weak learners while keeping previously learned learners fixed. We revisit this design choice and ask whether boosting can benefit from jointly refining previously added learners as new learners are introduced. We investigate this question for gradient-boosted decision trees by replacing discontinuous split thresholds with differentiable routing functions, treating both tree decision boundaries and leaf values as continuously learnable parameters. Based on this formulation, we propose the Co-adaptive Gradient Boosting (CGB) framework and three algorithmic variants: CGB-AP, CGB-OB, and CGB-RO. All three methods perform gradient-based updates to the parameters of previously added trees. CGB-AP restricts boundary updates to axis-parallel directions, while CGB-OB permits oblique boundary updates. In both methods, newly added trees are initialized using standard purity-based greedy construction. CGB-RO instead bypasses greedy initialization entirely, starting each new tree randomly as an oblique decision tree and performing a localized gradient-based update before incorporating it into the ensemble. Across standard tabular benchmark datasets, we find that continuously updating previously added trees consistently matches or improves upon conventional discrete gradient boosting. These results suggest that freezing earlier weak learners is not a strict necessity of the boosting framework, and that incremental, differentiable refinement of the ensemble provides a promising alternative to the standard stagewise optimization paradigm.

1 INTRODUCTION

Tabular data remains a prevalent modality across machine learning applications; for predictive modeling on these datasets, Gradient Boosted Decision Trees (GBDTs) are widely utilized due to their robust performance (Chen & Guestrin, 2016; Ke et al., 2017; Prokhorenkova et al., 2018). The effectiveness of GBDTs stems from their partition-based routing, which naturally handles heterogeneous features and scale variances without requiring the extensive preprocessing typically needed for deep neural networks (Borisov et al., 2022; Grinsztajn et al., 2022; Shwartz-Ziv & Armon, 2022). GBDTs construct models through stagewise additive expansions (Friedman, 2001; Mason et al., 1999); at each boosting iteration, a new weak learner is fitted to the negative gradient, or pseudo-residuals, of the preceding ensemble.

However, this stagewise additive process is inherently greedy. Once a tree is fitted and incorporated into the ensemble, its internal parameters: split features, thresholds, and leaf values, are permanently fixed. Because earlier trees are locked, the ensemble must rely entirely on the addition of new weak learners to correct residual errors, rather than refining existing decision boundaries. In contrast, deep learning architectures utilize backpropagation to simultaneously update all parameters, allowing earlier layers to continuously adapt to the gradients flowing from the final objective.

To address the limitations of greedy stagewise learning, we introduce the Co-adaptive Gradient Boosting (CGB) framework in which the trees added in the past rounds are also updated incrementally. Building on prior research that replaces discrete node splits with smooth, probabilistic routing (Irsoy et al., 2012; Kontschieder et al., 2015) and integrates differentiable trees into ensemble architectures (Popov et al., 2020), our approach utilizes continuous routing functions and backpropagation to relax the hard constraints of traditional boosting. As new weak learners are incrementally

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

added, the parameters of historical trees are globally co-adapted. This allows the model to retain the structural properties of decision tree ensembles while leveraging the end-to-end global optimization capabilities of neural networks.

Our main contributions are as follows:

- **Co-adaptive Gradient Boosting (CGB):** We propose a fully differentiable boosting framework that relaxes the greedy stagewise constraint, allowing historical decision trees (i.e., decision tress of the past rounds) to be updated via backpropagation as new trees are added to the ensemble.
- **Initialization and Routing Formulations:** We introduce and evaluate three novel algorithmic variants: CGB-AP (XGBoost initialization with axis-parallel updates), CGB-OB (XGBoost initialization with oblique updates), and CGB-RO (purely random oblique initialization). This demonstrates the framework's flexibility to optimize both pre-existing greedy boundaries as well as randomly initialized boundaries.
- **Windowed Co-adaptation:** We introduce a sliding-window update mechanism that restricts gradient-based refinement to a subset of the most recently added trees. This approach balances historical co-adaptation with training stability, mitigating forgetting of the original residual targets.
- **Empirical Validation:** We benchmark our variants against standard XGBoost across diverse tabular datasets, demonstrating that the co-adaptive relaxation of stagewise locking yields competitive predictive performance.

2 Related Work

Our framework intersects two major domains of ensemble learning: gradient-boosted decision trees and differentiable tree routing. We review the recent advancements in these areas and the specific structural constraints our work addresses.

2.1 Gradient Boosted Decision Trees and Greedy Optimization

For tabular data, Gradient Boosted Decision Trees (GBDTs) such as XGBoost (Chen & Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018) demonstrate competitive performance. These models construct strong learners by sequentially adding weak decision trees to fit the pseudo-residuals of a loss objective (Friedman, 2001). However, the standard GBDT framework relies on a greedy, stagewise growth mechanism: once a tree is built and added to the ensemble, its internal parameters are fixed. Methods such as Tree Alternating Optimization (TAO) (Carreira-Perpinán & Tavallali, 2018) and post-creation backfitting (Chipman et al., 2010) attempt to mitigate this by iteratively refining subsets of nodes or trees, but they still operate within discrete optimization spaces. The inability to continuously and globally co-adapt historical trees to newly discovered feature interactions remains a structural constraint of traditional boosting.

2.2 Differentiable Routing in Tree Ensembles

To relax the constraints of discrete tree building, prior research has focused on making decision trees differentiable. Architectures such as Soft Decision Trees (Frosst & Hinton, 2017; Irsoy et al., 2012) and Deep Neural Decision Forests (Kontschieder et al., 2015) replace hard binary splits with smooth, probabilistic activation functions, allowing gradients to flow through the routing nodes. More recently, frameworks like Neural Oblivious Decision Ensembles (NODE) (Popov et al., 2020) and differentiable tree layers (Hazimeh et al., 2020) have unified differentiable routing with ensemble learning to optimize tree structures collectively. However, contemporary differentiable tree models generally forgo the sequential growth mechanism of boosting, relying instead on the joint optimization of predefined, fixed-size architectures (Arik & Pfister, 2021). Our Co-adaptive Gradient Boosting (CGB) framework reconciles these paradigms. It preserves the stagewise additive expansion of traditional GBDTs, yet employs differentiable splits to relax the greedy optimization constraint. This formulation enables the continuous, global co-adaptation of historical weak learners as the ensemble grows.

3 BACKGROUND

Boosting. Consider a standard supervised learning setup where we are given a training dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$. Here, $\mathbf{x}_i \in \mathbb{R}^d$ represents a d-dimensional feature vector, and $y_i \in \mathcal{Y} = \{0, 1\}$ is the corresponding target label for binary classification. At its core, boosting operates by sequentially combining an ensemble of base models into a single, strong predictive model.

Weak Learners. The base models, commonly referred to as "weak learners," are the fundamental building blocks of the ensemble. The boosting framework is flexible, allowing various types of weak learners to map the input space to the target residuals.

Parameters of Weak Learners. The mathematical formulation of a chosen weak learner $f(\mathbf{x}; \theta)$ dictates the nature of its trainable parameter space θ. The optimization and expressive capacity of the ensemble depend on this θ.

Examples of Weak Learner Parameters. Depending on the architecture, the trainable parameters of these weak learners are different:

- **Linear Models:** If the algorithm utilizes a linear weak learner, expressed as $f(\mathbf{x}; \theta) = \mathbf{w}^\top \mathbf{x} + b$, then the trainable parameters are the weight vector and bias: $\theta = \{\mathbf{w}, b\}$.
- **Decision Trees:** Gradient Boosted Decision Trees (GBDTs) characteristically employ discrete trees as their weak learners, which can be expressed as $f(\mathbf{x}; \theta) = \sum_{j=1}^J v_j \mathbf{1}(\mathbf{x} \in R_j)$. The topology of these trees is determined by a hierarchical collection of axis-parallel splitting hyperplanes. Each hyperplane routes data based on a discrete scalar threshold. Let $\mathbf{b}$ represent this set of biases, and let $\mathbf{v} \in \mathbb{R}^J$ represent the scalar weights assigned to the disjoint regions R_j (leaves). We can therefore define the trainable parameters of the decision tree as $\theta = \{\mathbf{b}, \mathbf{v}\}$. Due to the combinatorial nature of the discrete splitting biases $\mathbf{b}$, they are constructed using a greedy, impurity-based recursive partitioning algorithm.

Gradient Boosting. Classical boosting algorithms, such as AdaBoost (Freund & Schapire, 1997), adapt to errors by iteratively shifting the weights of misclassified training samples. Gradient boosting takes a different approach. It frames this sequential growth as a functional optimization problem. Instead of manipulating sample weights, it focuses on minimizing a differentiable loss function. To achieve this, it trains consecutive weak learners to fit the negative gradients of that loss. These negative gradients, evaluated against the current ensemble's predictions, are commonly referred to as pseudo-residuals.

The Stagewise Update Step in Gradient Boosting. GBDTs progressively construct their predictive model $F_T(\mathbf{x})$ through stagewise additive modeling. At any given iteration t, the ensemble evolves according to:

$$F_t(\mathbf{x}) = F_{t-1}(\mathbf{x}) + \alpha f_t(\mathbf{x}; \theta_t) \tag{1}$$

where $f_t \in \mathcal{F}$ is the decision tree introduced at step t, $\mathcal{F}$ represents the space of all possible decision trees, and $\alpha \in (0, 1]$ acts as a shrinkage parameter (learning rate).

Learning the Parameters of Weak Learners. To learn the optimal values of parameters θ_t for candidate tree f_t, standard frameworks seek to minimize a regularized objective function. First, the pseudo-residuals for each training sample is computed. These are defined as the negative gradients of a differentiable convex loss function ℓ (such as Binary Cross Entropy) evaluated against the current predictions of the ensemble:

$$r_{i,t} = -\left[\frac{\partial \ell(y_i, F)}{\partial F}\right]_{F=F_{t-1}(\mathbf{x}_i)} \tag{2}$$

Next, using an impurity-based greedy splitting criterion, the new tree f_t is grown to minimize the squared error against these target residuals. This objective is regularized by a structural complexity penalty Ω:

$$\tilde{\mathcal{L}}^{(t)}(f_t) = \sum_{i=1}^N \left(r_{i,t} - f_t(\mathbf{x}_i)\right)^2 + \Omega(f_t) \tag{3}$$

For completeness, the standard residual-fitting procedure for GBDTs is detailed in Appendix C.

The Greedy Lock. Here we make an observation: once the parameters θ_t of f_t are evaluated and integrated into the ensemble, they are permanently frozen in place. Under traditional stagewise optimization, the parameters defining all previously added trees $\{\theta_1, \ldots, \theta_{t-1}\}$ are treated as immutable during the minimization of $\tilde{\mathcal{L}}^{(t)}(f_t)$. It prevents the ensemble from retroactively adjusting its historical boundaries if necessary.

4 METHODOLOGY: CO-ADAPTIVE GRADIENT BOOSTING

The greedy lock inherent to traditional stagewise optimization invites a natural question:

What if the parameters of the classifier are updated during every boosting round?

To explore this idea and allow older trees to adjust their parameters, we must first abandon the discrete, non-differentiable nature of standard decision trees. By reformulating the weak learners as differentiable trees, we can successfully unify additive stagewise growth with global gradient descent.

4.1 DIFFERENTIABLE TREE FORMULATION

We formulate our decision tree weak learners as follows. Let $\boldsymbol{\theta}_t = \{\mathbf{w}_n, b_n\}_{n\in\mathcal{N}} \cup \{v_j\}_{j\in\mathcal{L}}$ denote the parameter set for the t-th tree f_t, where $\mathcal{N}$ represents the internal branch nodes and $\mathcal{L}$ represents the terminal leaf nodes.

For a given input vector $\mathbf{x} \in \mathbb{R}^d$, the probability of routing a sample to the right child of an internal node $n \in \mathcal{N}$ is computed using the function:

$$p_n(\mathbf{x}; \boldsymbol{\theta}_t) = \sigma\left(\tau(\mathbf{w}_n^\top \mathbf{x} - b_n)\right) \tag{4}$$

where $\mathbf{w}_n \in \mathbb{R}^d$ is the feature selector weight vector, $b_n \in \mathbb{R}$ is the split threshold, $\sigma(\cdot)$ is the sigmoid activation function, and $\tau > 0$ is a temperature parameter controlling the steepness of the routing boundary. For axis-parallel trees, $\mathbf{w}_n$ is constrained to be a one-hot vector, whereas for oblique trees, $\mathbf{w}_n$ is a dense continuous vector. The probability of routing to the left child is $1 - p_n(\mathbf{x}; \boldsymbol{\theta}_t)$.

The activation probability of a specific leaf node $j \in \mathcal{L}$, denoted as $\mu_j(\mathbf{x}; \boldsymbol{\theta}_t)$, is calculated as the product of the routing probabilities along the unique path from the root down to that leaf. The final output of the tree is the expected value over all leaves:

$$f_t(\mathbf{x}; \boldsymbol{\theta}_t) = \sum_{j\in\mathcal{L}} v_j \mu_j(\mathbf{x}; \boldsymbol{\theta}_t) \tag{5}$$

where $v_j \in \mathbb{R}$ represents the weight assigned to the leaf j.

4.2 THE TWO-STAGE CO-ADAPTATION STRATEGY

With weak learners reformulated as differentiable functions, we introduce the Co-adaptive Gradient Boosting (CGB) framework. CGB breaks each boosting iteration's optimization into two distinct stages. Rather than appending a new learner to a frozen ensemble, it first updates all previously added trees via backpropagation. It then recalculates the target pseudo-residuals from this refined ensemble and fits the new weak learner to them. We term this process of refining historical boundaries before adding new learners **co-adaptation**, a mechanism that enables continuous ensemble optimization while preserving stagewise additive growth.

Let $F_t(\mathbf{x}; \Theta_t)$ denote the state of the ensemble at boosting step t, defined as:

$$F_t(\mathbf{x}; \Theta_t) = \sum_{k=1}^{t} \alpha f_k(\mathbf{x}; \boldsymbol{\theta}_k) \tag{6}$$

where $\Theta_t = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_t\}$ represents the combined parameter set of all trees up to step t. While standard GBDTs permanently freeze Θ_{t-1} during step t, our two-stage framework optimizes these parameters as follows:

Stage 1: Global Co-Adaptation
Before initializing the new weak learner, we globally co-adapt the parameters of the existing historical ensemble, Θ_{t-1}. We learn the updated parameter set Θ^*_{t-1} by performing gradient steps using the Adam optimizer with a learning rate η to minimize a global objective function:

$$\Theta^*_{t-1} \leftarrow \Theta_{t-1} - \eta \nabla_{\Theta_{t-1}} \mathcal{L}(\Theta_{t-1}) \tag{7}$$

where $\mathcal{L}(\Theta_{t-1})$ denotes the loss evaluated over the training data, defined as:

$$\mathcal{L}(\Theta_{t-1}) = \frac{1}{N} \sum_{i=1}^{N} \ell \left(y_i, F_{t-1}(\mathbf{x}_i; \Theta_{t-1}) \right) \tag{8}$$

Here, ℓ represents a differentiable convex loss function, such as Binary Cross Entropy for binary classification tasks.

This stage breaks the greedy lock, allowing older trees to shift their routing boundaries, thresholds, and leaf values in response to the overall ensemble error. To reduce computational overhead as the ensemble grows, this update can be windowed to target only the most recent h trees, updating $\{\boldsymbol{\theta}_{t-h}, \ldots, \boldsymbol{\theta}_{t-1}\}$ while freezing the older learners.

Stage 2: Local Initialization
Following global co-adaptation, we temporarily freeze the updated ensemble Θ^*_{t-1} and compute the pseudo-residuals for each training sample based on the standard gradient boosting formulation:

$$r_{i,t} = - \left[\frac{\partial \ell(y_i, F)}{\partial F} \right]_{F = F_{t-1}(\mathbf{x}_i; \Theta^*_{t-1})} \tag{9}$$

The t-th tree is initialized by fitting its parameters $\boldsymbol{\theta}_t$ to these pseudo-residuals via a Mean Squared Error (MSE) objective via gradient descent:

$$\boldsymbol{\theta}_t^{(k+1)} \leftarrow \boldsymbol{\theta}_t^{(k)} - \eta_{\text{local}} \nabla_{\boldsymbol{\theta}_t} \left[\frac{1}{N} \sum_{i=1}^{N} \left(f_t(\mathbf{x}_i; \boldsymbol{\theta}_t^{(k)}) - r_{i,t} \right)^2 \right] \tag{10}$$

Once this local optimization is complete, the newly fitted tree is scaled by the shrinkage parameter α and appended to the ensemble, forming $F_t(\mathbf{x}; \Theta_t)$ where $\Theta_t = \Theta^*_{t-1} \cup \{\boldsymbol{\theta}_t\}$.

4.3 Initialization Regimes and Novel Algorithms

We will now describe the three novel algorithms, namely CGB-OB, CGB-AP and CGB-RO.

- **CGB-AP [Stage 1: $\mathbf{w}_n$ Frozen | Stage 2: XGBoost Init]:** The new weak learner (Stage 2) is added by extracting exact splits and leaf values from a standard XGBoost tree fitted to the residuals. Note that for this tree $\mathbf{w}_n$s are one-hot features which are frozen while optimization. This maintains strict axis-parallel boundaries, while the thresholds $\boldsymbol{b}_n$ and leaves v_j are globally co-adapted via Stage 1 in the next rounds (see Algorithm 1).
- **CGB-OB [Stage 1: All Parameters Unfrozen | Stage 2: XGBoost Init]:** The new weak learner (Stage 2) is selected in the same manner as CGB-AP. While the $\mathbf{w}_n$s are one-hot in the round when the tree is added first, in subsequent rounds $\mathbf{w}_n$s are not frozen. This allows oblique updates, and hence the boundaries in the subsequent rounds are not necessarily axis-parallel (see Algorithm 2).
- **CGB-RO [Stage 1: All Parameters Unfrozen | Stage 2: Random Init]:** The new weak learner in Stage 2 is initialized using random weights (i.e., we do not use the standard XGBoost procedure to obtain axis-parallel trees). This tree with random weights is then fitted to the residuals via gradient descent (Stage 2) before being added to the ensemble. This is then followed by the global co-adaptation (Stage 1) in the next rounds (see Algorithm 3).

5 Experiments

To evaluate the efficacy of the Co-adaptive Gradient Boosting (CGB) framework, we benchmark our proposed variants against a standard XGBoost baseline. Our experiments are designed to answer two

Algorithm 1 CGB-AP: Axis-Parallel Co-adaptive Gradient Boosting

Require: Dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, iterations T, shrinkage α, global LR η, window size h

Initialize ensemble $F_0(\mathbf{x}) = \log\left(\frac{p}{1-p}\right)$ where $p = \frac{1}{N}\sum_{i=1}^N y_i$, parameter set $\Theta_0 = \emptyset$
for $t = 1$ **to** T **do**
  # Stage 1: Global Co-Adaptation (Thresholds & Leaves Only)
  if $t > 1$ **then**
    Define active window: $\Theta_{\text{active}} \leftarrow \{\boldsymbol{\theta}_k \in \Theta_{t-1} \mid k \geq t - h\}$
    (Freeze feature selectors $\mathbf{w}_n$ to maintain axis-parallel constraints)
    Update $\{b_n, v_j\} \in \Theta_{\text{active}}$ via Adam: minimize $\sum_{i=1}^N \ell(y_i, F_{t-1}(\mathbf{x}_i; \Theta_{t-1}))$
    Update ensemble state to $F_{t-1}(\mathbf{x}; \Theta^*_{t-1})$
  end if
  # Stage 2: Local Initialization (XGB Warm-Start)
  Compute pseudo-residuals $r_{i,t} = -\left[\frac{\partial \ell}{\partial F}\right]_{F=F_{t-1}(\mathbf{x}_i;\Theta^*_{t-1})}$
  Initialize $\boldsymbol{\theta}_t$ by extracting exact splits/leaves from an XGBoost tree fitted to $r_{i,t}$
  # Ensemble Update
  $\Theta_t \leftarrow \Theta^*_{t-1} \cup \{\boldsymbol{\theta}_t\}$
  $F_t(\mathbf{x}; \Theta_t) \leftarrow F_{t-1}(\mathbf{x}; \Theta^*_{t-1}) + \alpha f_t(\mathbf{x}; \boldsymbol{\theta}_t)$
end for
return Fully co-adapted ensemble $F_T(\mathbf{x}; \Theta_T)$

Algorithm 2 CGB-OB: Oblique Co-adaptive Gradient Boosting (XGB Init)

Require: Dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, iterations T, shrinkage α, global LR η, window size h

Initialize ensemble $F_0(\mathbf{x}) = \log\left(\frac{p}{1-p}\right)$ where $p = \frac{1}{N}\sum_{i=1}^N y_i$, parameter set $\Theta_0 = \emptyset$
for $t = 1$ **to** T **do**
  # Stage 1: Global Co-Adaptation (All Parameters Unfrozen)
  if $t > 1$ **then**
    Define active window: $\Theta_{\text{active}} \leftarrow \{\boldsymbol{\theta}_k \in \Theta_{t-1} \mid k \geq t - h\}$
    Update $\{w_n, b_n, v_j\} \in \Theta_{\text{active}}$ via Adam: minimize $\sum_{i=1}^N \ell(y_i, F_{t-1}(\mathbf{x}_i; \Theta_{t-1}))$
    Update ensemble state to $F_{t-1}(\mathbf{x}; \Theta^*_{t-1})$
  end if
  # Stage 2: Local Initialization (XGB Warm-Start)
  Compute pseudo-residuals $r_{i,t} = -\left[\frac{\partial \ell}{\partial F}\right]_{F=F_{t-1}(\mathbf{x}_i;\Theta^*_{t-1})}$
  Initialize $\boldsymbol{\theta}_t$ by extracting exact splits/leaves from an XGBoost tree fitted to $r_{i,t}$
  # Ensemble Update
  $\Theta_t \leftarrow \Theta^*_{t-1} \cup \{\boldsymbol{\theta}_t\}$
  $F_t(\mathbf{x}; \Theta_t) \leftarrow F_{t-1}(\mathbf{x}; \Theta^*_{t-1}) + \alpha f_t(\mathbf{x}; \boldsymbol{\theta}_t)$
end for
return Fully co-adapted ensemble $F_T(\mathbf{x}; \Theta_T)$

primary questions: (1) Does breaking the greedy stagewise lock and enabling global co-adaptation improve predictive performance? (2) How do different initialization strategies (greedy priors vs. random initialization) and update scopes (full ensemble vs. windowed) impact the final model accuracy?

5.1 Experimental Setup

Datasets: We evaluate our models across 24 publicly available tabular datasets from the UCI Machine Learning Repository (Dua & Graff, 2019) and OpenML (Vanschoren et al., 2014), encompassing a diverse range of dimensions, sample sizes, and feature distributions. All datasets are formulated as binary classification tasks.

Evaluation Protocol: We use a stratified 80-20 train-test split across three random seeds. Mean test accuracies are reported in the main text, with complete results with standard deviations detailed in Appendix B.

Algorithm 3 CGB-RO: Oblique Co-adaptive Gradient Boosting (Random Init)

Require: Dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, iterations T, shrinkage α, global LR η, window size h

Initialize ensemble $F_0(\mathbf{x}) = \log\left(\frac{p}{1-p}\right)$ where $p = \frac{1}{N}\sum_{i=1}^N y_i$, parameter set $\Theta_0 = \emptyset$
for $t = 1$ **to** T **do**
  # Stage 1: Global Co-Adaptation (All Parameters Unfrozen)
  if $t > 1$ **then**
    Define active window: $\Theta_{\text{active}} \leftarrow \{\boldsymbol{\theta}_k \in \Theta_{t-1} \mid k \geq t - h\}$
    Update $\{w_n, b_n, v_j\} \in \Theta_{\text{active}}$ via Adam: minimize $\sum_{i=1}^N \ell(y_i, F_{t-1}(\mathbf{x}_i; \Theta_{t-1}))$
    Update ensemble state to $F_{t-1}(\mathbf{x}; \Theta^*_{t-1})$
  end if
  # Stage 2: Local Initialization (Random & Gradient Descent)
  Compute pseudo-residuals $r_{i,t} = -\left[\frac{\partial \ell}{\partial F}\right]_{F=F_{t-1}(\mathbf{x}_i;\Theta^*_{t-1})}$
  Initialize $\boldsymbol{\theta}_t$ using random weights
  Fit $\boldsymbol{\theta}_t$ to $r_{i,t}$ via gradient descent: $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_t - \eta_{\text{local}} \nabla_{\boldsymbol{\theta}_t} \left[\frac{1}{N}\sum_{i=1}^N \left(f_t(\mathbf{x}_i; \boldsymbol{\theta}_t) - r_{i,t}\right)^2\right]$
  # Ensemble Update
  $\Theta_t \leftarrow \Theta^*_{t-1} \cup \{\boldsymbol{\theta}_t\}$
  $F_t(\mathbf{x}; \Theta_t) \leftarrow F_{t-1}(\mathbf{x}; \Theta^*_{t-1}) + \alpha f_t(\mathbf{x}; \boldsymbol{\theta}_t)$
end for
return Fully co-adapted ensemble $F_T(\mathbf{x}; \Theta_T)$

5.2 Implementation Details

As the **CGB-RO** configurations lack the structural priors of greedy tree building, optimizing them from a random initialization is not trivial. To guide the trees toward a better optimum we constrain the learning process. First, we apply an L^1 regularization penalty to the feature selector weights during both the local and global stages. Second, CGB-RO employs a dynamic temperature annealing schedule to ensure decision boundaries transition from soft, exploratory linear combinations early in training to crisp, discrete regions by convergence. The ablation baseline, **CGB-RO (No Global)**, correspondingly retains the local L^1 penalty and the local temperature annealing during its local fitting stage.

For detailed hyperparameter configurations see Appendix A.2.

5.3 Results and Discussion

Table 1 summarizes mean predictive performance and aggregate win/tie/loss records across 24 datasets (the maximum standard deviation is 0.0769; full table with variances are in Appendix B). Overall, these results demonstrate that relaxing boosting's greedy constraint yields competitive, frequently superior accuracy.

Aggregate Performance and Consistency. While no single co-adaptive configuration dominates across every individual dataset, the aggregate metrics reveal a highly consistent trend: every proposed CGB variant achieves a higher average accuracy than the standard baseline. The CGB-AP ($h = 3$) configuration demonstrates remarkable stability, outperforming XGBoost on 23 of 24 datasets. On average, the best overall variant is CGB-OB ($h = 3$), which achieves an average accuracy gain of $+2.51\%$ over the XGBoost baseline across the 24 evaluated datasets.

The Efficacy of XGBoost Initialization and Windowing. Our empirical results highlight **CGB-OB** ($h = 3$) as a robust configuration. By utilizing exact XGBoost splits for initialization, the model gets a head start. Unfreezing these boundaries and applying oblique co-adaptation over a localized sliding window ($h = 3$) yields the highest performance. The localized window acts as a stability guardrail; it allows the ensemble to rotate the most recent decision boundaries to capture residual variance without causing the earliest trees to forget their original routing logic.

Viability of Random Initialization. CGB-RO proves greedy initialization is not strictly necessary. Despite starting with random Gaussian weights, local optimization and global co-adaptation allow

it to outperform XGBoost on 22 of 24 datasets, confirming differentiable ensembles can organically learn effective topologies given proper regularization and temperature scaling.

Table 1: Mean test accuracy across 3 seeds (best in bold; full table with variances in Appendix B). Bottom rows show aggregate Win/Tie/Loss counts against XGBoost.

	Baseline	**CGB-AP**		**CGB-OB**		**CGB-RO**	
Dataset	XGBoost	Full	$h = 3$	Full	$h = 3$	Full	$h = 3$
Adult	0.8488	**0.8562**	0.8537	0.8398	0.8463	0.8503	0.8470
Bank Marketing	0.8987	0.9065	0.9047	0.9008	0.9046	**0.9077**	0.9033
Blood Transfusion	0.7800	0.7933	0.7911	0.7778	0.7889	**0.8067**	**0.8067**
Breast Cancer	0.9524	0.9571	0.9571	0.9619	**0.9667**	**0.9667**	**0.9667**
Chess	0.9401	0.9750	0.9651	0.9839	**0.9859**	0.9839	0.9792
Ecoli	0.9412	0.9314	0.9265	0.9412	0.9412	**0.9461**	**0.9461**
Electricity	0.7836	0.8007	0.7937	**0.8444**	0.8361	0.8079	0.7836
German	0.7383	0.7350	0.7550	0.7417	0.7600	**0.7617**	0.7583
Gisette	0.9506	0.9575	0.9653	0.9750	0.9794	0.9800	**0.9828**
Glass	0.8450	0.8450	**0.8605**	0.8372	0.8450	0.8295	0.8295
Higgs	0.6879	0.7141	0.7109	0.7116	**0.7162**	0.7119	0.6939
Isolet	0.9908	0.9936	0.9934	**0.9979**	**0.9979**	**0.9979**	0.9959
Magic	0.8453	0.8740	0.8693	0.8766	**0.8792**	0.8789	0.8644
Mushroom	0.9990	0.9996	0.9992	**1.0000**	**1.0000**	**1.0000**	0.9992
Musk	0.9394	0.9611	0.9432	**0.9947**	0.9932	0.9896	0.9725
Parkinsons	0.9060	0.8889	0.9231	0.9145	0.9402	**0.9487**	0.9316
Phishing	0.9264	0.9549	0.9492	0.9708	**0.9729**	0.9622	0.9417
Pima	0.7662	0.7771	**0.7792**	0.7576	0.7749	0.7641	0.7706
Sonar	0.7937	0.8175	0.8651	0.8492	0.8651	0.8571	**0.8968**
Spambase	0.9305	0.9428	0.9356	0.9399	**0.9457**	0.9446	0.9396
Waveform	0.8707	0.8820	0.8900	0.8930	0.8963	**0.9040**	0.9013
WDBC	0.9474	0.9532	**0.9620**	**0.9620**	0.9591	**0.9620**	**0.9620**
Wine	0.9444	0.9444	0.9722	**1.0000**	**1.0000**	**1.0000**	0.9907
WPBC	0.8000	0.7917	**0.8417**	0.7917	0.8333	0.8333	**0.8417**
Average Accuracy	0.8761	0.8855	0.8920	0.8943	**0.9012**	0.8998	0.8960
Win / Tie / Loss	—	18/2/4	**23/0/1**	18/1/5	21/2/1	22/0/2	21/1/2

6 Ablation Analysis

Isolating Co-adaptation vs. Oblique Tree Representational Power. To isolate the effect of global co-adaptation from the inherent representational capacity of oblique trees, we introduce an ablation baseline: **CGB-RO (No Global)**. This variant retains random oblique initialization and local residual fitting (Stage 2) but disables global co-adaptation (Stage 1). It functions as a standard, greedy boosting algorithm using oblique trees.

As Table 2 shows, disabling historical co-adaptation degrades performance. If majority of the gains come from the capacity of oblique hyperplanes, this baseline should match or exceed XGBoost. Instead, the CGB-RO (No Global) baseline underperforms XGBoost on 12 of the 24 datasets, whereas the fully co-adaptive CGB-RO variant outperforms XGBoost on 22 of the 24 datasets.

This confirms that higher representational power alone is insufficient. When oblique trees are permanently frozen, the algorithm cannot effectively align random boundaries.

7 Computational Complexity

Standard GBDTs utilize histogram-based split finding. For N samples and d features, building a discrete tree of depth D requires $\mathcal{O}(D \cdot N \cdot d)$ operations, after which parameters are permanently locked.

Table 2: Ablation study isolating the impact of global co-adaptation. Mean test accuracy ± standard deviation across 3 random seeds. The best performing method(s) for each dataset are bolded.

Dataset	XGBoost Baseline	CGB-RO (No Global)	CGB-RO (Full)
Adult	0.8488 ± 0.0049	0.7854 ± 0.0048	$\mathbf{0.8503} \pm 0.0047$
Bank Marketing	0.8987 ± 0.0004	0.8830 ± 0.0000	$\mathbf{0.9077} \pm 0.0020$
Blood Transfusion	0.7800 ± 0.0200	0.7600 ± 0.0000	$\mathbf{0.8067} \pm 0.0133$
Breast Cancer	0.9524 ± 0.0041	$\mathbf{0.9667} \pm 0.0041$	$\mathbf{0.9667} \pm 0.0082$
Chess	0.9401 ± 0.0039	0.9807 ± 0.0024	$\mathbf{0.9839} \pm 0.0018$
Ecoli	0.9412 ± 0.0389	$\mathbf{0.9510} \pm 0.0085$	0.9461 ± 0.0170
Electricity	0.7836 ± 0.0021	0.7695 ± 0.0017	$\mathbf{0.8079} \pm 0.0035$
German	0.7383 ± 0.0340	0.7000 ± 0.0000	$\mathbf{0.7617} \pm 0.0161$
Gisette	0.9506 ± 0.0025	0.9742 ± 0.0050	$\mathbf{0.9800} \pm 0.0044$
Glass	$\mathbf{0.8450} \pm 0.0585$	0.7829 ± 0.0269	0.8295 ± 0.0269
Higgs	0.6879 ± 0.0030	0.6300 ± 0.0032	$\mathbf{0.7119} \pm 0.0044$
Isolet	0.9908 ± 0.0004	0.9957 ± 0.0023	$\mathbf{0.9979} \pm 0.0027$
Magic	0.8453 ± 0.0100	0.8263 ± 0.0165	$\mathbf{0.8789} \pm 0.0083$
Mushroom	0.9990 ± 0.0007	0.9992 ± 0.0004	$\mathbf{1.0000} \pm 0.0000$
Musk	0.9394 ± 0.0061	0.8472 ± 0.0017	$\mathbf{0.9896} \pm 0.0037$
Parkinsons	0.9060 ± 0.0534	0.8547 ± 0.0148	$\mathbf{0.9487} \pm 0.0256$
Phishing	0.9264 ± 0.0033	0.9370 ± 0.0060	$\mathbf{0.9622} \pm 0.0038$
Pima	$\mathbf{0.7662} \pm 0.0260$	0.7316 ± 0.0099	0.7641 ± 0.0262
Sonar	0.7937 ± 0.0275	$\mathbf{0.8889} \pm 0.0364$	0.8571 ± 0.0412
Spambase	0.9305 ± 0.0105	0.9421 ± 0.0033	$\mathbf{0.9446} \pm 0.0022$
Waveform	0.8707 ± 0.0101	0.8867 ± 0.0103	$\mathbf{0.9040} \pm 0.0095$
WDBC	0.9474 ± 0.0088	0.9561 ± 0.0175	$\mathbf{0.9620} \pm 0.0134$
Wine	0.9444 ± 0.0556	0.9907 ± 0.0160	$\mathbf{1.0000} \pm 0.0000$
WPBC	0.8000 ± 0.0250	0.7750 ± 0.0000	$\mathbf{0.8333} \pm 0.0382$

Conversely, CGB maintains a continuous computation graph. Computing routing probabilities involves matrix multiplications between input features ($X \in \mathbb{R}^{N \times d}$) and routing weights ($W \in \mathbb{R}^{(2^D-1) \times d}$). Backpropagating through a full ensemble of t trees requires $\mathcal{O}(t \cdot N \cdot 2^D \cdot d)$ time and memory per epoch.

Our windowed update ($h = 3$) partially resolves this bottleneck. By freezing older trees and tracking gradients only for the newest w learners, the backward pass complexity is bounded to $\mathcal{O}(h \cdot N \cdot 2^D \cdot d)$ per epoch. Because w and D are small constants, the memory requirements of CGB remain computationally tractable per boosting iteration.

8 Limitations

While CGB offers predictive advantages, it introduces computational trade-offs. Standard GBDTs utilize fast, memory-efficient histogram-based split finding, whereas CGB must maintain a computation graph for historical co-adaptation. Although the sliding-window mechanism ($h = 3$) reduces this overhead, CGB remains more memory-intensive and slower per epoch than purely greedy baselines. Additionally, the CGB-RO variant introduces new hyperparameters like temperature annealing and an L^1 sparsity penalty, which require careful tuning to ensure convergence and prevent structural overfitting.

9 Conclusion

We challenge the assumption that gradient boosting requires strictly greedy, stagewise optimization by introducing Co-adaptive Gradient Boosting (CGB). Unifying additive growth with global parameter updates, CGB breaks the greedy lock to let historical weak learners co-adapt. Across tabular benchmarks, relaxing this constraint yields competitive predictive performance using either XGBoost priors or random oblique trees. Ablation confirms these gains stem primarily from unfreezing historical trees rather than oblique capacity alone.

AI USE STATEMENT

In this work, we used generative AI tools for proofreading, formatting the LaTeX manuscript to meet ICLR 2027 guidelines, and refining the clarity of mathematical notations. We have not used generative AI tools for generating the core algorithmic ideas, writing the experimental code, or synthesizing the empirical results. We have reviewed all AI-assisted text revisions and take full responsibility for the final content of this work.

ETHICS STATEMENT

This paper presents foundational research in machine learning algorithms and ensemble methods. The datasets utilized for empirical validation are standard, publicly available tabular benchmarks that do not contain sensitive or personally identifiable information. We do not foresee any immediate negative societal impacts or ethical concerns directly arising from the methodologies proposed in this work.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our findings, we have detailed all hyperparameter configurations, initialization regimes, and dataset splits in Sections 5.1 and 5.2. The characteristics and origins of all publicly available tabular datasets are documented in Appendix A.1. Furthermore, an anonymized implementation of the Co-adaptive Gradient Boosting (CGB) framework, including the differentiable tree architectures (`differentiable_tree.py`) and the multi-stage training pipelines (`cgb_earlystopping.py`), is provided in the supplementary materials. The benchmarking code (`cgb_xgb_benchmarking.py`) used to generate all empirical results is also included in supplementary materials.

REFERENCES

Stefan Aeberhard, Danny Coomans, and Olivier de Vel. Comparative analysis of statistical pattern recognition methods in high dimensional settings. *Pattern Recognition*, 25(12):1061–1067, 1992.

Sercan O Arik and Tomas Pfister. TabNet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6679–6687, 2021.

Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5(1):4308, 2014.

RK Bock, A Chilingarian, M Gaug, F Hakl, T Hengstebeck, M Jiřina, J Klaschka, E Kotrč, P Savickỳ, S Towers, et al. Methods for multidimensional event classification: a case study using images from a cherenkov gamma-ray telescope. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 516(2-3): 511–528, 2004.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 1984.

Miguel A Carreira-Perpinán and Pooya Tavallali. Alternating optimization of decision trees, with application to learning sparse oblique trees. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

Hugh A Chipman, Edward I George, and Robert E McCulloch. BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

Ron Cole and Mark Fanty. Isolet spoken letter recognition. Technical report, Oregon Graduate Institute of Science and Technology, 1990.

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2):31–71, 1997.

Dheeru Dua and Casey Graff. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences, 2019. URL `http://archive.ics.uci.edu/ml`.

Ian W Evett and Edgar J Spiehler. Rule-based induction for forensic science. *Knowledge Based Systems*, 1987.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pp. 1189–1232, 2001.

Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

R Paul Gorman and Terrence J Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural networks*, 1(1):75–89, 1988.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *Advances in Neural Information Processing Systems*, volume 35, pp. 507–520, 2022.

Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the NIPS 2003 feature selection challenge. In *Advances in Neural Information Processing Systems*, pp. 545–552, 2004.

Michael Harries. Splice-2 comparative evaluation: Electricity pricing. Technical report, School of Computer Science and Engineering, University of New South Wales, 1999.

Hussein Hazimeh, Natalia Ponomareva, Petros Mol, Zhenyu Jiang, and Alex Beutel. The tree ensemble layer: differentiability meets conditional computation. In *International Conference on Machine Learning*, pp. 4138–4148. PMLR, 2020.

Hans Hofmann. Statlog (german credit data) data set. UCI Machine Learning Repository, 1994.

Mark Hopkins, Erik Scholz, George Forman, and Jaap Suermondt. Spambase data set. UCI Machine Learning Repository, 1999.

Paul Horton and Kenta Nakai. A probabilistic classification system for predicting the cellular localization sites of proteins. *Intelligent systems in molecular biology*, 4:109–115, 1996.

Ozan Irsoy, Olcay Taner Yıldız, and Ethem Alpaydın. Soft decision trees. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pp. 1819–1822, 2012.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *KDD*, volume 96, pp. 202–207, 1996.

Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1467–1475, 2015.

Max A Little, Patrick E McSharry, Stephen J Roberts, Declan AE Costello, and Irene M Moroz. Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. *BioMedical Engineering OnLine*, 6(1):1–23, 2007.

Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems*, volume 12, 1999.

Rami M Mohammad, Fadi Thabtah, and Lee McCluskey. An assessment of features related to phishing websites using an automated technique. *International Journal of Computer Applications*, 48(15):49–54, 2012.

Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. In *International Conference on Learning Representations*, 2020.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Dorenko, and Andrey Gulin. CatBoost: unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, 31, 2018.

Jeff Schlimmer. Mushroom data set. UCI Machine Learning Repository, 1987.

Alen D Shapiro. *Structured induction in expert systems*. Addison-Wesley Longman Publishing Co., Inc., 1989.

Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

Jack W Smith, JE Everhart, WC Dickson, WC Knowler, and RS Johannes. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, pp. 261. American Medical Informatics Association, 1988.

W Nick Street, William H Wolberg, and Olvi L Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical Image Processing and Biomedical Visualization*, volume 1905, pp. 861–870. SPIE, 1993.

W Nick Street, Olvi L Mangasarian, and William H Wolberg. Computer-aided diagnosis of breast tumors. In *Mathematical Methods in Medical Imaging III*, volume 2299, pp. 2–11. SPIE, 1995.

Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

William H Wolberg and Olvi L Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the National Academy of Sciences*, 87(23): 9193–9196, 1990.

I-Cheng Yeh, Kuo-Jui Yang, and Tao-Ming Ting. Knowledge discovery on rfm model using bernoulli sequence. *Expert Systems with Applications*, 36(3):5866–5871, 2009.

A Appendix

A.1 Dataset Characteristics and Preprocessing

To ensure robust evaluation across diverse feature spaces, we used 24 publicly available tabular datasets from the UCI Machine Learning Repository and OpenML. All tasks are formulated as binary classification problems. The structural characteristics of these datasets are detailed in Table 3.

Preprocessing Pipeline: Prior to training, all datasets underwent a standardized preprocessing pipeline. Missing values were handled depending on the dataset domain: incomplete records containing missing categorical data (e.g., in the Adult dataset) were dropped, whereas missing numerical values in continuous datasets were imputed using the respective feature's column mean. Categorical variables were transformed using one-hot encoding (specifically, dummy encoding dropping the first

category to prevent collinearity). All models across all baselines were evaluated on the exact same preprocessed data splits (stratified 80-20 train-test split, evaluated across 3 distinct random seeds).

Binarization Strategies for Multi-class Datasets: For datasets that originally contained multiple classes, we applied specific binarization strategies:

- **Isolet (Speech):** The original task is to classify 26 English letters. We convert this into a binary problem by isolating the "E-set" letters (B, C, D, E, G, P, T, V, Z), which are difficult to distinguish due to rhyming, against all other letters.
- **Wine (Tabular):** The original dataset contains three cultivator classes (0, 1, 2). We cast this as a binary detection task for Class 0 (Cultivator A) against the rest (Cultivators B and C).
- **Glass, Ecoli, and Waveform:** For multi-class datasets, we applied One-vs-Rest binarization. The first class is labeled as the positive class ($y = 1$), while all remaining classes are grouped as the negative class ($y = 0$).

Table 3: Characteristics of the 24 tabular benchmark datasets used in our experiments. Note that the feature count (d) denotes the original number of features prior to one-hot encoding.

Dataset	**Samples** (N)	**Features** (d)	**Original Classes**	**Reference**
Adult	48,842	14	2	(Kohavi, 1996)
Bank Marketing	45,211	16	2	(Moro et al., 2014)
Blood Transfusion	748	4	2	(Yeh et al., 2009)
Breast Cancer	699	9	2	(Wolberg & Mangasarian, 1990)
Chess	3,196	36	2	(Shapiro, 1989)
Ecoli	336	8	8	(Horton & Nakai, 1996)
Electricity	45,312	8	2	(Harries, 1999)
German	1,000	20	2	(Hofmann, 1994)
Gisette	7,000	5,000	2	(Guyon et al., 2004)
Glass	214	9	6	(Evett & Spiehler, 1987)
Higgs	98,050	28	2	(Baldi et al., 2014)
Isolet	7,797	617	26	(Cole & Fanty, 1990)
Magic	19,020	10	2	(Bock et al., 2004)
Mushroom	8,124	22	2	(Schlimmer, 1987)
Musk	6,598	166	2	(Dietterich et al., 1997)
Parkinsons	195	22	2	(Little et al., 2007)
Phishing	11,055	30	2	(Mohammad et al., 2012)
Pima	768	8	2	(Smith et al., 1988)
Sonar	208	60	2	(Gorman & Sejnowski, 1988)
Spambase	4,601	57	2	(Hopkins et al., 1999)
Waveform	5,000	21	3	(Breiman et al., 1984)
WDBC	569	30	2	(Street et al., 1993)
Wine	178	13	3	(Aeberhard et al., 1992)
WPBC	198	33	2	(Street et al., 1995)

A.2 Extended Implementation Details and Hyperparameters

To ensure a fair and consistent comparison across all algorithmic variants, we established a standardized set of base hyperparameters for both the XGBoost baseline and the Co-adaptive Gradient Boosting (CGB) framework. The maximum ensemble size was fixed at $T = 30$ trees, the maximum tree depth was constrained to $D = 3$, and the boosting shrinkage parameter was set to $\alpha = 0.1$.

Table 4 details the structural and optimization configurations across all variants. Because the XGBoost-initialized models rely on exact mathematical priors, they do not require routing regularization. Conversely, the CGB-RO variants initialize from random weights. To steer these random topologies toward optimal structures, CGB-RO employs L_1 sparsity regularization and dynamic temperature annealing throughout both the local initialization and global co-adaptation stages. For the ablation baseline, CGB-RO (No Global), these regularizations are restricted to the local fitting phase.

Table 4: Configuration summary for all algorithmic variants, detailing initialization strategies, routing topologies, and the application scope of sparsity penalties and temperature annealing.

Algorithm Variant	Initialization	Routing Topology	Global Co-adaptation	L_1 Penalty	Temp. Annealing
XGBoost Baseline	Greedy Prior	Axis-Parallel	None (Frozen)	None	None
CGB-AP (Full)	XGBoost Warm-start	Axis-Parallel	Full Ensemble	None	None
CGB-AP ($h = 3$)	XGBoost Warm-start	Axis-Parallel	Window ($h = 3$)	None	None
CGB-OB (Full)	XGBoost Warm-start	Oblique	Full Ensemble	None	None
CGB-OB ($h = 3$)	XGBoost Warm-start	Oblique	Window ($h = 3$)	None	None
CGB-RO (Full)	Random Weights	Oblique	Full Ensemble	Local & Global	Local & Global
CGB-RO ($h = 3$)	Random Weights	Oblique	Window ($h = 3$)	Local & Global	Local & Global
CGB-RO (No Global)	Random Weights	Oblique	None (Frozen)	Local Only	Local Only

Optimization and Early Stopping (CGB Framework): The differentiable decision trees in the CGB framework were implemented using PyTorch, with continuous routing probabilities computed via a sigmoid activation function. The model parameters were updated using the Adam optimizer across a two-stage process:

- **Stage 1 (Global Co-adaptation):** Optimized with a learning rate of $\eta = 0.01$ for a maximum of 100 epochs. As outlined in Table 4, windowed variants restrict this gradient update to the most recent $h = 3$ trees.
- **Stage 2 (Local Initialization):** Optimized with a learning rate of 0.05 for a maximum of 60 epochs.

To prevent structural overfitting on the pseudo-residuals, both optimization stages employed an early stopping mechanism. A hold-out validation loss was monitored, triggering a halt if the loss failed to improve for 10 consecutive epochs. Upon early stopping, the framework automatically restored the model weights to the best-performing state.

Regularization and Temperature Annealing Setup: For the randomly initialized CGB-RO variants, we applied specific numerical constraints to enforce sparsity:

- L_1 **Sparsity Penalty:** An L_1 regularization penalty with $\lambda = 0.001$ was applied to the feature selector weights ($\mathbf{w}_n$). For fully co-adaptive models, this penalty was integrated into the loss function during both local and global optimization phases, forcing the discovery of sparse linear combinations.
- **Dynamic Temperature Annealing:** To ensure the soft routing probabilities harden into crisp decision boundaries by convergence, the routing temperature τ was annealed linearly over the course of the epochs. During Stage 2 (Local Initialization), τ was scaled linearly from $1.0 \rightarrow 10.0$. During Stage 1 (Global Co-adaptation), τ was scaled linearly from $2.0 \rightarrow 10.0$.

For the XGBoost-initialized variants (CGB-AP and CGB-OB), temperature annealing is bypassed and the routing temperature is held fixed at $\tau = 10.0$ throughout all training stages.

B Full Experimental Results with Variance

This section provides the complete empirical results corresponding to the main text. Table 5 details the mean test accuracy and the standard deviation for all 24 tabular datasets.

C Standard Gradient Boosted Decision Trees (GBDT)

For completeness we detail the standard stagewise optimization procedure utilized by traditional GBDTs in Algorithm 4.

Table 5: Complete experimental results showing mean test accuracy ± standard deviation across 3 random seeds. The best performing method(s) for each dataset are bolded.

	Baseline	CGB-AP		CGB-OB		CGB-RO	
Dataset	XGBoost	Full	$h = 3$	Full	$h = 3$	Full	$h = 3$
Adult	0.8488 ± 0.0049	$\mathbf{0.8562} \pm 0.0015$	0.8537 ± 0.0039	0.8398 ± 0.0025	0.8463 ± 0.0025	0.8503 ± 0.0047	0.8470 ± 0.0033
Bank Marketing	0.8987 ± 0.0004	0.9065 ± 0.0017	0.9047 ± 0.0017	0.9008 ± 0.0024	0.9046 ± 0.0011	$\mathbf{0.9077} \pm 0.0020$	0.9033 ± 0.0031
Blood Transfusion	0.7800 ± 0.0200	0.7933 ± 0.0240	0.7911 ± 0.0301	0.7778 ± 0.0154	0.7889 ± 0.0315	$\mathbf{0.8067} \pm 0.0133$	$\mathbf{0.8067} \pm 0.0200$
Breast Cancer	0.9524 ± 0.0041	0.9571 ± 0.0143	0.9571 ± 0.0143	0.9619 ± 0.0109	$\mathbf{0.9667} \pm 0.0082$	$\mathbf{0.9667} \pm 0.0082$	$\mathbf{0.9667} \pm 0.0109$
Chess	0.9401 ± 0.0039	0.9750 ± 0.0000	0.9651 ± 0.0033	0.9839 ± 0.0024	$\mathbf{0.9859} \pm 0.0016$	0.9839 ± 0.0018	0.9792 ± 0.0024
Ecoli	0.9412 ± 0.0389	0.9314 ± 0.0225	0.9265 ± 0.0147	0.9412 ± 0.0147	0.9412 ± 0.0147	$\mathbf{0.9461} \pm 0.0170$	$\mathbf{0.9461} \pm 0.0170$
Electricity	0.7836 ± 0.0021	0.8007 ± 0.0037	0.7937 ± 0.0047	$\mathbf{0.8444} \pm 0.0017$	0.8361 ± 0.0032	0.8079 ± 0.0035	0.7836 ± 0.0005
German	0.7383 ± 0.0340	0.7350 ± 0.0218	0.7550 ± 0.0087	0.7417 ± 0.0306	0.7600 ± 0.0391	$\mathbf{0.7617} \pm 0.0161$	0.7583 ± 0.0375
Gisette	0.9506 ± 0.0025	0.9575 ± 0.0055	0.9653 ± 0.0063	0.9750 ± 0.0068	0.9794 ± 0.0086	0.9800 ± 0.0044	$\mathbf{0.9828} \pm 0.0051$
Glass	0.8450 ± 0.0585	0.8450 ± 0.0484	$\mathbf{0.8605} \pm 0.0403$	0.8372 ± 0.0403	0.8450 ± 0.0134	0.8295 ± 0.0269	0.8295 ± 0.0355
Higgs	0.6879 ± 0.0030	0.7141 ± 0.0025	0.7109 ± 0.0029	0.7116 ± 0.0000	$\mathbf{0.7162} \pm 0.0027$	0.7119 ± 0.0044	0.6939 ± 0.0022
Isolet	0.9908 ± 0.0004	0.9936 ± 0.0036	0.9934 ± 0.0026	$\mathbf{0.9979} \pm 0.0021$	$\mathbf{0.9979} \pm 0.0021$	$\mathbf{0.9979} \pm 0.0027$	0.9959 ± 0.0013
Magic	0.8453 ± 0.0100	0.8740 ± 0.0098	0.8693 ± 0.0074	0.8766 ± 0.0054	$\mathbf{0.8792} \pm 0.0041$	0.8789 ± 0.0083	0.8644 ± 0.0099
Mushroom	0.9990 ± 0.0007	0.9996 ± 0.0007	0.9992 ± 0.0004	$\mathbf{1.0000} \pm 0.0000$	$\mathbf{1.0000} \pm 0.0000$	$\mathbf{1.0000} \pm 0.0000$	0.9992 ± 0.0004
Musk	0.9394 ± 0.0061	0.9611 ± 0.0019	0.9432 ± 0.0020	$\mathbf{0.9947} \pm 0.0040$	0.9932 ± 0.0047	0.9896 ± 0.0037	0.9725 ± 0.0062
Parkinsons	0.9060 ± 0.0534	0.8889 ± 0.0534	0.9231 ± 0.0769	0.9145 ± 0.0645	0.9402 ± 0.0392	$\mathbf{0.9487} \pm 0.0256$	0.9316 ± 0.0392
Phishing	0.9264 ± 0.0033	0.9549 ± 0.0030	0.9492 ± 0.0032	0.9708 ± 0.0041	$\mathbf{0.9729} \pm 0.0039$	0.9622 ± 0.0038	0.9417 ± 0.0053
Pima	0.7662 ± 0.0260	0.7771 ± 0.0198	$\mathbf{0.7792} \pm 0.0172$	0.7576 ± 0.0246	0.7749 ± 0.0135	0.7641 ± 0.0262	0.7706 ± 0.0320
Sonar	0.7937 ± 0.0275	0.8175 ± 0.0275	0.8651 ± 0.0364	0.8492 ± 0.0550	0.8651 ± 0.0137	0.8571 ± 0.0412	$\mathbf{0.8968} \pm 0.0137$
Spambase	0.9305 ± 0.0105	0.9428 ± 0.0070	0.9356 ± 0.0072	0.9399 ± 0.0006	$\mathbf{0.9457} \pm 0.0029$	0.9446 ± 0.0022	0.9396 ± 0.0027
WDBC	0.9474 ± 0.0088	0.9532 ± 0.0221	$\mathbf{0.9620} \pm 0.0183$	$\mathbf{0.9620} \pm 0.0183$	0.9591 ± 0.0203	$\mathbf{0.9620} \pm 0.0134$	$\mathbf{0.9620} \pm 0.0134$
WPBC	0.8000 ± 0.0250	0.7917 ± 0.0289	$\mathbf{0.8417} \pm 0.0289$	0.7917 ± 0.0144	0.8333 ± 0.0382	0.8333 ± 0.0382	$\mathbf{0.8417} \pm 0.0144$
Waveform	0.8707 ± 0.0101	0.8820 ± 0.0104	0.8900 ± 0.0089	0.8930 ± 0.0113	0.8963 ± 0.0085	$\mathbf{0.9040} \pm 0.0095$	0.9013 ± 0.0121
Wine	0.9444 ± 0.0556	0.9444 ± 0.0481	0.9722 ± 0.0278	$\mathbf{1.0000} \pm 0.0000$	$\mathbf{1.0000} \pm 0.0000$	$\mathbf{1.0000} \pm 0.0000$	0.9907 ± 0.0160

Algorithm 4 Standard Gradient Boosted Decision Trees (GBDT)

Require: Dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, iterations T, shrinkage α

1: Initialize ensemble $F_0(\boldsymbol{x}) = \arg\min_c \sum_{i=1}^{N} \ell(y_i, c)$
2: **for** $t = 1$ **to** T **do**
3: # *Stage 1: Compute Pseudo-Residuals (No Global Update)*
4: Compute first-order pseudo-residuals: $r_{i,t} = -\left[\frac{\partial \ell(y_i, F)}{\partial F}\right]_{F=F_{t-1}(\boldsymbol{x}_i)}$
5: # *Stage 2: Greedy Tree Construction (Local Fit)*
6: Initialize a new discrete regression tree f_t
7: Greedily grow f_t by finding an axis-parallel split (feature j, threshold s) that partitions a node's samples I into left ($I_L = \{i \in I \mid x_{ij} \leq s\}$) and right ($I_R = \{i \in I \mid x_{ij} > s\}$) subsets to minimize the Mean Squared Error (MSE):

$$\text{Split} = \arg\min_{j,s} \left[\sum_{i \in I_L} (r_{i,t} - \bar{r}_L)^2 + \sum_{i \in I_R} (r_{i,t} - \bar{r}_R)^2 \right]$$

8: *(where $\bar{r}_L$ and $\bar{r}_R$ are the mean pseudo-residuals in I_L and I_R)*
9: Compute the optimal weight for each terminal leaf node R_{tj}:

$$v_{tj} = \arg\min_{v} \sum_{\boldsymbol{x}_i \in R_{tj}} \ell\left(y_i, F_{t-1}(\boldsymbol{x}_i) + v\right)$$

10: Freeze all parameters (boundaries and leaf weights) of f_t permanently
11: # *Ensemble Update*
12: $F_t(\boldsymbol{x}) \leftarrow F_{t-1}(\boldsymbol{x}) + \alpha f_t(\boldsymbol{x})$
13: **end for**
14: **return** Frozen ensemble $F_T(\boldsymbol{x})$