Even Faster Hyperbolic Random Forests: A Beltrami-Klein Wrapper Approach

Anonymous authors
Paper under double-blind review

Abstract

Decision trees and models that use them as primitives are workhorses of machine learning in Euclidean spaces. Recent work has further extended these models to the Lorentz model of hyperbolic space by replacing axis-parallel hyperplanes with homogeneous hyperplanes when partitioning the input space. In this paper, we show how the HyperDT algorithm can be elegantly reexpressed in the Beltrami-Klein model of hyperbolic spaces. This preserves the thresholding operation used in Euclidean decision trees, enabling us to further rewrite HyperDTas simple pre– and post-processing steps that form a wrapper around existing tree-based models designed for Euclidean spaces. The wrapper approach unlocks many optimizations already available in Euclidean space models, improving flexibility, speed, and accuracy while offering a simpler, more maintainable, and extensible codebase.

1 Introduction

All machine learning classifiers implicitly encode geometric assumptions about their feature space. Among these, standard decision trees and classifiers built on them stand out as being remarkably indifferent to geometry: Decision tree learning is ultimately a discrete optimization over label statistics, one feature at a time. The metric structure of the multi-feature space only matters when evaluating its final partition.

In contrast, all linear and neural models relying on matrix multiplication implicitly work with signed distances to a decision-boundary hyperplane; probabilistic methods based on Gaussian densities such as Naive Bayes have probability densities that depend on distance to the centroid; not to mention explicitly distance-based methods such as support vector machines (SVMs) and k-nearest neighbors. Yet decision trees rely only on label statistics for candidate partitions which depend only on the ordering of features along each dimension: a property we exploit to reformalize their behavior in hyperbolic space.

There are two existing approaches in the literature to extending decision trees to hyperbolic spaces. Doorenbos et al. (2023) proposes HoroRF, an ensemble of horospherical support vector machines (Fan et al., 2023), equating between horospheres in hyperbolic and hyperplanes in Euclidean space, as well as between SVMs and linear splits at a decision tree node. In contrast, Chlenski et al. (2024) proposes HyperDT taking a hyperplane-based perspective on Euclidean decision tree learning and following up with a modification of the splitting criterion to accommodate axis-inclined hyperplane splits.

By exploiting the (almost-Euclidean) behavior of hyperplanes in the ambient space of the Lorentz model of hyperbolic space, HyperDT is able to more closely parallel Euclidean tree learning. In particular, it is able to consider each spacelike dimension separately and has identical training and inference complexity to its Euclidean counterparts. However, its modified, hyperplane-based splitting criteria are no longer compatible with the threshold-based decision tree algorithms, necessitating its own ad hoc implementation. This implementation cannot make use of the many optimizations made to the speed and accuracy of existing decision tree methods.

In this paper, we rewrite HyperDT as Fast-HyperDT, relying on the Beltrami-Klein model of hyperbolic space. This representation allows geodesic decision boundaries to become (axis-parallel) Euclidean hyperplanes, recovering a thresholding-based variant of HyperDT. Fast-HyperDT unlocks three advantages:

1. Leveraging optimized implementations is thousands of times faster than existing HyperDT;

2. It is intrinsically compatible with well-known Euclidean methods like Scikit-Learn; and

3. It easily extends to other tree-based paradigms, such as oblique decision trees or XGBoost.

1.1 Related work

Non-Euclidean Decision Trees. This work is most closely related to other works proposing the use of tree-based models in non-Euclidean spaces. Chlenski et al. (2024) is the starting point for our model; Chlenski et al. (2025a) extends decision trees to (products of) any constant-curvature space; Doorenbos et al. (2023) offers a complementary, horosphere-based perspective on random forests in hyperbolic space. Tsagkrasoulis & Montana (2017) proposes a method to fit random forests for non-Euclidean *labels*, while (Bauerschmidt et al., 2021) explores connections between random spanning forests and hyperbolic symmetry.

Hyperbolic Classifiers. The problem of classification and regression in hyperbolic spaces has received a lot of attention in the literature. Many approaches involve modifying neural networks to work with hyperbolic data (Chami et al., 2019; Ganea et al., 2018; Chen et al., 2022; Bdeir et al., 2023; Khan et al., 2024). We are indebted to the literature on hyperbolic SVMs (Cho et al., 2018; Fan et al., 2023); the former makes use of the Klein model as well. Regression on hyperbolic manifolds has been studied by Marconi et al. (2020).

Machine Learning in the Beltrami-Klein Model. Mao et al. (2024) extends the Klein model for neural nets and provides an excellent overview of the state of the literature, including applications of the Klein model for graph embeddings (McDonald & He, 2019; Yang et al., 2023), protein sequences (Ali et al., 2024), minimal spanning trees (García-Castellanos et al., 2025), and scene images (Bi et al., 2017). We highlight Nielsen & Nock (2010), an early work using the Klein model to find Voronoi diagrams in hyperbolic space, whose discussion of partitions of the Beltrami-Klein model was foundational for our work.

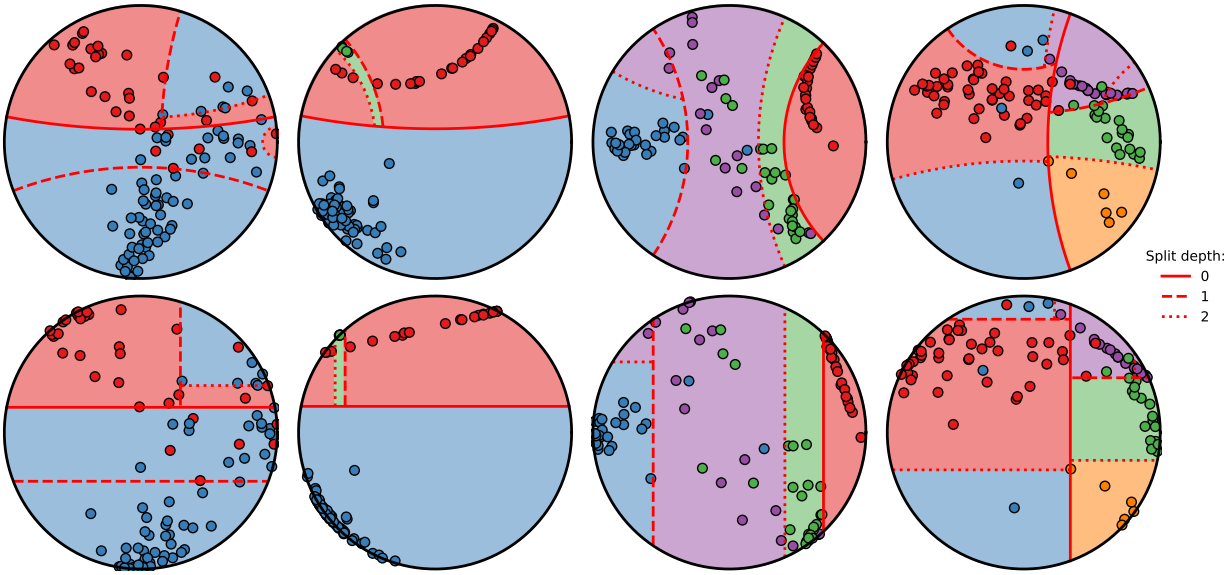

Figure 1: Original HyperDT decision boundaries visualized in the Poincaré model (top) and in the Beltrami-Klein model (bottom). The top half of the figure is a reproduction of Figure 2 in Chlenski et al. (2024).

2 Preliminaries

2.1 Hyperbolic Space

2.1.1 The Lorentz Model of Hyperbolic Space

The Lorentz Model with curvature $K < 0$ in d dimensions, $\mathbb{L}_K^d$, is embedded inside Minkowski space, a vector space in $\mathbb{R}^{d+1}$ equipped with the Minkowski inner product:

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathbb{L}} = -u_0 v_0 + \sum_{i=1}^{d} u_i v_i. \tag{1}$$

Points in the Lorentz Model are constrained to lie on the upper sheet of a two-sheeted hyperboloid with a constant Minkowski inner product K, which is also called the curvature:

$$\mathbb{L}_K^d = \left\{ \mathbf{u} \in \mathbb{R}^{d+1} : \langle \mathbf{u}, \mathbf{u} \rangle_{\mathbb{L}} = \frac{1}{K},\ u_0 > 0 \right\}. \tag{2}$$

Distances between two points $\mathbf{u}, \mathbf{v} \in \mathbb{L}$, called geodesic distances, are computed as[1]

$$\delta_{\mathbb{L}}(\mathbf{u}, \mathbf{v}) = \frac{1}{\sqrt{-K}} \cosh^{-1}\left(-K \langle \mathbf{u}, \mathbf{v} \rangle_{\mathbb{L}}\right). \tag{3}$$

2.1.2 The Beltrami-Klein Model of Hyperbolic Space

The Beltrami-Klein Model realizes hyperbolic space as the open ball of radius $-1/K$

$$\mathbb{B}_K^d = \left\{ \mathbf{u} \in \mathbb{R}^d : \langle \mathbf{u}, \mathbf{u} \rangle < -\frac{1}{K} \right\}. \tag{4}$$

$\mathbb{L}_K^d$ and $\mathbb{B}_K^d$ are related to through the gnomonic projection $\phi_K : \mathbb{L}_K^d \to \mathbb{B}_K^d$ and its inverse $\phi_K^{-1} : \mathbb{B}_K^d \to \mathbb{L}_K^d$:

$$\phi_K(u_0, u_1, \ldots, u_d) = \left(\frac{u_1}{u_0}, \frac{u_2}{u_0}, \ldots, \frac{u_d}{u_0} \right) \tag{5}$$

$$\phi_K^{-1}(v_1, \ldots v_d) = \left(\frac{1}{\sqrt{1 - K\|\mathbf{v}\|^2}},\ \frac{v_1}{\sqrt{1 - K\|\mathbf{v}\|^2}},\ \ldots,\ \frac{v_d}{\sqrt{1 - K\|\mathbf{v}\|^2}} \right). \tag{6}$$

Note that the indices for $\mathbf{u}$ start at 0, whereas the indices of $\mathbf{v}$ start at 1, reflecting the use of an extra dimension in the Hyperboloid model. This correspondence not only maps points but also sends hyperbolic geodesics (great hyperbola arcs on $\mathbb{L}$) to chords in $\mathbb{B}$, preserving the hyperbolic distance between points.

The distance between $\mathbf{u}, \mathbf{v} \in \mathbb{B}$ is computed by composing $\phi_K^{-1}(\cdot)$ and $\delta_{\mathbb{L}}(\cdot)$ as follows:

$$\begin{aligned} \delta_{\mathbb{B}}(\mathbf{u}, \mathbf{v}) &= \delta_{\mathbb{L}}(\phi_K^{-1}(\mathbf{u}), \phi_K^{-1}(\mathbf{v})) \\ &= \frac{1}{\sqrt{-K}} \cosh^{-1}\big(-K\ \langle \phi_K^{-1}(\mathbf{u}), \phi_K^{-1}(\mathbf{v}) \rangle\big) \\ &= \frac{1}{\sqrt{-K}} \cosh^{-1}\left(-K \frac{1 - \langle \mathbf{u}, \mathbf{v} \rangle}{\sqrt{(1 - K\|\mathbf{u}\|^2)(1 - K\|\mathbf{v}\|^2)}} \right). \end{aligned} \tag{7}$$

Equivalently, $\delta_{\mathbb{B}}$ can be computed using the following geometric construction: if the chord through two interior points $\mathbf{b}$ and $\mathbf{c}$ meets the boundary of the Klein disk at points $\mathbf{a}$ and $\mathbf{b}$ (with the ordering $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ along the chord), then the hyperbolic distance between $\mathbf{b}$ and $\mathbf{c}$ is given by

$$\delta_{\mathbb{B}}(\mathbf{b}, \mathbf{c}) = \frac{1}{2} \ln\left(\frac{|\mathbf{ac}||\mathbf{bd}|}{|\mathbf{ab}||\mathbf{cd}|} \right), \tag{8}$$

where $|\mathbf{ab}|$ denotes the Euclidean norm of the line segment connecting $\mathbf{a}$ and $\mathbf{b}$, and so on.

[1] ip: I think we should use arccosh arctan etc. instead of the inverse (negative 1 superscript)

Einstein Midpoints. A key advantage of the Beltrami-Klein model we will make use of in this paper is its closed-form formulation of geodesic midpoints. Given two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{B}$, the Einstein midpoint is given by

$$m_{\mathbb{B}}(\mathbf{u}, \mathbf{v}) = \frac{\gamma_{\mathbf{u}}\mathbf{u} + \gamma_{\mathbf{v}}\mathbf{v}}{\gamma_{\mathbf{u}} + \gamma_{\mathbf{v}}}, \text{ where } \gamma_{\mathbf{x}} = \frac{1}{\sqrt{1 - K\|\mathbf{x}\|^2}}. \tag{9}$$

Note that $\gamma_{\mathbf{x}}$ is simply the timelike dimension under the inverse gnomonic projection ϕ_K^{-1} given in Equation 6.

2.2 HyperDT

HyperDT and HyperRF, which were introduced in (Chlenski et al., 2024), are generalizations of Decision Trees (Classification and Regression Trees, (Breiman, 2017)) and Random Forests (Breiman, 2001) when $\mathbf{X} \in \mathbb{L}_K^{n \times d}$, $\mathbf{y} \in \mathbb{R}^n$.

2.2.1 Geodesically-Convex Splits

The central contribution of the HyperDT algorithm is to observe that splits in conventional decision trees can be thought of in terms of dot products with the normal of a separating axis-parallel hyperplane:

$$S(\mathbf{x}, i, t) = \mathbf{1}_{x_d > t} = \text{Sign}(\mathbf{x} \cdot \mathbf{e}^{(i)} - t)^+, \tag{10}$$

where $\mathbf{e}^{(i)}$ is a one-hot vector in dimension $i \in \{0, \dots d\}$, and t can be thought of as a bias term inducing a translation of the decision boundary by t away from the origin along $\mathbf{e}_i$. This observation naturally allows us to consider a more generic split function

$$S(\mathbf{x}, \mathbf{n}, t) = \text{Sign}(\mathbf{x} \cdot \mathbf{n} - t), \tag{11}$$

where $\mathbf{n} \in \mathbb{R}^{d+1}$ is the normal vector for any separating hyperplane, which we will call H. In HyperDT, the set of candidate separating hyperplanes is restricted to those whose normal vectors take the form

$$\mathbf{n}(i, \theta) = (-\cos(\theta),\ 0,\ \dots,\ 0,\ \sin(\theta),\ \dots,\ 0), \tag{12}$$

where dimensions 0 and i are the only nonzero dimensions, and where $t = 0$, ultimately yielding the HyperDT splitting criterion:

$$S(\mathbf{x}, i, \theta) = \text{Sign}(\mathbf{x} \cdot \mathbf{n}(i, \theta)) = \text{Sign}(\mathbf{x}_i \sin(\theta) - x_0 \cos(\theta)). \tag{13}$$

Removing the bias term ensures geodesic convexity, as only homogeneous hyperplanes (i.e. hyperplanes containing the origin) have geodesically convex intersections with $\mathbb{L}$; on the other hand, parameterizing splits in terms of a spacelike dimension i and an angle θ allows HyperDT to retain the expressiveness and time complexity of traditional decision tree algorithms.

2.2.2 Hyperbolic Angular Midpoints

In CART, if a split falls between $\mathbf{u}, \mathbf{v} \in \mathbf{X}_{\text{train}}$, then the boundary is conventionally placed directly in between $\mathbf{u}$ and $\mathbf{v}$. That is, for a split in dimension i, we set the threshold using a simple arithmetic mean. In Euclidean space, this has the property that $\mathbf{u}$ and $\mathbf{v}$ are now equidistant from the separating hyperplane through $h_i = t$, which is a reasonable inductive bias for choosing where to separate two classes.

In hyperbolic space, the naive midpoint is likewise the angular bisector

$$\theta_{\text{naive}} = \frac{\theta_1 + \theta_2}{2}, \tag{14}$$

but in this case the hyperplanes parameterized by $h_0 \cos(\theta_1) = h_d \sin(\theta_1)$ and $h_0 \cos(\theta_2) = h_d \sin(\theta_2)$ would not intersect $\mathbb{L}$ at points that are equidistant to the hyperplane $h_0 \cos(\theta) = h_d \sin(\theta)$ under the hyperbolic distance metric $\delta_{\mathbb{L}}$. In order to ensure equidistance, HyperDT instead uses a more complex midpoint formula derived from $\delta_{\mathbb{L}}$:

$$m_{\mathbb{L}}(\theta_1, \theta_2) = \cot^{-1}(\alpha - \beta\sqrt{\alpha^2 - 1}), \text{ where } \alpha = \frac{\sin(2\theta_1 - 2\theta_2)}{2\sin(\theta_1 + \theta_2)\sin(\theta_2 - \theta_1)},\ \beta = \text{Sign}\left(\theta_1 + \theta_2 - \pi\right). \tag{15}$$

3 Speeding Up HyperDT

The algorithm for speeding up HyperDT training (see Algorithm 1) involves three key steps:

1. **Preprocessing:** To preprocess $\mathbf{X} \in \mathbb{L}^{n \times d}$ for compatibility with Euclidean classifiers, it suffices to project $\mathbf{X}$ to the Beltrami-Klein model $\mathbf{X}_{\mathbb{B}} \in \mathbb{B}^{n \times d}$ using ϕ_K as defined in Equation 5.
2. **Train using Scikit-Learn:** Train a Scikit-Learn-compatible Decision Tree or Random Forest on the transformed data $\mathbf{X}_{\mathbb{B}}$, yielding a trained Euclidean predictor $\mathcal{R}$.
3. **Postprocess:** Recompute geodesic midpoints for all decision boundaries in $\mathcal{R}$ and store them in the corrected hyperbolic predictor $\mathcal{R}_{\mathbb{B}}$. Given two points $\mathbf{u}, \mathbf{v} \in \mathbb{B}$, the corrected midpoint is simply the Einstein midpoint along the line connecting $\mathbf{u}$ and $\mathbf{v}$, as in Equation 9. This modification can be applied recursively to a trained tree, taking u and v as the closest values to either side of the learned threshold.

This approach allows us to leverage efficient Euclidean implementations of decision trees and random forests (for instance, Scikit-Learn's RandomForestClassifier class) while maintaining the correct geometry of hyperbolic space with curvature K.

To speed up inference, we propose Algorithm 2, which preserves the asymptotic complexity of CART and HyperDT (see Theorem 4.6). For most practical purposes, it is generally faster to preprocess and use the built-in prediction functionality of the base model instead, as in Algorithm 3.

Algorithm 1: Fast-HyperDT training

Input: Dataset $\mathbf{X} \in \mathbb{L}_K^{n \times d}$, labels $\mathbf{y} \in \mathbb{R}^n$, curvature $K < 0$
Output: Hyperbolic Random Forest $\mathcal{R} = \{\mathcal{T}\}_{i=1}^{n_{\text{trees}}}$ (where $n_{\text{trees}} = 1$ for a single tree)

```
Function Preprocess(X, K):
    return φ_K(X)                                          // Project to Klein model
Function AdjustThresholds(T, X, K):
    if T is a leaf then
        return
    d ← T.feature
    t ← T.threshold
    X ← X[T.subsample_indices]     // Random forests may use bootstrapped subsamples of X
    X⁺ ← {x ∈ X : x_d > t}
    X⁻ ← {x ∈ X : x_d ≤ t}
    L ← max_{x∈X⁻}{x_d}
    R ← min_{x∈X⁺}{x_d}
    node.threshold ← EinsteinMidpoint(L, R, K)                // Uses Equation 9
    AdjustThresholds(T.left, X⁻)
    AdjustThresholds(T.right, X⁺)
Function Postprocess(R, X, K):
    for T ∈ R do
        AdjustThresholds(T, X, K, T)
    return R
Function Train(X_𝕃, y, K):
    X_𝔹 ← Preprocess(X_𝕃, K)
    R_𝔼 ← TrainEuclideanModel(X_𝔹, y)        // For instance, DecisionTreeClassifier.fit()
    R_𝔹 ← Postprocess(R_𝔼, X_𝔹, K)
    return R_𝔹
```

Algorithm 2: Fast-HyperDT inference ($O(nh)$ version)

Input: Dataset $\mathbf{X} \in \mathbb{L}_K^{n \times d}$, trained model $\mathcal{R}_{\mathbb{B}}$
Output: Predictions $\hat{\mathbf{y}} \in \mathbb{R}^d$

Function `PredictNode`$(\mathbf{x}, \mathcal{T})$**:**
  if $\mathcal{T}$ *is a leaf* **then**
    return $\mathcal{T}$.label
  $d \leftarrow \mathcal{T}$.feature
  $t \leftarrow \mathcal{T}$.threshold
  $\phi(\mathbf{x})_d \leftarrow x_d / x_0$ `// Compute Klein model coordinates selectively`
  if $\phi(\mathbf{x})_d \leq t$ **then**
    return `PredictNode`$(\mathbf{x}, \mathcal{T}$.left$)$
  else
    return `PredictNode`$(\mathbf{x}, \mathcal{T}$.right$)$

Function `Predict`$(\mathbf{X}_{\mathbb{L}}, \mathcal{R}_{\mathbb{B}})$**:**
  $\hat{\mathbf{y}} \leftarrow$ array of size $|\mathbf{X}_{\mathbb{L}}|$
  for $i \leftarrow 1$ **to** n **do**
    $\mathrm{V} \leftarrow \{\}$ `// Initialize empty multiset`
    for $\mathcal{T} \in |\mathcal{R}_{\mathbb{B}}|$ **do**
      $v \leftarrow$ `PredictNode`$(\mathbf{X}_{\mathbb{B}}[i], \mathcal{T})$
      $\mathrm{V} \leftarrow V \cup \{v\}$ `// Add prediction to multiset`
    $\hat{\mathbf{y}}[i] \leftarrow$ `Resolve`(V) `// However your ensemble aggregates tree votes`
  return $\hat{\mathbf{y}}$

Algorithm 3: Fast-HyperDT inference (simple $O(n(d+h))$ version)

Input: Dataset $\mathbf{X} \in \mathbb{L}_K^{n \times d}$, curvature $K < 0$, trained model $\mathcal{R}_{\mathbb{B}}$
Output: Predictions $\hat{\mathbf{y}} \in \mathbb{R}^d$

Function `Predict`$(\mathbf{X}_{\mathbb{L}}, K, \mathcal{R}_{\mathbb{B}})$**:**
  $\mathbf{X}_{\mathbb{B}} \leftarrow$ `Preprocess`$(\mathbf{X}_{\mathbb{L}}, K)$
  return `PredictEuclidean`$(\mathbf{X}_{\mathbb{B}}, \mathcal{R}_{\mathbb{B}})$ `// For instance, DecisionTreeClassifier.predict()`

3.1 Extensions

3.1.1 Poincaré Ball Model

It is common for hyperbolic machine learning approaches to use the Poincaré ball model (which we will denote $\mathbb{P}_K$) rather than the hyperboloid or Beltrami-Klein models. We omit a detailed discussion of the properties of the Poincaré ball model, except to mention that it is possible to apply our approach to $\mathbf{X}_P \in \mathbb{P}_K^{n,d}$ using a projection $\rho_K : \mathbb{P}_K^d \to \mathbb{B}_K^d$. We include its inverse for completeness.

$$\rho_K(u_1, \ldots, u_d) = \left(\frac{u_1}{1 + \sqrt{1 - K\|\mathbf{u}\|^2}}, \frac{u_2}{1 + \sqrt{1 - K\|\mathbf{u}\|^2}}, \ldots, \frac{u_d}{1 + \sqrt{1 - K\|\mathbf{u}\|^2}} \right) \tag{16}$$

$$\rho_K^{-1}(v_1, \ldots, v_d) = \left(\frac{2v_1}{1 - K\|\mathbf{v}\|^2}, \frac{2v_2}{1 - K\|\mathbf{v}\|^2}, \ldots, \frac{2v_d}{1 - K\|\mathbf{v}\|^2} \right) \tag{17}$$

Once the points are projected to the Beltrami-Klein model, the rest of the computations proceed exactly as in the hyperboloid case.

All HyperDT classifiers are initialized with an `input_geometry` hyperparameter, which could be one of `hyperboloid`, `klein`, or `poincare`, and the appropriate conversion is applied during preprocessing.

3.1.2 XGBoost

The extension of our approach to XGBoost (Chen & Guestrin, 2016) follows naturally from the decision tree implementation, as XGBoost fundamentally relies on the same axis-parallel splitting mechanism. While XGBoost introduces gradient boosting, regularization, and sophisticated loss function optimization, the geometric interpretation of its decision boundaries remains unchanged. Each tree in the trained XGBoost ensemble can be postprocessed analogously to a Scikit-Learn Random Forest.

The primary implementation challenge lies in accessing individual tree structures within the XGBoost model, as the internal representation differs from Scikit-Learn's. Nevertheless, once the nodes are accessible, the threshold adjustment process remains identical, maintaining the correct hyperbolic geometry throughout the ensemble. We allow XGBoost as a valid backend in our implementation.

Because XGBoost, unlike Scikit-Learn, does not store a record of sample indices used to train each individual tree, we introduce a `override_subsample` hyperparameter to our XGBoost model. By default, `override_subsample = True`, resetting the `subsample` hyperparameter of all XGBoost models to 1.0. If users choose to set `override_subsample = False`, the XGBoost model trains using subsampling, but the postprocessing uses the entire training set. In this case, a warning is issued that postprocessing is approximate.

3.1.3 LightGBM

Similar to XGBoost, LightGBM (Ke et al., 2017) is a popular library for gradient-boosted decision trees. Also analogous to XGBoost, the innovations of LightGBM do not affect split geometry, so trained LightGBM models can be postprocessed as usual by editing thresholds in LightGBM's proprietary text-based model format. The subsampling issue also affects LightGBM models, so if users choose to set `bagging_fraction != 0`, the model also issues a warning and performs approximate postprocessing on the entire dataset.

3.1.4 Oblique Decision Trees

For oblique decision trees, which use linear combinations of features rather than axis-parallel splits, our approach requires a natural extension to the projection mechanism. Instead of projecting onto basis dimension axes, we project data points onto the normal vector of each oblique hyperplane. Given a hyperplane defined by $\mathbf{x} \cdot \mathbf{n} - t = 0$, we compute the projection of each point onto the normal vector $\mathbf{n}$. The threshold correction then proceeds identically to the axis-parallel case, applying the Einstein midpoint formula from Equation 9 to the scalar projections rather than to individual feature values.

This generalization maintains the geometric consistency of the decision boundaries in hyperbolic space while leveraging the increased flexibility of oblique splits. The implementation requires modifying the node structure to store the hyperplane parameters $\mathbf{w}$ and applying the projection during both training and inference phases, but the core principle of threshold adjustment remains unchanged.

Using this projection-based approach, we provide a hyperbolic variant of the HHCart (Wickramarachchi et al., 2015; 2019) and CO2 Forest (Norouzi et al., 2015) algorithms, as implemented in the `scikit-obliquetree` library (ECNU, 2021).

4 Theoretical Results

In this section, we establish the theoretical foundation for our Fast-HyperDT algorithm by proving its equivalence to the original HyperDT method while demonstrating its computational advantages. The key insight of our approach lies in exploiting the geometric properties of the Beltrami-Klein model, where geodesics appear as straight lines, to leverage efficient Euclidean decision tree implementations.

Lemma 4.1 (Hyperplane Classification Equivalence)**.** *Let* $\mathbf{n} = (-\cos\theta, 0, ..., 0, \sin\theta, 0, ...) \in \mathbb{R}^{d+1}$ *be a normal vector for a hyperplane. For any* $\mathbf{x} \in \mathbb{L}_K^d$*, the* HyperDT *decision rule satisfies:*

$$\text{Sign}(\langle \mathbf{x}, \mathbf{n} \rangle) = \text{Sign}\left(\phi_K(\mathbf{x})_i - \cot(\theta)\right). \tag{18}$$

Proof.

$$\langle \mathbf{x}, \mathbf{n} \rangle > 0 \iff -x_0 \cos(\theta) + x_i \sin(\theta) > 0 \iff \frac{x_i}{x_0} > \frac{\cos(\theta)}{\sin(\theta)} \iff \phi_K(\mathbf{x})_i > \cot(\theta). \tag{19}$$

Since $x_0 > 0$ in the Lorentz model, dividing by x_0 never affects the direction of this inequality. □

Lemma 4.2 (Threshold Invariance)**.** *For any decision tree trained to optimize an information gain objective, the feature space partitioning is invariant to the choice of threshold placement between adjacent feature values. Specifically, if $u < v$ are consecutive observed values of a feature, then any threshold $t \in (u, v)$ produces identical tree structures and decision boundaries.*

Proof. Let us define the Information Gain (IG) for splitting a dataset $(\mathbf{X}, \mathbf{y})$ using threshold t on feature i:

$$\begin{aligned} \mathrm{IG}(\mathbf{y}, t) = H(\mathbf{y}) - \left(\frac{|\mathbf{y}^+|}{|\mathbf{y}|} H(\mathbf{y}^+) + \frac{|\mathbf{y}^-|}{|\mathbf{y}|} H(\mathbf{y}^-) \right), \text{ where} \\ \mathbf{y}^+ = \{ y_j \in \mathbf{y} \ : \ (x_j)_i \geq t \}, \quad \mathbf{y}^- = \{ y_j \in \mathbf{y} \ : \ (x_j)_i < t \}. \end{aligned} \tag{20}$$

Here, $H(\cdot)$ denotes some impurity measure, e.g. Gini impurity. Now, for some values $u < v$ such that no $\mathbf{x} \in \mathbf{X}$ has a value between u and v at feature i, then the partition of $\mathbf{y}$ into $\mathbf{y}^+$ and $\mathbf{y}^-$ remains identical for any choice of $t \in (u, v)$.

Because information gain depends solely on this partition and not on the specific threshold value, the IG value remains constant for all $t \in (u, v)$. Consequently, any such threshold produces identical tree structures and decision boundaries when optimizing an information gain objective. □

Lemma 4.3 (Midpoint Equivalence)**.** *Let $\mathbf{u}, \mathbf{v} \in \mathbb{L}_K^d$ be two points in the hyperboloid model. Define $\theta_x = \cot^{-1}\left(\frac{x_d}{x_0}\right)$ for any point $\mathbf{x} = (x_0, \ \ldots, \ x_d, \ \ldots) \in \mathbb{L}_K^d$. Similarly, let $\phi : \mathbb{L}_K^d \to \mathbb{B}_K^d$ be the usual gnomonic projection into the Klein model so that $\phi(\mathbf{x})_d = \frac{x_d}{x_0}$. Then the hyperbolic angular midpoint of $\mathbf{u}, \mathbf{v}$ in $\mathbb{L}_K^d$ as defined in Eq. 15 is the same as the Einstein midpoint of $\phi(\mathbf{u}), \phi(\mathbf{v})$ in $\mathbb{B}_K^d$ as defined in Eq. 9:*

$$\theta_{\mathbb{L}}(\theta_u, \theta_v) = \theta_{m_{\mathbb{B}}(\phi(\mathbf{u}), \phi(\mathbf{v}))} = \cot^{-1}\left(\frac{m_{\mathbb{B}}\left(\phi(\mathbf{u})_d, \phi(\mathbf{v})_d\right)_d}{m_{\mathbb{B}}\left(\phi(\mathbf{u})_d, \phi(\mathbf{v})_d\right)_0} \right) \tag{21}$$

Proof. We will show that the hyperbolic angular midpoint in the hyperboloid model and the Einstein midpoint in the Klein model correspond to the same point by verifying that both produce points equidistant from the original endpoints.

In hyperbolic geometry, midpoints are unique: given any two points, there exists exactly one point equidistant from both along the geodesic connecting them. This is implied by the geodesic convexity of hyperbolic spaces (Ratcliffe, 2019). Therefore, if we can show that both constructions yield points in the same 2-D subspace with equal distances to the endpoints, they must be the same point.

Consider two points $\mathbf{u}, \mathbf{v} \in \mathbb{L}_K^d$. Let $\theta_{\mathbb{L}}(\mathbf{u}, \mathbf{v})$ be the hyperbolic angular midpoint of $\mathbf{u}$ and $\mathbf{v}$, as defined in Equation 15. Let $\mathbf{m}_{\mathbb{L}}$ denote the unique point in $\mathbb{L} \cap h_0 \cos(\theta_L) - h_d \sin(\theta_{\mathbb{L}}) = 0$. Then by the definition of a midpoint,

$$\delta_{\mathbb{L}}(m_{\mathbb{L}}, \mathbf{u}) = \delta_{\mathbb{L}}(m_{\mathbb{L}}, \mathbf{v}). \tag{22}$$

This property of the hyperbolic midpoint is derived in Chlenski et al. (2024).

The Einstein midpoint of two points $\mathbf{u}, \mathbf{v} \in \mathbb{L}_K^n$ can equivalently be written as:

$$m_{\mathbb{B}} = \frac{\mathbf{u} + \mathbf{v}}{\|\mathbf{u} + \mathbf{v}\|_{\mathbb{L}}} \cdot \frac{1}{\sqrt{K}}, \tag{23}$$

where $\|\mathbf{w}\|_{\mathbb{L}} = \sqrt{-K \langle \mathbf{w}, \mathbf{w} \rangle_{\mathbb{L}}}$ for a timelike vector $\mathbf{w}$.

Without loss of generality, assume $\mathbf{u} = (u_0,\ 0,\ \ldots,\ u_d,\ \ldots,\ 0)$ and $\mathbf{v} = (v_0,\ 0,\ \ldots,\ v_d,\ \ldots,\ 0)$ where the nonzero components are in positions 0 and d.[2] Simplifying the Einstein midpoint formula,

$$m_{\mathbb{B}} = \frac{(u_0 + v_0,\ 0,\ \ldots,\ u_d + v_d,\ \ldots,\ 0)}{K\sqrt{-[(u_0+v_0)^2 - (u_d+v_d)^2]}} \tag{24}$$

To prove equidistance, we compute:

$$\langle m_{\mathbb{B}}, \mathbf{u}\rangle_{\mathbb{L}} = \frac{(u_0+v_0)u_0 - (u_d+v_d)u_d}{K\sqrt{-[(u_0+v_0)^2 - (u_d+v_d)^2]}} \tag{25}$$

$$\langle m_{\mathbb{E}}, \mathbf{v}\rangle_{\mathbb{L}} = \frac{(u_0+v_0)v_0 - (u_d+v_d)v_d}{K\sqrt{-[(u_0+v_0)^2 - (u_d+v_d)^2]}} \tag{26}$$

For these to be equal, we need:

$$\begin{aligned}(u_0+v_0)u_0 - (u_d+v_d)u_d &= (u_0+v_0)v_0 - (u_d+v_d)v_d \\ u_0^2 + u_0v_0 - u_d^2 - u_dv_d &= v_0^2 + u_0v_0 - v_d^2 - u_dv_d \\ u_0^2 - u_d^2 &= v_0^2 - v_d^2. \end{aligned} \tag{27}$$

Since $u_0^2 - u_d^2 = v_0^2 - v_d^2 = 1/K$ from the hyperboloid constraint (Eq. 2), the equation is satisfied. Therefore, $\delta_{\mathbb{L}}(m_{\mathbb{B}}, \mathbf{u}) = \delta_{\mathbb{L}}(m_{\mathbb{B}}, \mathbf{v})$, proving that the Einstein midpoint is equidistant from points $\mathbf{u}$ and $\mathbf{v}$ under the hyperbolic distance. □

Remark 4.4. The three lemmas correspond directly to our algorithm's stages: (1) Klein projection enables Euclidean thresholds (Lemma 4.1; (2) Invariance permits off-the-shelf training (Lemma 4.2); and (3) Midpoint correction recovers hyperbolically equidistant decision boundaries (Lemma 4.3).

Theorem 4.5 (Algorithmic Equivalence)**.** *Assuming that ties in information gain never occur, or are handled identically by* HyperDT *and Fast-*HyperDT*, both methods produce identical decision boundaries. More precisely: letting $\mathcal{D}_H$ be the partition of $\mathbb{L}$ induced by* HyperDT *and $\mathcal{D}_F$ be the partition of $\mathbb{B}$ induced by Fast-*HyperDT *on the same dataset, $\phi(\mathcal{D}_H) \equiv \mathcal{D}_F$.*

Proof. We proceed by induction on the depth of a split.

Base case. At depth 0, the manifold is unpartitioned: $\phi(\mathcal{D}_H) = \phi(\mathbb{L}) = \mathbb{B} = \mathcal{D}_F$.

Inductive hypothesis. Assume that we have $\phi(\mathcal{D}_H) = \mathcal{D}_F$ for all D, F of maximum depth d.

Inductive step. By the inductive hypothesis, any region $R \subseteq \mathbb{L}$ being split at depth $d+1$ is already equivalent via $\phi(\mathcal{D}_H) = \mathcal{D}_F$. By Lemma 4.1, partitions by equivalent hyperplanes induce equivalent splits, and by Lemma 4.2, both algorithms learn the same partition of the labels. Finally, by Lemma 4.3 the midpoints of the partitioned labels also coincide. □

Theorem 4.6 (Computational Complexity)**.** *Let n be the number of training points, d the number of input features, and $h \leq \log n$ the height of each (balanced) tree. With presorted feature lists, Fast-*HyperDT *trains in $O(nd \log n)$ time and performs inference in $O(hn)$ time, matching both Euclidean CART and* HyperDT*.*

Proof. Let t denote the number of trees in the ensemble (set $t = 1$ for a single tree). We analyze one tree and multiply by t at the end.

CART. With presorted features, a standard Euclidean decision tree fits in $O(dn \log n)$ time and predicts in $O(hn)$ time (Sani et al., 2018). The level-order scan variant costs $O(dnh)$ to fit, but we quote the presorted bound for definiteness. Bounding $h \leq \log_2(n)$, a reasonable assumption for balanced trees, equates the two complexities.

[2]This is already the case for the axis-parallel splits used in conventional decision trees. For oblique boundaries, it is always possible to rotate the input such that its normal vector is sparse in this way.

HyperDT. HyperDT differs from CART by a constant-time modification to the thresholding rule, so its core fitting cost is asymptotically that of CART.

Preprocessing. Applying $\phi_K(\cdot)$ to every sample touches each coordinate once, giving $O(nd)$.

Postprocessing. Each internal node (at most $2^h - 1 < 2n$) needs the two sample values nearest its threshold. Because the active sample set halves at every depth, a single linear pass suffices, so this stage is $O(n)$ per depth, giving $O(nh)$ total.

Training. Thus, the total training time for Fast-HyperDT is

$$O(\text{train}) = O(dn) + O(dn \log n) + O(hn) = O(dn \log n) \tag{28}$$

under the assumption that $h \leq \log_2(n)$. Multiplying by t yields $O(tdn \log n)$ for an ensemble.

Inference. Algorithm 2 visits exactly h nodes per example and evaluates the projection at a single dimension for each visit, so prediction is $O(hn)$ per tree, or $O(thn)$ for the ensemble. □

5 Experimental Results

5.1 Experimental Setup

To match Scikit-Learn's tiebreaking behavior more closely, we modified HyperDT by changing the split criterion from $x < t$ to $x \leq t$ and reversing the order in which points are considered. We also set the random seed for Scikit-Learn decision trees to a constant value. This is necessary because, even when Scikit-Learn decision trees have subsampling disabled, they still randomly permute the features, which can affect the tiebreaking behavior of the algorithm.

Except where otherwise specified, we used trees with a maximum depth of 3 and no further restrictions on what splits are considered (e.g. minimum number of points in a leaf). We sample synthetic data from a mixture of wrapped Gaussians (Nagano et al., 2019), a common way of benchmarking hyperbolic classifiers (Cho et al., 2018; Chlenski et al., 2024). For regression datasets, we follow Chlenski et al. (2025b) in applying cluster-specific slopes and intercepts to the initial vectors sampled from Gaussian distribution to generate regression targets.

5.2 Agreement and Timing Benchmarks

To evaluate the time to train Fast-HyperDT, we trained HyperDT and Fast-HyperDT decision trees on varying numbers of samples from a mixture of wrapped Gaussian distributions. Figure 2 compares the training speeds of HyperDT and Fast-HyperDT on varying numbers of samples, revealing a 3,752× speedup when training decision trees on 32,768 samples.

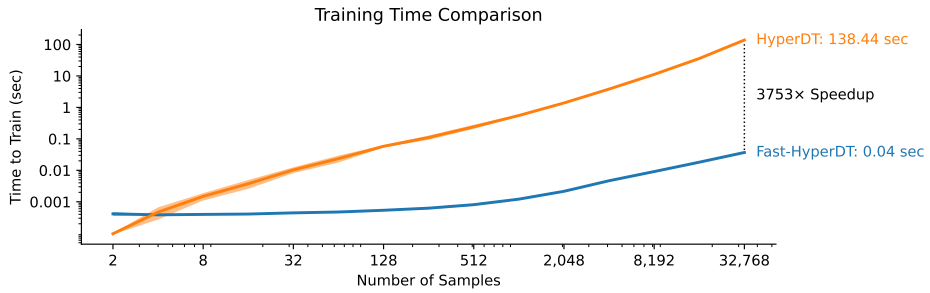

Figure 2: We compare the training time of HyperDT and Fast-HyperDT across different training set sizes. Fast-HyperDT is consistently faster than HyperDT for 8 or more samples; for 32,768 samples, Fast-HyperDT trained on average 3,752× faster.

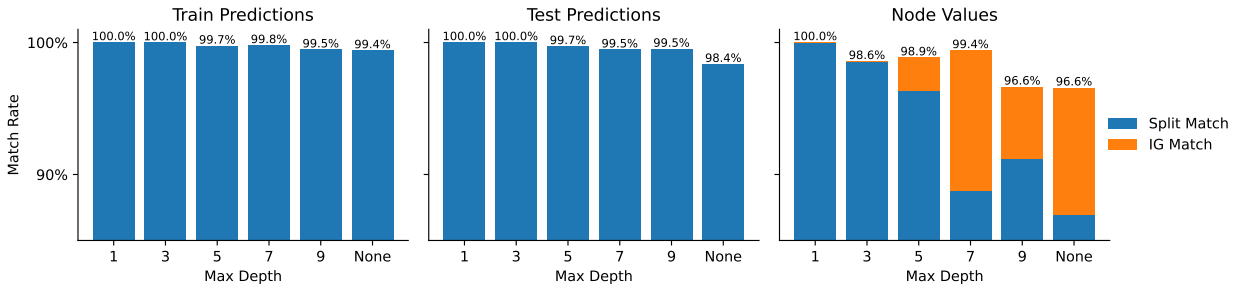

Figure 3: We compare the predictions and split values of HyperDT and Fast-HyperDT. **Left:** The agreement between both models on the training data. **Middle:** The agreement between both models on the testing data. **Right:** Node-by-node split agreement, distinguishing between exact matches (blue) and matches that are equivalent in information gain (orange).

Given the theoretical results in Section 4, it is natural to expect the predictions and splits of HyperDT and Fast-HyperDT to match exactly; instead, Figure 3 reveals a high but imperfect degree of correspondence between the two. We speculated that differences in pragmatic factors, such as tiebreaking rules (which split to prefer when two splits have the same information gain) and numerical precision, might be behind these slight differences in training behavior.

To investigate, we tested 10,000 different seeds comparing HyperDT and Fast-HyperDT on Gaussian mixtures of 1,000 points. Of these, we found that 9,934 splits matched exactly, while 43 splits matched in information gain (up to a tolerance of 0.0001). Figure 4 shows an example of information gains for 4 sets of features, revealing a tie between the best split along Features 1 and 2. Although we attempted to align the tiebreaking behavior of Scikit-Learn and HyperDT decision trees, we were never able to perfectly match the splits in all cases.

For the remaining 22 splits, Fast-HyperDT always achieved a higher information gain than HyperDT did, and angles tended to cluster near $\pi/4$ and $3\pi/4$. This seems to suggest that HyperDT suffers from some numerical stability issues for extreme angles, which Fast-HyperDT, likely by virtue of omitting the inverse tangent operation used to compute angles in the original HyperDT algorithm, manages to avoid.

5.3 Other Models

In Tables 1 and 2, we evaluate Fast-HyperDT alongside a variety of other models on classification and regression (respectively) of Gaussian mixtures. We evaluate two baselines using Scikit-Learn on coordinates

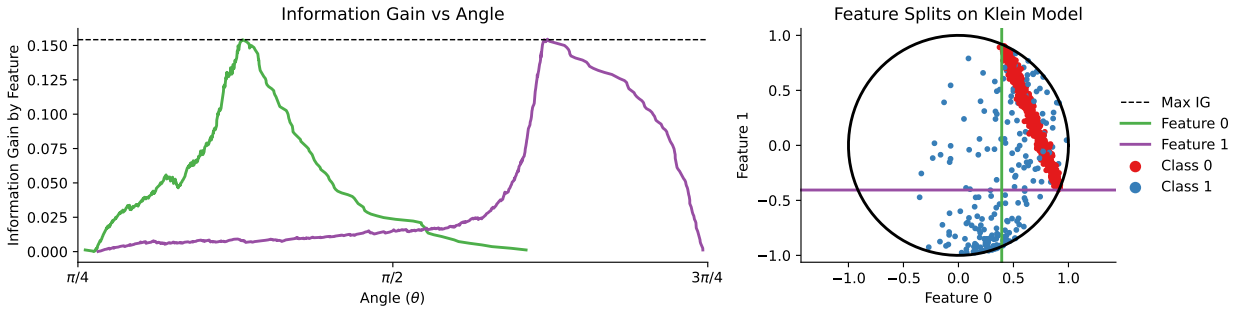

Figure 4: A Gaussian mixture dataset for which multiple splits attain the same information gain. **Left:** The information gains attained by each candidate split for each feature. **Right:** Each point in the Klein model, colored by class, and two splits on different features that attain the same information gain.

Table 1: Classification accuracies across 100 synthetic 8-class benchmarks using Gaussian mixture datasets of varying dimensionalities.

	Scikit-Learn DT		Fast-HyperDT			Scikit-Learn RF		Fast-HyperRF		
d	Lorentz	Klein	HyperDT	CO2	HHC	Lorentz	Klein	HyperRF	LightGBM	XGBoost
2	48.98	50.75	50.75	18.07	11.44	49.63	54.59	54.34	57.94	59.44
4	41.76	43.24	43.24	16.68	13.63	46.35	49.77	49.99	56.66	59.60
8	36.76	37.41	37.41	15.59	11.88	42.76	45.21	45.23	54.10	58.06
16	32.15	32.29	32.27	13.85	13.46	37.08	39.82	39.83	49.86	54.20
32	28.66	28.91	28.90	12.38	12.58	32.92	35.06	35.29	44.26	48.67
64	25.73	25.86	25.88	13.10	12.22	28.89	30.45	30.41	38.23	41.02
128	24.66	25.56	25.55	13.69	12.87	26.76	28.11	28.07	33.44	35.20

Table 2: Regression mean squared error (MSE) scores across 100 synthetic regression datasets. Aside from using regression variants of all models, the benchmarking setup is otherwise identical to Table 1.

	Scikit-Learn DT		Fast-HyperDT			Scikit-Learn RF		Fast-HyperRF		
d	Lorentz	Klein	HyperDT	CO2	HHC	Lorentz	Klein	HyperRF	LightGBM	XGBoost
2	.0297	.0273	.0273	.0293	.1294	.0267	.0253	.0252	.0279	.0214
4	.0299	.0277	.0277	.0293	.1461	.0258	.0236	.0238	.0253	.0174
8	.0245	.0242	.0242	.0247	.1643	.0207	.0199	.0200	.0202	.0140
16	.0258	.0251	.0251	.0251	.1821	.0215	.0209	.0206	.0210	.0146
32	.0241	.0236	.0236	.0235	.2035	.0204	.0198	.0198	.0198	.0146
64	.0246	.0246	.0246	.0236	.2159	.0210	.0207	.0207	.0206	.0164
128	.0239	.0237	.0237	.0229	.2261	.0206	.0201	.0202	.0201	.0175

in the Lorentz model and the Beltrami-Klein model. We evaluate the base model of Fast-HyperDT against oblique decision trees (CO2 and HHC), as well as XGBoost and LightGBM.

Oblique decision trees vastly underperformed even the baselines, but are able to learn better-than-random decision rules; for regression, CO2 trees are even competitive with the baseline models. In all cases, XGBoost models performed best, highlighting the value of extending this technique to hyperbolic space.

Finally, we find that base HyperDT performance is only slightly better than Scikit-Learn in Beltrami-Klein coordinates. Because the latter can be thought of as Fast-HyperDT without the final postprocessing step, this effectively functions as an ablation of postprocessing. This result recapitulates earlier findings in (Chlenski et al., 2024) suggesting that ablating the hyperbolic midpoints does not significantly impact HyperDT performance. We note that the two ablations actually coincide, as the angular bisector and the average of the Klein coordinates are equal.[3]

6 Conclusions

In this work, we proposed Fast-HyperDT, which rewrites HyperDT as a wrapper around Euclidean tree-based models using the Beltrami-Klein model of hyperbolic space. We prove our method is equivalent to HyperDT while being simpler, faster, and more flexible. We also demonstrate the superior speed and extensibility of our method empirically, in particular noting that the XGBoost variant of Fast-HyperDT is vastly more accurate than base HyperDT.

Future work can focus on extending Fast-HyperDT to other methods, such as Isolation Forests (Liu et al., 2008) and rotation forests (Bagnall et al., 2020), which would address the absence of privileged basis dimensions (Elhage et al., 2023) in hyperbolic embedding methods; extending Fast-HyperDT to hyperspherical data as in Chlenski et al. (2025a); and extending the connection between HyperDT, and decision trees to neural networks, as in Aytekin (2022) or via the polytope lens (Black et al., 2022).

[3]This can be shown by converting angles $\theta_u = \cot^{-1}(u_d/u_0)$ and $\theta_v = \cot^{-1}(v_d/v_0)$ to their bisector $\theta_m = (\theta_u + \theta_v)/2$, then mapping back to Klein coordinates as $\cot(\theta_m) = (u_d/x_0 + x_d/x_0)/2$.

References

Sarwan Ali, Haris Mansoor, Prakash Chourasia, Yasir Ali, and Murray Patterson. Gaussian Beltrami-Klein Model for Protein Sequence Classification: A Hyperbolic Approach. In Wei Peng, Zhipeng Cai, and Pavel Skums (eds.), *Bioinformatics Research and Applications*, pp. 52–62, Singapore, 2024. Springer Nature. ISBN 9789819751280. doi: 10.1007/978-981-97-5128-0_5.

Caglar Aytekin. Neural Networks are Decision Trees, October 2022. URL http://arxiv.org/abs/2210.05189. arXiv:2210.05189 [cs].

A. Bagnall, M. Flynn, J. Large, J. Line, A. Bostrom, and G. Cawley. Is rotation forest the best classifier for problems with continuous features?, April 2020. URL http://arxiv.org/abs/1809.06705. arXiv:1809.06705 [cs].

Roland Bauerschmidt, Nicholas Crawford, Tyler Helmuth, and Andrew Swan. Random spanning forests and hyperbolic symmetry. *Communications in Mathematical Physics*, 381(3):1223–1261, February 2021. ISSN 0010-3616, 1432-0916. doi: 10.1007/s00220-020-03921-y. URL http://arxiv.org/abs/1912.04854. arXiv:1912.04854 [math].

Ahmad Bdeir, Kristian Schwethelm, and Niels Landwehr. Hyperbolic Geometry in Computer Vision: A Novel Framework for Convolutional Neural Networks, March 2023. URL https://arxiv.org/abs/2303.15919v2.

Yanhong Bi, Bin Fan, and Fuchao Wu. Multiple Cayley-Klein metric learning. *PLoS ONE*, 12(9):e0184865, September 2017. ISSN 1932-6203. doi: 10.1371/journal.pone.0184865. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5608239/.

Sid Black, Lee Sharkey, Leo Grinsztajn, Eric Winsor, Dan Braun, Jacob Merizian, Kip Parker, Carlos Ramón Guevara, Beren Millidge, Gabriel Alfour, and Connor Leahy. Interpreting Neural Networks through the Polytope Lens, November 2022. URL http://arxiv.org/abs/2211.12312. arXiv:2211.12312 [cs].

Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

Leo Breiman. *Classification and Regression Trees*. Routledge, New York, October 2017. ISBN 978-1-315-13947-0. doi: 10.1201/9781315139470.

Ines Chami, Rex Ying, Christopher Ré, and Jure Leskovec. Hyperbolic Graph Convolutional Neural Networks, October 2019. URL http://arxiv.org/abs/1910.12933. arXiv:1910.12933 [cs, stat].

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, August 2016. doi: 10.1145/2939672.2939785. URL http://arxiv.org/abs/1603.02754. arXiv:1603.02754 [cs].

Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Fully Hyperbolic Neural Networks, March 2022. URL http://arxiv.org/abs/2105.14686. arXiv:2105.14686 [cs].

Philippe Chlenski, Ethan Turok, Antonio Moretti, and Itsik Pe'er. Fast hyperboloid decision tree algorithms, March 2024. URL http://arxiv.org/abs/2310.13841. arXiv:2310.13841 [cs].

Philippe Chlenski, Quentin Chu, Raiyan R. Khan, Kaizhu Du, Antonio Khalil Moretti, and Itsik Pe'er. Mixed-curvature decision trees and random forests, February 2025a. URL http://arxiv.org/abs/2410.13879. arXiv:2410.13879 [cs].

Philippe Chlenski, Kaizhu Du, Dylan Satow, and Itsik Pe'er. Manify: A Python Library for Learning Non-Euclidean Representations, March 2025b. URL http://arxiv.org/abs/2503.09576. arXiv:2503.09576 [cs] version: 1.

Hyunghoon Cho, Benjamin DeMeo, Jian Peng, and Bonnie Berger. Large-Margin Classification in Hyperbolic Space, June 2018. URL http://arxiv.org/abs/1806.00437. arXiv:1806.00437 [cs, stat].

Lars Doorenbos, Pablo Márquez-Neila, Raphael Sznitman, and Pascal Mettes. Hyperbolic Random Forests, August 2023. URL http://arxiv.org/abs/2308.13279. arXiv:2308.13279 [cs].

ECNU. Oblique Decision Tree in Python, 2021. URL https://github.com/zhenlingcn/scikit-obliquetree. Publication Title: GitHub repository.

Nelson Elhage, Robert Lasenby, and Christopher Olah. Privileged Bases in the Transformer Residual Stream, 2023. URL https://transformer-circuits.pub/2023/privileged-basis/index.html.

Xiran Fan, Chun-Hao Yang, and Baba C. Vemuri. Horospherical Decision Boundaries for Large Margin Classification in Hyperbolic Space, June 2023. URL http://arxiv.org/abs/2302.06807. arXiv:2302.06807 [cs, stat].

Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic Neural Networks, June 2018. URL http://arxiv.org/abs/1805.09112. arXiv:1805.09112 [cs, stat].

Alejandro García-Castellanos, Aniss Aiman Medbouhi, Giovanni Luca Marchetti, Erik J. Bekkers, and Danica Kragic. *HyperSteiner: Computing Heuristic Hyperbolic Steiner Minimal Trees*. January 2025. URL http://arxiv.org/abs/2409.05671. arXiv:2409.05671 [cs].

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. 2017.

Raiyan R. Khan, Philippe Chlenski, and Itsik Pe'er. Hyperbolic Genome Embeddings. October 2024. URL https://openreview.net/forum?id=NkGDNM8LB0.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, December 2008. doi: 10.1109/ICDM.2008.17. URL https://ieeexplore.ieee.org/document/4781136. ISSN: 2374-8486.

Yidan Mao, Jing Gu, Marcus C. Werner, and Dongmian Zou. Klein Model for Hyperbolic Neural Networks, October 2024. URL http://arxiv.org/abs/2410.16813. arXiv:2410.16813 [cs].

Gian Maria Marconi, Lorenzo Rosasco, and Carlo Ciliberto. Hyperbolic Manifold Regression, May 2020. URL http://arxiv.org/abs/2005.13885. arXiv:2005.13885 [cs, stat].

David McDonald and Shan He. HEAT: Hyperbolic Embedding of Attributed Networks, May 2019. URL http://arxiv.org/abs/1903.03036. arXiv:1903.03036 [cs].

Yoshihiro Nagano, Shoichiro Yamaguchi, Yasuhiro Fujita, and Masanori Koyama. A Wrapped Normal Distribution on Hyperbolic Space for Gradient-Based Learning, May 2019. URL http://arxiv.org/abs/1902.02992. arXiv:1902.02992 [cs, stat].

Frank Nielsen and Richard Nock. Hyperbolic Voronoi diagrams made easy. In *2010 International Conference on Computational Science and Its Applications*, pp. 74–80, 2010. doi: 10.1109/ICCSA.2010.37. URL http://arxiv.org/abs/0903.3287. arXiv:0903.3287 [cs].

Mohammad Norouzi, Maxwell D. Collins, David J. Fleet, and Pushmeet Kohli. CO2 Forest: Improved Random Forest by Continuous Optimization of Oblique Splits, June 2015. URL http://arxiv.org/abs/1506.06155. arXiv:1506.06155 [cs].

John G. Ratcliffe. Hyperbolic Geometry. In John G. Ratcliffe (ed.), *Foundations of Hyperbolic Manifolds*, pp. 52–96. Springer International Publishing, Cham, 2019. ISBN 9783030315979. doi: 10.1007/978-3-030-31597-9_3. URL https://doi.org/10.1007/978-3-030-31597-9_3.

Habiba Muhammad Sani, Ci Lei, and Daniel Neagu. Computational Complexity Analysis of Decision Tree Algorithms. In Max Bramer and Miltos Petridis (eds.), *Artificial Intelligence XXXV*, pp. 191–197, Cham, 2018. Springer International Publishing. ISBN 9783030041915. doi: 10.1007/978-3-030-04191-5_17.

Dimosthenis Tsagkrasoulis and Giovanni Montana. Random Forest regression for manifold-valued responses, February 2017. URL http://arxiv.org/abs/1701.08381. arXiv:1701.08381 [stat].

D. C. Wickramarachchi, B. L. Robertson, M. Reale, C. J. Price, and J. Brown. HHCART: An Oblique Decision Tree, April 2015. URL http://arxiv.org/abs/1504.03415. arXiv:1504.03415 [stat].

D. C. Wickramarachchi, B. L. Robertson, M. Reale, C. J. Price, and J. A. Brown. A reflected feature space for CART. *Australian & New Zealand Journal of Statistics*, 61(3):380–391, 2019. ISSN 1467-842X. doi: 10.1111/anzs.12275. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/anzs.12275.

Meimei Yang, Qiao Liu, Xinkai Sun, Na Shi, and Hui Xue. Towards kernelizing the classifier for hyperbolic data. *Frontiers of Computer Science*, 18(1):181301, August 2023. ISSN 2095-2236. doi: 10.1007/s11704-022-2457-y. URL https://doi.org/10.1007/s11704-022-2457-y.