Mixed-Curvature Decision Trees and Random Forests

Philippe Chlenski [1] **Quentin Chu** [1] **Itsik Pe'er** [1]

Abstract

We extend decision tree and random forest algorithms to product space manifolds: Cartesian products of Euclidean, hyperspherical, and hyperbolic manifolds. Such spaces have extremely expressive geometries capable of representing many arrangements of distances with low metric distortion. To date, all classifiers for product spaces fit a single linear decision boundary, and no regressor has been described. Our method enables a simple, expressive method for classification and regression in product manifolds. We demonstrate the superior accuracy of our tool compared to Euclidean methods operating in the ambient space or the tangent plane of the manifold across a range of constant-curvature and product manifolds. Code for our implementation and experiments is available at `https://github.com/pchlenski/embedders`.

1. Introduction

While much of machine learning focuses on data that is well-suited to Euclidean space, in certain situations non-Euclidean representations can better represent the underlying structure of the data. Typically, work on non-Euclidean representations tends to assume that data comes from a single manifold of constant curvature, such as a hyperspherical or hyperbolic space. However, even constant-curvature manifolds can fail to model certain kinds of structure.

As an alternative, Gu et al. (2018) proposed **product space manifolds**: a general, flexible, and expressive class of manifolds capable of capturing even more complex patterns in pairwise distances with much lower distortion than single manifolds achieve. Product spaces are simply Cartesian products of one or more constant-curvature **component manifolds**, with operations such as distance computation neatly factorizing across component manifolds. In the single-component case, product manifolds recover the geometry of their component manifold.

[1]Department of Computer Science, Columbia University, New York, USA. Correspondence to: Philippe Chlenski <pac@cs.columbia.edu>.

Product manifolds have found applications in diverse fields such as biology (McNeela et al., 2024) and knowledge graph representation (Wang et al., 2021).

However, the uptake of product manifold embeddings has been limited. This is partially due to a lack of tools for downstream inference tasks such as classification and regression on the basis of product manifold coordinates. To our knowledge, only Tabaghi et al. (2021) have ever described a product space classifier so far. As their classifier relies on a single linear decision boundary, it lacks the expressiveness of tools like decision trees and random forests.

Decision trees' inductive bias favors partitioning the space into convex decision regions using boundaries equidistant to the pair of points they separate. Since Euclidean decision trees violate convexity for some manifolds, we must modify them to suit these manifolds better. We propose such a method that works for all component manifolds, modify it further to work on product space manifolds, and benchmark both methods' effectiveness. For efficiency, we would also like to learn the entire tree in $O(ndt)$ time, where n is the number of points, d is the number of dimensions, and t is the maximum depth of the tree.

Our contributions. Concretely, we contribute:

1. A generalized algorithm for fitting decision trees to constant-curvature manifolds,
2. A novel algorithm for fitting decision trees on product space manifolds,
3. A generalized technique for sampling Gaussian mixtures in product space manifolds, and
4. A preliminary benchmark demonstrating the effectiveness of our component- and product-space trees over classical decision trees for synthetic datasets.

2. Preliminaries

2.1. Riemannian manifolds

We will begin by reviewing key details of hyperspheres, hyperboloids, and Euclidean spaces. For more details, readers can consult Do Carmo (1992).

Each space described is a **Riemannian manifold**, meaning that it is locally isomorphic to Euclidean space and equipped with a distance metric. The shortest paths be-

tween two points $\mathbf{u}$ and $\mathbf{v}$ on a manifold are referred to as **geodesics**. As all three spaces we consider have constant Gaussian curvature, we define simple closed-forms for **geodesic distances** in each of the following subsections in lieu of a more general discussion of geodesic distances in arbitrary Riemannian manifolds.

Any constant-curvature manifold $\mathcal{M}$ is parameterized by a **dimensionality** D and a **curvature** K. They can also all be considered embedded in an ambient space $\mathbb{R}^{D+1}$.

Finally, for each point $\mathbf{x} \in \mathcal{M}$, we can define the **tangent plane** at $\mathbf{x}$ as $T_{\mathbf{x}}\mathcal{M}$. The tangent plane at $\mathbf{x}$ is the space of all tangent vectors at $\mathbf{x}$.

2.1.1. EUCLIDEAN SPACE

A **Euclidean space** is equivalent $\mathbb{R}^D$:

$$\mathbb{E}^D = \mathbb{R}^D \tag{1}$$

First, we define the **inner product** (the dot product) used in Euclidean space:

$$\langle \mathbf{u}, \mathbf{v} \rangle = u_0 v_0 + u_1 v_1 + \ldots + u_2 v_2 \tag{2}$$

Additionally, we use $\|\mathbf{u}\| = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle}$. Then, we can define the **Euclidean distance function** (the ℓ_2 norm):

$$\delta_{\mathbb{E}}(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\| \tag{3}$$

2.1.2. HYPERSPHERICAL SPACE

We will describe hyperspherical space in terms of its coordinates in the ambient space. Hyperspherical space uses the same inner products as Euclidean space. The **hypersphere** is the set of points of constant Euclidean norm:

$$\mathbb{S}^{D,K} = \{\mathbf{x} \in \mathbb{R}^D \ : \ \|x\| = 1/K\} \tag{4}$$

We use the **hyperspherical distance function**:

$$\delta_{\mathbb{S}}(\mathbf{u}, \mathbf{v}) = \cos^{-1}(K^2 \langle \mathbf{u}, \mathbf{v} \rangle)/K. \tag{5}$$

2.1.3. HYPERBOLIC SPACE

We will describe the hyperbolic space from the perspective of the **hyperboloid model**. First, we need to define the ambient **Minkowski space**. This is a vector space equipped with the **Minkowski inner product**:

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{L}} = -u_0 v_0 + u_1 v_1 + \ldots + u_n v_n \tag{6}$$

Analogous to the Euclidean case, we let $\|\mathbf{u}\|_{\mathcal{L}} = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle_{\mathcal{L}}}$. The **hyperboloid** of dimension D and curvature $K < 0$, written $\mathbb{H}^{D,K}$, is a set of points with constant Minkowski norm:

$$\mathbb{H}^{D,K} = \{\mathbf{x} \in \mathbb{R}^D \ : \ \|\mathbf{x}\|_{\mathcal{L}} = 1/K, \ x_0 > 0\}, \tag{7}$$

Finally, this space uses a **hyperbolic distance function**:

$$\delta_{\mathbb{H}}(\mathbf{u}, \mathbf{v}) = -\cosh^{-1}(K^2 \langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{L}})/K \tag{8}$$

2.1.4. MIXED-CURVATURE PRODUCT SPACES

We reiterate the definition of product space manifolds, most of which comes from Gu et al. (2018). Following the convention of using $\prod_i \mathbf{X_i}$ to refer to the iterated *Cartesian* product over sets, we define a **product space manifold** as

$$\mathcal{P} = \prod_{i=1}^{n} \mathbb{S}^{s_i, K_i} \times \prod_{j=1}^{m} \mathbb{H}^{h_j, K'_j} \times \mathbb{E}^d \tag{9}$$

The total number of dimensions is $\sum_i s_i + \sum_j h_j + d$. Each individual manifold is called a **component manifold**. The decomposition of the product space into component manifolds is called the **signature**. Informally, the signature can be thought of as a list of dimensionalities and curvatures for each component manifold.

Distances in product manifolds decompose as the ℓ_2 norm of the distances in each of the component manifolds:

$$\delta_{\mathcal{P}}(\mathbf{u}, \mathbf{v}) = \sqrt{\sum_{\mathcal{M} \in \mathcal{P}} \delta_{\mathcal{M}}(\mathbf{u}_{\mathcal{M}}, \mathbf{v}_{\mathcal{M}})^2}, \tag{10}$$

where $\mathbf{u}_{\mathcal{M}}$ and $\mathbf{v}_{\mathcal{M}}$ denotes the restriction of $\mathbf{u}$ and $\mathbf{v}$ to their components in $\mathcal{M}$ and $\delta_{\mathcal{M}}$ refers the distance function appropriate to $\mathcal{M}$.

For $\mathbf{x} \in \mathcal{P}$, the tangent plane at $\mathbf{x}$ $T_{\mathbf{x}}\mathcal{P}$ is simply the concatenation of all component tangent planes:

$$T_{\mathbf{x}}\mathcal{P} = \prod_{\mathcal{M} \in \mathcal{P}} T_{\mathbf{x}_{\mathcal{M}}}\mathcal{M}. \tag{11}$$

We additionally define the origin of $\mathcal{P}$, μ_0, as the concatenation of the origins of each respective manifold. The origin is $(1/|K|, 0, \ldots)$ for $\mathbb{H}^K$ and $\mathbb{S}^K$, and $(0, 0, \ldots)$ for $\mathbb{E}$.

2.2. Decision trees

The **Classification and Regression Trees (CART)** (Breiman, 2017) algorithm fits decision trees to a set of labels $\mathbf{y}$. Specifically, it greedily selects a split at each set to partition the dataset in such a way as to maximize the **information gain**:

$$\text{IG}(\mathbf{y}) = C(\mathbf{y}) - \frac{|\mathbf{y}^+|}{|\mathbf{y}|} C(\mathbf{y}^+) - \frac{|\mathbf{y}^-|}{|\mathbf{y}|} C(\mathbf{y}^-) \tag{12}$$

In this case, $C(\cdot)$ is some sort of **impurity function** (in this paper, Gini impurity for classification, and variance for regression). Some splitting function $S(\cdot)$ is used to partition the *labels* $\mathbf{y}$ into two classes, $\mathbf{y}^+$ and $\mathbf{y}^-$; however, $S(\cdot)$ also partitions the *input space* (corresponding to some $\mathbf{X}$ that does not appear in Eq. 12) into **decision regions**. Classically, $S(\cdot)$ is a thresholding function and thus breaks the input space into box-like regions.

This algorithm is applied recursively to each of the decision regions until some stopping condition is met (for instance, information gain fails to hit some minimum threshold). The result is a fitted decision tree, $\mathcal{T}$, which can be used for inference. During inference, an unseen point $\mathbf{x}$ is passed through the decision tree until it reaches a leaf node corresponding to some decision region. For classification, the point is then assigned the majority label inside that region; for regression, it is assigned the mean value inside that region.

Finally, a **random forest** is an ensemble of decision trees that is typically trained on a subsample of the points and features in $\mathbf{X}$ (Breiman, 2001).

3. Mixed-curvature decision trees

For each component manifold, we adapt the method, first described in Chlenski et al. (2024), to fit decision trees using **homogenous hyperplanes**, i.e. hyperplanes that contain the origin of the ambient space. Such hyperplanes intersect any component manifold at **geodesic submanifolds**, meaning that if we let $\mathbb{P}$ be a plane, then $\mathcal{M} \cap \mathbb{P}$ contains all shortest paths between its own elements *according to* $\delta_{\mathcal{M}}$.

For any decision tree, we must transform the input $\mathbf{X}$ into a set of candidate hyperplanes. We consider a restricted set of homogenous hyperplanes whose normal vectors are nonzero in at most two dimensions: one dimension d, which varies, and one dimension (always the first dimension in our component manifolds, which we denote using index 0) called the **special dimension**. The special dimension is used in every decision boundary fitting.

Given a dimension d and a manifold $\mathcal{M}$, we first characterize each point as an angle:

$$\theta(\mathbf{x}, d) = \tan^{-1}(x_0 / x_d) \tag{13}$$

Note that, in our implementation, we use the PyTorch `arctan2` function to ensure that we can recover the full range of angles in $[0, 2\pi)$. This is essential for properly specifying decision boundaries in hyperspherical cases.

Once all angles are computed, we must compute **angular midpoints** such that we can find a hyperplane that intersects $\mathcal{M}$ in a location that is *geodesically equidistant to both of the points* considered. We provide angular midpoint formulas for each component manifold in the following sections.

Given a set of angular midpoints, we fit a decision tree using the usual dot-product reformulation of decision boundaries by greedily maximizing the information gain (as defined in Eq. 12) at each decision split until the maximum depth is reached, or other stopping criteria are met. The exact split function for a single point $\mathbf{x}$, a dimension index d, and an angle θ, is given as

$$S(\mathbf{x}, d, \theta) = \text{sign}(\sin(\theta) x_d - \cos(\theta) x_0). \tag{14}$$

The rest of the algorithm exactly follows the description in Section 2.2.

3.1. Euclidean decision trees

While homogenous hyperplanes *in* $\mathbb{R}^D$ trivially intersect $\mathbb{E}^D$ at geodesic submanifolds, these lack the expressiveness of an ambient-space formulation. Instead, we embed $\mathbb{E}^D$ in $\mathbb{R}^{D+1}$ by applying a trivial lift:

$$\phi : \mathbb{E}^D \to \mathbb{R}^{D+1},\ \phi(\mathbf{u}) = (1, \mathbf{u}). \tag{15}$$

For two points $\mathbf{u}, \mathbf{v} \in \mathbb{E}^D$, the midpoint angles in $\mathbb{E}^D$ are

$$m_{\mathbb{E}}(\mathbf{u}, \mathbf{v}) = \tan^{-1}(2/(u_d + v_d)). \tag{16}$$

While this presentation of Euclidean decision trees is unconventional, it is completely equivalent to thresholding in the basis dimensions. See Appendix C for the proof.

3.2. Hyperbolic decision trees

For two points $\mathbf{u}, \mathbf{v} \in \mathbb{H}^{D,K}$, we compute $\theta_{\mathbf{u}}$ and $\theta_{\mathbf{v}}$ according to Eq 13 and follow Chlenski et al. (2024) in computing the **hyperbolic midpoint angle** in $\mathbb{H}^{D,K}$ as:

$$V := \frac{\sin(2\theta_{\mathbf{u}} - 2\theta_v)}{2\sin(\theta_{\mathbf{u}} + \theta_{\mathbf{v}})\sin(\theta_{\mathbf{v}} - \theta_{\mathbf{u}})} \tag{17}$$

$$m_{\mathbb{H}}(\mathbf{u}, \mathbf{v}) = \begin{cases} \cot^{-1}(V - \sqrt{V^2 - 1}) & \text{if } \theta_{\mathbf{u}} + \theta_{\mathbf{v}} < \pi \\ \cot^{-1}(V + \sqrt{V^2 - 1}) & \text{otherwise} \end{cases} \tag{18}$$

3.3. Hyperspherical decision trees

The hyperspherical case is quite simple, except that unlike hyperbolic space and the "lifted" Euclidean space after applying Eq 15, we lack a natural choice of special dimension. We follow the convention of using the first dimension of the embedding space as the special dimension, which intuitively corresponds to fixing a "north pole" in the first dimension.

Given $\mathbf{u}, \mathbf{v} \in \mathbb{S}^{D,K}$, the **hyperspherical midpoint angle** is the arithmetic mean of θ_u and θ_v (computed using Eq 13):

$$m_{\mathbb{S}}(\mathbf{u}, \mathbf{v}) = (\theta_u + \theta_v)/2 \tag{19}$$

3.4. Product space algorithm

Intuitively, the only modification of a decision tree in a single component manifold is that we now iterate over all (non-special) dimensions of $\mathcal{P}$ and keep track of which special dimension must be used when computing angles for candidate hyperplanes. Complete pseudocode for this algorithm is in Appendix B.

Allowing for a single decision tree to span all components—rather than, for instance, an ensemble of decision trees each

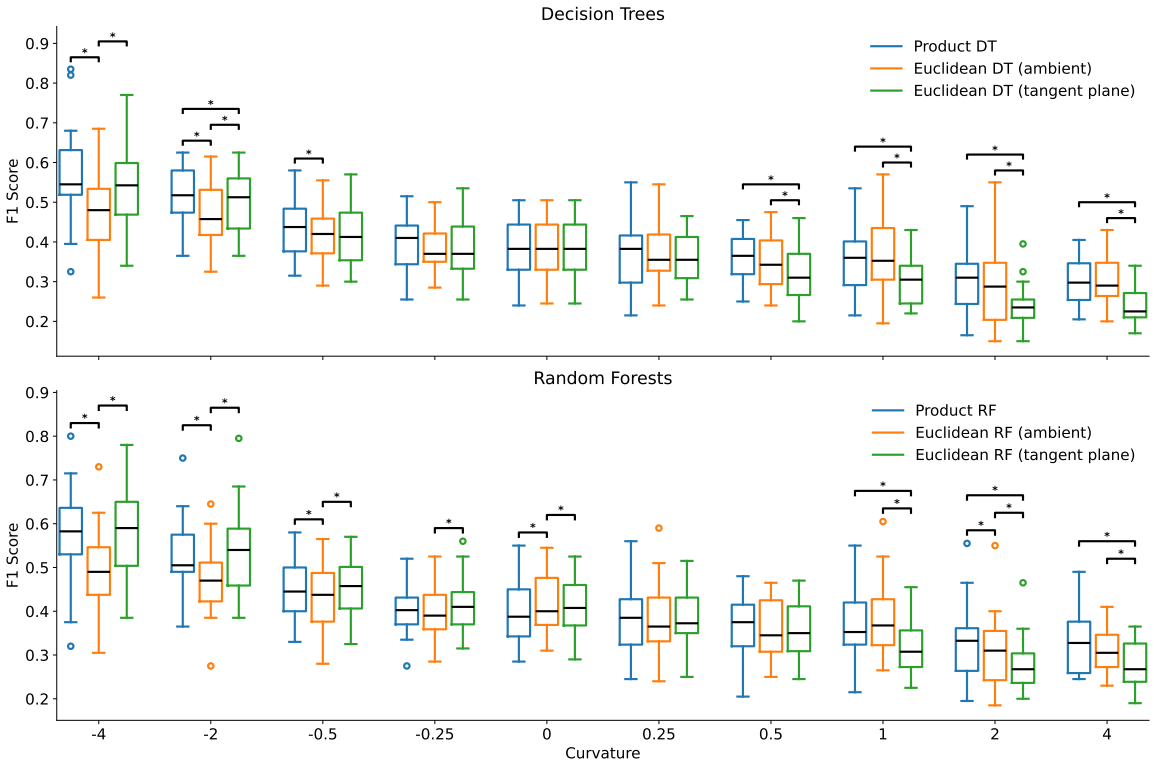

Figure 1. Benchmark comparison of decision trees (top) and random forests (bottom). We report micro-averaged F1 scores on a synthetic data classification task involving mixtures of 10 Gaussians in manifolds of varying constant curvatures K. We compare our method (**blue**), a Euclidean decision tree operating the latent space (**orange**), and a Euclidean decision tree operating in $T_{\mu_0}\mathcal{P}$ (**green**). Differences in performance increase with $|K|$. Notably, all methods are *identical* (not just comparable) for decision trees when $K = 0$. Statistical significance ($p < 0.05$) is marked with an asterisk.

operating in a single component—allows the model to discover which component manifolds are most relevant to a given task, and to allocate more capacity to those.

The regression and random forest variants in Section 2.2 can be applied unmodified to product space decision trees.

4. Benchmarks

4.1. Sampling Gaussian mixtures in $\mathcal{P}$

We extend the method developed by Nagano et al. (2019) for Gaussians in $\mathbb{H}$ to sample Gaussian mixtures in $\mathcal{P}$. Following Chlenski et al. (2024), we define Gaussian *mixtures* in $\mathbb{H}$ by sampling a set of means in $T_{\mu_0}\mathcal{P}$ and projecting them directly onto $\mathcal{P}$ via the exponential map. We add three further modifications:

1. We use the Wishart distribution (Wishart, 1928) to generate covariance matrices for Gaussian mixtures (Chatfield & Collins, 1980);
2. We rescale the covariance matrix according to the curvature to ensure the distribution of distances to the mean stays consistent across all submanifolds; and
3. To generate mixtures of gaussians in $\mathcal{P}$, we concatenate together embeddings from each component $\mathcal{M}$.

Full details of our sampling method are in Appendix A.

4.2. Baselines and parameters

We use scikit-learn (Pedregosa et al., 2011) decision trees and random forests as Euclidean classifiers operating in both the ambient space R^{D+1} and the tangent plane $T_{\mu_0}\mathcal{P}$ as baselines. Ambient space models use coordinates directly as features. These models have more degrees of freedom than product space or tangent plane models do because they can fit boundaries in any ambient dimension. For tangent plane models, we apply the logarithmic map at μ_0 to project points from $\mathcal{P}$ to $T_{\mu_0}\mathcal{P}$, then feed the resulting coordinates into our models.

All benchmarks were run on 1,000 points using an 80:20 train/test split. Decision trees use a maximum depth of 3, and random forests all consist of 12 decision trees trained on subsampled data.

Table 1. Benchmark results. We report 95% confidence intervals for classification accuracies on mixtures of four Gaussians over a representative sample of product manifolds. Each manifold has 10 total dimensions. Superscripts indicate statistical significance ($p < 0.05$): * for beating product spaces, † for beating Euclidean classifiers, and ‡ for beating tangent space classifiers.

Signature	Product DT	Euclidean DT	Tangent DT	Product RF	Euclidean RF	Tangent RF
$(\mathbb{H}^5)^2$	$\mathbf{97.6 \pm 1.0}^{\dagger\ddagger}$	92.8 ± 2.3	$96.1 \pm 1.4^{\dagger}$	$\mathbf{98.0 \pm 1.0}^{\dagger}$	94.0 ± 1.9	$97.3 \pm 1.2^{\dagger}$
$(S5)^2$	61.9 ± 3.5	60.9 ± 3.9	$\mathbf{62.5 \pm 3.5}$	64.1 ± 3.6	$\mathbf{64.6 \pm 3.4}$	64.3 ± 3.6
$\mathbb{H}^5 \times \mathbb{S}^5$	$\mathbf{97.6 \pm 1.0}^{\dagger\ddagger}$	92.8 ± 2.2	$96.2 \pm 1.3^{\dagger}$	$\mathbf{98.0 \pm 1.0}^{\dagger\ddagger}$	92.4 ± 2.2	$96.7 \pm 1.2^{\dagger}$
$(\mathbb{H}^2)^5$	$\mathbf{80.9 \pm 4.5}$	79.8 ± 4.3	80.3 ± 4.4	82.0 ± 4.1	81.2 ± 4.1	$\mathbf{82.6 \pm 4.2}^{\dagger}$
$(\mathbb{S}^2)^5$	$\mathbf{59.5 \pm 4.6}$	58.9 ± 4.6	57.9 ± 4.5	61.2 ± 4.2	$\mathbf{62.2 \pm 4.7}$	60.8 ± 4.5
$(\mathbb{H}^2)^2 \times \mathbb{E}^2 \times (\mathbb{S}^2)^2$	$\mathbf{82.0 \pm 4.1}^{\dagger}$	80.8 ± 4.3	81.8 ± 3.9	$\mathbf{82.7 \pm 4.0}^{\dagger}$	81.3 ± 4.2	82.1 ± 4.1

4.3. Component manifold benchmarks

In Figure 1, we compare our classifier and baselines on the task of classifying Gaussian mixtures *on a single component manifold* $\mathcal{M}$ with K ranging from -4 to 4. For each curvature, we sampled 20 mixtures of 10 Gaussians.

The negative-curvature results extend earlier findings by Chlenski et al. (2024) to the full range of negative curvatures; the positive-curvature results are novel. All three decision trees perform *identically* for curvature 0, corroborating their equivalence in the Euclidean case. A formal proof of this equivalence can be found in Appendix C.

4.4. Product manifold benchmarks

In Table 1, we compare our method to baseline models across multiple signatures. For each signature, we sampled 10 mixtures of four Gaussians. We follow Gu et al. (2018) in our choice of the six signatures tested, but opt to use a Gaussian mixture rather than a graph embedding for our benchmark. In most cases, our method beats the baselines.

5. Discussion

5.1. Contributions

We present strong preliminary evidence in favor of mixed-curvature decision trees and random forests. In particular, we motivate and describe our entire algorithm and demonstrate its effectiveness for classifying Gaussian mixtures in both constant- and mixed-curvature manifolds.

5.2. Limitations

Distance functions must change during the transition from negative to zero to positive curvature, creating an inelegant discontinuity. Moreover, though the limit of the radius as K approaches ∞ is also ∞, we arbitrarily lift our Euclidean embeddings to $x_0 = 1$ instead. A continuous unification—such as the projective geometry approach described in Skopek et al. (2020)—as well as more thorough benchmarks would substantially improve this work.

5.3. Related work

Machine learning in product spaces. In product spaces, Tabaghi et al. (2021) describe linear classifiers, including perceptron and support vector machines; Tabaghi et al. (2024) adapt principal component analysis; and Cho et al. (2023) generalize Transformer architectures.

Learning with product space-derived features. Tsagkrasoulis & Montana (2017) train random forest classifiers on distance matrices from arbitrary manifolds, e.g. product spaces. Both Sun et al. (2021) and Borde et al. (2023) use product manifold manifolds to compute rich similarity measures, which are fed into a classifier as features.

Hyperbolic random forests. Our method is inspired by recent work by Chlenski et al. (2024) and Doorenbos et al. (2023) extending decision tree and random forest algorithms to hyperbolic space. In particular, the use of homogeneous hyperplanes as decision boundaries is a synthesis of the ideas in Chlenski et al. (2024) and Tabaghi et al. (2021).

Applications of product spaces. Product space manifolds are popular for embedding knowledge graphs (Wang et al., 2021; Li et al., 2024; Nguyen-Van et al., 2023). In biology, they have been used to represent pathway graphs (McNeela et al., 2024), cryo-EM images (Zhang et al., 2021), and single-cell transcriptomic profiles (Tabaghi et al., 2021).

5.4. Future work

Future work will prioritize more thorough benchmarking of our methods: we will compare our method to the product space perceptron and SVM in Tabaghi et al. (2021), and we will classify real data embedded using the methods described in Gu et al. (2018) (pairwise graph distances) and Skopek et al. (2020). We are especially interested in high-quality regression benchmarks and applications to complex biological data such as spatial transcriptomics.

Acknowledgements

This work was funded by NSF GRFP grant DGE-2036197.

References

Borde, H. S. d. O., Kazi, A., Barbero, F., and Liò, P. Latent Graph Inference using Product Manifolds, June 2023. URL http://arxiv.org/abs/2211.16199. arXiv:2211.16199 [cs].

Breiman, L. 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.

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

Chatfield, C. and Collins, A. J. *Introduction to Multivariate Analysis*. Springer US, Boston, MA, 1980. ISBN 978-0-412-16030-1 978-1-4899-3184-9. doi: 10.1007/978-1-4899-3184-9. URL http://link.springer.com/10.1007/978-1-4899-3184-9.

Chlenski, P., Turok, E., Moretti, A., and Pe'er, I. Fast hyperboloid decision tree algorithms, March 2024. URL http://arxiv.org/abs/2310.13841. arXiv:2310.13841 [cs].

Cho, S., Cho, S., Park, S., Lee, H., Lee, H., and Lee, M. Curve Your Attention: Mixed-Curvature Transformers for Graph Representation Learning, September 2023. URL http://arxiv.org/abs/2309.04082. arXiv:2309.04082 [cs].

Do Carmo, M. *Riemannian Geometry*. Springer US, 1992. URL https://link.springer.com/book/9780817634902.

Doorenbos, L., Márquez-Neila, P., Sznitman, R., and Mettes, P. Hyperbolic Random Forests, August 2023. URL http://arxiv.org/abs/2308.13279. arXiv:2308.13279 [cs].

Gu, A., Sala, F., Gunel, B., and Ré, C. Learning Mixed-Curvature Representations in Product Spaces. September 2018. URL https://openreview.net/forum?id=HJxeWnCcF7.

Li, K., Zhang, Y., Zhu, J., Li, X., and Wang, X. Multi-space interaction learning for disentangled knowledge-aware recommendation. *Expert Systems with Applications*, 254:124458, November 2024. ISSN 0957-4174. doi: 10.1016/j.eswa.2024.124458. URL https://www.sciencedirect.com/science/article/pii/S0957417424013241.

McNeela, D., Sala, F., and Gitter, A. Product Manifold Representations for Learning on Biological Pathways, January 2024. URL http://arxiv.org/abs/2401.15478. arXiv:2401.15478 [cs, q-bio].

Nagano, Y., Yamaguchi, S., Fujita, Y., and Koyama, M. 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].

Nguyen-Van, T., Le, D. D., and Ta, T.-A. Improving Heterogeneous Graph Learning with Weighted Mixed-Curvature Product Manifold, July 2023. URL http://arxiv.org/abs/2307.04514. arXiv:2307.04514 [cs].

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011. ISSN 1533-7928. URL http://jmlr.org/papers/v12/pedregosa11a.html.

Skopek, O., Ganea, O.-E., and Bécigneul, G. Mixed-curvature Variational Autoencoders, February 2020. URL http://arxiv.org/abs/1911.08411. arXiv:1911.08411 [cs, stat].

Sun, L., Zhang, Z., Ye, J., Peng, H., Zhang, J., Su, S., and Yu, P. S. A Self-supervised Mixed-curvature Graph Neural Network, December 2021. URL http://arxiv.org/abs/2112.05393. arXiv:2112.05393 [cs].

Tabaghi, P., Pan, C., Chien, E., Peng, J., and Milenkovic, O. Linear Classifiers in Product Space Forms, February 2021. URL http://arxiv.org/abs/2102.10204. arXiv:2102.10204 [cs, stat] version: 1.

Tabaghi, P., Khanzadeh, M., Wang, Y., and Mirarab, S. Principal Component Analysis in Space Forms, July 2024. URL http://arxiv.org/abs/2301.02750. arXiv:2301.02750 [cs, eess, math, stat].

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

Wang, S., Wei, X., Nogueira dos Santos, C. N., Wang, Z., Nallapati, R., Arnold, A., Xiang, B., Yu, P. S., and Cruz, I. F. Mixed-Curvature Multi-Relational Graph Neural Network for Knowledge Graph Completion. In *Proceedings of the Web Conference 2021*, WWW '21,

pp. 1761–1771, New York, NY, USA, June 2021. Association for Computing Machinery. ISBN 978-1-4503-8312-7. doi: 10.1145/3442381.3450118. URL https://doi.org/10.1145/3442381.3450118.

Wishart, J. The generalised product moment distribution in samples from a normal multivariate population. *Biometrika*, 20A(1-2):32–52, December 1928. ISSN 0006-3444. doi: 10.1093/biomet/20A.1-2.32. URL https://doi.org/10.1093/biomet/20A.1-2.32.

Zhang, S., Moscovich, A., and Singer, A. Product Manifold Learning. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pp. 3241–3249. PMLR, March 2021. URL https://proceedings.mlr.press/v130/zhang21j.html. ISSN: 2640-3498.

A. Gaussian mixture details

A.1. Overall structure

The structure of our sampling algorithm is as follows. Note that, rather than letting $\mathcal{M}$ be a manifold of arbitrary curvature, we force its curvature to be one of $\{-1, 0, 1\}$ for implementation reasons. This necessitates rescaling steps, which take place in Equations 26, 30, and 36. The result is equivalent to performing the equivalent steps, without rescaling, on a manifold of the proper curvature.

1. Generate $\mathbf{c}$, a vector that divides m samples into n clusters:

$$\mathbf{p_{raw}} = \langle p_0, p_1, \ldots, p_{n-1} \rangle \tag{20}$$
$$p_i \sim \text{Uniform}(0, 1) \tag{21}$$
$$\mathbf{p_{norm}} = \frac{\mathbf{p_{raw}}}{\sum_{i=0}^{n-1} p_i} \tag{22}$$
$$\mathbf{c} = \langle c_0, c_1, \ldots c_{m-1} \rangle \tag{23}$$
$$c_i \sim \text{Categorical}(n, \mathbf{p_{norm}}) \tag{24}$$

2. Sample $\mathbf{M_{euc}}$, an $n \times D$ matrix of n class means:

$$\mathbf{M_{euc}} = \langle \mathbf{m_0}, \mathbf{m_1}, \ldots, \mathbf{m_{n-1}} \rangle^T \tag{25}$$
$$\mathbf{m_i} \sim \mathcal{N}(0, \sqrt{K}\mathbf{I}). \tag{26}$$

3. Move $\mathbf{M_{euc}}$ into $T_0\mathcal{M}$, the tangent plane at the origin of $\mathcal{M}$, by applying $\psi : \mathbf{x} \to (0, \mathbf{x})$ per-row to $\mathbf{M_{euc}}$:

$$\mathbf{M_{tan}} = \langle \psi(\mathbf{m_0}), \psi(\mathbf{m_1}), \ldots \psi(\mathbf{m_{n-1}}) \rangle^T, \tag{27}$$
$$\psi : \mathbb{R}^D \to \mathbb{R}^{D+1}, \ \mathbf{x} \to \langle 0, \mathbf{x} \rangle. \tag{28}$$

4. Project $\mathbf{M_{tan}}$ onto $\mathcal{M}$ using the exponential map from $T_0\mathcal{M}$ to $\mathbf{M_{tan}}$:

$$\mathbf{M} = \exp_0(\mathbf{M_{tan}}). \tag{29}$$

5. For $0 \leq i < n$, sample a corresponding covariance matrix. Here, σ is a variance scale parameter than can be set:

$$\mathbf{\Sigma_i} \sim \text{Wishart}(\sigma\sqrt{K}\mathbf{I}, D) \tag{30}$$

6. For $0 \leq j < m$, sample $\mathbf{X_{euc}}$, a matrix of m points according to their clusters' covariance matrices:

$$\mathbf{X_{euc}} = \langle \mathbf{x_0}, \mathbf{x_1}, \ldots \mathbf{x_{m-1}} \rangle^T \tag{31}$$
$$x_j \sim \mathcal{N}(0, \mathbf{\Sigma_{c_j}}). \tag{32}$$

7. Apply $\psi(\cdot)$ from Eq 28 to each $\mathbf{x_j}$ to move it into $T_0\mathcal{M}$:

$$\mathbf{X_{tan}} = \langle \psi(\mathbf{x_0}), \psi(\mathbf{x_1}), \ldots \psi(\mathbf{x_{m-1}}) \rangle^T. \tag{33}$$

8. For each row in $\mathbf{X_{tan}}$, apply parallel transport from $T_0\mathcal{M}$ to its class mean:

$$\mathbf{X_{PT}} = \langle \mathbf{x_{0,\mu}}, \mathbf{x_{1,\mu}}, \ldots, \mathbf{x_{m-1,\mu}} \rangle \tag{34}$$
$$\mathbf{x_{j,\mu}} = PT_{0 \to \mathbf{m_{c_j}}}(\mathbf{x_j}) \tag{35}$$

9. Use the exponential map at $T_\mu\mathcal{M}$ to move the points onto the manifold:

$$\mathbf{X_{\mathcal{M}}} = \langle \mathbf{x_{0,\mathcal{M}}}, \mathbf{x_{1,\mathcal{M}}}, \ldots, \mathbf{x_{m-1,\mathcal{M}}} \rangle \tag{36}$$
$$\mathbf{x_{j,\mathcal{M}}} = \frac{\exp_{\mathbf{m_{c_j}}}(\mathbf{x_{j,\mu}})}{\sqrt{K}} \tag{37}$$

10. Repeat steps 2–9 for as many manifolds as desired; produce a final embedding by concatenating all component embeddings column-wise:

$$\mathbf{X} = \langle \mathbf{X}_{\mathcal{M}_0}, \mathbf{X}_{\mathcal{M}_1}, \ldots \mathbf{X}_{\mathcal{M}_p} \rangle \tag{38}$$

A.2. Equations for manifold operations

First, we provide the forms of the **parallel transport** operation in hyperbolic, hyperspherical, and Euclidean spaces:

$$PT^{\mathbb{H}}_{\nu\to\mu}(\mathbf{v}) = \mathbf{v} + \frac{\langle \mu - \alpha\nu, \nu\rangle_{\mathcal{L}}}{\alpha+1}(\nu+\mu) \tag{39}$$

$$\alpha = -\langle \nu, \mu\rangle_{\mathcal{L}} \tag{40}$$

$$PT^{\mathbb{S}}_{\nu\to\mu}(\mathbf{v}) = \mathbf{v}\cos(d) + \frac{\sin(d)}{d}(\mu - \cos(d)\nu) \tag{41}$$

$$d = \cos^{-1}(\nu\cdot\mu) \tag{42}$$

$$PT^{\mathbb{E}}_{\nu\to\mu}(\mathbf{v}) = \mathbf{v} + \mu - \nu. \tag{43}$$

The **exponential map** is defined as follows in each of the three spaces:

$$\exp_{\mu}(\mathbf{u}) = \cosh(\|\mathbf{u}\|_{\mathcal{L}})\mu + \sinh(\|\mathbf{u}\|_{\mathcal{L}})\frac{\mathbf{u}}{\|\mathbf{u}\|_{\mathcal{L}}} \tag{44}$$

$$\exp_{\mu}(\mathbf{u}) = \cos(\|\mathbf{u}\|)\mu + \sin(\|\mathbf{u}\|)\frac{\mathbf{u}}{\|\mathbf{u}\|} \tag{45}$$

$$\exp_{\mu}(\mathbf{u}) = \mathbf{u}. \tag{46}$$

A.3. Relationship to other work

Nagano et al. (2019) developed the overall technique used for a single cluster and a single manifold, i.e. steps 6–9. Chlenski et al. (2024) modified this method to work for mixtures of Gaussians in $\mathbb{H}^{d,1}$, and deployed it for $d \in 2, 4, 8, 16$. This corresponds to steps 1–5 of our procedure (although note that our covariance matrices are sampled differently in step 5). Thus, our contribution is simply to add step 10, modify step 5 to use the Wishart distribution, and to add curvature-related scaling factors in Equations 26, 30, and 36.

Additionally, to the best of our knowledge we are the first to apply this to *hyperspherical* manifolds, for which the **von Mises-Fisher (VMF) distribution** is typically preferred. We do not argue for the superiority of our approach over the VMF distribution in general; however, we prefer to use ours for these benchmarks, as it allows us to draw simpler parallels between manifolds of different curvatures.

B. Product space decision tree pseudocode

Algorithm 1 Product Space Decision Tree

Procedure FIT:
  $\mathcal{P}$ (signature of) product manifold
  $\mathbf{X}$ data points
  $\mathbf{y}$ target labels
Initialize:
  $\mathcal{T}$ an empty tree
return FITTREE$(\mathbf{X}, \mathbf{y}, 0)$

Procedure FITTREE:
  $\mathbf{X}$ data points
  $\mathbf{y}$ target labels
  t current depth of the tree.
Initialize:
  d_{best} dimension of best split,
  θ_{best} angle of best split,
  IG_{best} information gain of best split.
for each $d \in \mathbf{D}'$ **do**
  $\mathcal{M} \leftarrow$ component manifold for dimension d
  $\Theta \leftarrow$ GETCANDIDATES$(\mathcal{M}, \mathbf{X}, d)$
  for each candidate $\theta \in \Theta$ **do**
    Partition $\mathbf{X}, \mathbf{y}$ into $\mathbf{X}^+, \mathbf{X}^-, \mathbf{y}^+, \mathbf{y}^-$ via Eq. 14.
    Apply Eq. 12 on $\mathbf{y}^+, \mathbf{y}^-$ to compute IG_{current}
    if $IG_{\text{current}} > IG_{\text{best}}$ **then**
      $d_{\text{best}},\ \theta_{\text{best}},\ IG_{\text{best}} \leftarrow d,\ \theta,\ IG_{\text{current}}$
    end if
  end for
end for
if no valid split was found **then**
  return $\mathcal{N}$, a new **leaf node** with $\mathbf{y}$ probabilities.
else
  Create $\mathcal{N}$, a decision node with d_{best} and θ_{best}
  $\mathcal{N}_L \leftarrow$ FITTREE$(\mathbf{X}^-,\ \mathbf{y}^-,\ t+1)$
  $\mathcal{N}_R \leftarrow$ FITTREE$(\mathbf{X}^+,\ \mathbf{y}^+,\ t+1)$
  return $\mathcal{N}$ with left child $\mathcal{N}_L$ and right child $\mathcal{N}_R$
end if

Procedure GETCANDIDATES:
  $\mathcal{M}$ A component manifold
  $\mathbf{X}$ A dataset of points in $\mathcal{M}$
  d A dimension index
if d is the special dimension **then**
  return empty array []
end if
$\Theta \leftarrow$ Angles of $\mathbf{X}$ via Eq. 13
$\Theta \leftarrow$ sort and deduplicate Θ
return [θ_m for $\theta_i, \theta_{i+1} \in \Theta$ via Eq. 16, 17, or 19] (depending on curvature of $\mathcal{M}$).

C. Proof of equivalence for Euclidean case

A classical CART tree splits data points according to whether their value in a given dimension is greater than or less than some threshold value t. Midpoints are simple arithmetic means. This can be written as:

$$S'(\mathbf{x}, d, t) = \begin{cases} 1 \text{ if } x_d > t, \\ 0 \text{ otherwise.} \end{cases} \tag{47}$$

$$m_{DT}(\mathbf{u}, \mathbf{v}) = \frac{u_d + v_d}{2}. \tag{48}$$

In our transformed decision tree, we lift the data points by applying $\phi : \mathbf{x} \to (1, \mathbf{x})$ and then check which side of an axis-inclined hyperplane they fall on. The splitting function is based on the angle θ of inclination with respect to the $(0, d)$ plane, i.e., $\langle 1, x_d \rangle$. Our midpoints are computed to ensure equidistance in the original manifold:

$$S(\mathbf{x}, d, \theta) = \text{sign}(\sin(\theta)x_d - \cos(\theta)x_0) \tag{49}$$

$$m_{\mathbb{E}}(\mathbf{u}, \mathbf{v}) = \tan^{-1}\left(\frac{2}{u_d + v_d}\right) \tag{50}$$

To demonstrate the equivalence of the classical decision tree formulation to our transformed algorithm in $\mathbb{E}$, we will show that Equation 47 is equivalent to Equation 49 and Equation 48 is equivalent to Equation 50 under

$$\theta = \cot^{-1}(t). \tag{51}$$

C.1. Equivalence of Splits

First, we show that Equations 47 and 49 are equivalent, assuming $t \neq 0$:

$$\begin{aligned} S(\mathbf{x}, d, \theta) = \text{sign}(\sin(\theta)x_d - \cos(\theta)x_0) = 1 & \quad (52) \\ \iff \sin(\theta)x_d - \cos(\theta) > 0 & \quad (53) \\ \iff \frac{\sin(\theta)}{\cos(\theta)}x_d = \tan(\theta)x_d > 1 & \quad (54) \\ \iff x_d/t > 1 & \quad (55) \\ \iff x_d > t & \quad (56) \\ \iff S'(\mathbf{x}, d, t) = 1 & \quad (57) \end{aligned}$$

C.2. Equivalence of midpoints

Now, we show that Equations 48 and 50 are equivalent:

$$\begin{aligned} \cot^{-1}(m_{DT}(\mathbf{u}, \mathbf{v})) &= \cot^{-1}\left(\frac{u_d + v_d}{2}\right) & (58) \\ &= \tan^{-1}\left(\frac{2}{u_d - v_d}\right) & (59) \\ &= m_{\mathbb{E}}(\mathbf{u}, \mathbf{v}) & (60) \end{aligned}$$