Hyperbolic Continuous Structural Entropy For Hierarchical Clustering

Guangjie Zeng[1], Hao Peng[1], Angsheng Li[1], Li Sun[2], Chunyang Liu[3], Shengze Li[4], Yicheng Pan[1*], Philip S. Yu[5]

[1]School of Computer Science and Engineering, Beihang University, China
[2]School of Control and Computer Engineering, North China Electric Power University, China
[3]Didi Chuxing, China
[4]Academy of Military Science, China
[5]Computer Science, University of Illinois Chicago, USA
{zengguangjie, penghao, angsheng, yichengp}@buaa.edu.cn, ccesunli@ncepu.edu
liuchunyang@didiglobal.com, lsz86591989@163.com, psyu@uic.edu

Abstract

Hierarchical clustering is a fundamental machine-learning technique for grouping data points into dendrograms. However, existing hierarchical clustering methods encounter two primary challenges: 1) Most methods specify dendrograms without a global objective. 2) Graph-based methods often neglect the significance of graph structure, optimizing objectives on complete or static predefined graphs. In this work, we propose **Hyp**erbolic **C**ontinuous **S**tructural **E**ntropy neural networks, namely HypCSE, for structure-enhanced continuous hierarchical clustering. Our key idea is to map data points in the hyperbolic space and minimize the relaxed continuous structural entropy (SE) on structure-enhanced graphs. Specifically, we encode graph vertices in hyperbolic space using hyperbolic graph neural networks and minimize approximate SE defined on graph embeddings. To make the SE objective differentiable for optimization, we reformulate it into a function using the lowest common ancestor (LCA) on trees and then relax it into continuous SE (CSE) by the analogy of hyperbolic graph embeddings and partitioning trees. To ensure a graph structure that effectively captures the hierarchy of data points for CSE calculation, we employ a graph structure learning (GSL) strategy that updates the graph structure during training. Extensive experiments on seven datasets demonstrate the superior performance of HypCSE.

Introduction

Hierarchical clustering is a classic unsupervised machine-learning technique that groups data points into nested clusters organized as a cluster tree known as a dendrogram (Ran et al. 2023). Unlike partitioning clustering, which divides data points into a predefined number of clusters, hierarchical clustering captures relationships between data points at a finer granularity through nested clusters, without requiring the number of clusters to be specified in advance. It has broad applications including image analysis (Yan et al. 2021), bioinformatics (Chen and Li 2023), and medicine (Ciaramella, Nardone, and Staiano 2020).

Despite their success, conventional hierarchical clustering algorithms specify dendrograms procedurally (Chierchia and Perret 2019). Among them, agglomerative methods (Gower and Ross 1969) iteratively merge closest cluster pairs, while divisive methods (Moseley and Wang 2017) iteratively split clusters by flat clustering. Thus, they encounter **Challenge 1:** most of them lack a global objective and often lead to suboptimal dendrograms. Recently, Dasgupta's cost (Dasgupta 2016) and its extensions (Cohen-Addad et al. 2019; Wang and Wang 2020) have been proposed to evaluate dendrogram quality globally. From an information-theoretic perspective, SE (Li and Pan 2016; Li 2024) measures dendrogram quality by quantifying the minimum length of random walks on graphs. These cost functions facilitate the analysis of algorithms and the comparison of dendrograms.

Based on the optimization algorithms for cost functions they adopted, hierarchical clustering methods can be divided into discrete and continuous methods. Continuous methods (Monath et al. 2017, 2019; Chierchia and Perret 2019; Chami et al. 2020; Zügner et al. 2022) relax a given cost function to be differentiable and perform optimization using gradient-based techniques. Their chosen cost functions include Dasgupta's cost (Dasgupta 2016) and Tree-Sampling-Divergence (Charpentier and Bonald 2019), both of which are defined on graphs. Compared to discrete methods, continuous methods offer flexibility since they can be integrated into commonly used end-to-end learning pipelines. However, existing continuous methods optimize objectives on complete graphs (Chami et al. 2020) or predefined graphs (Chierchia and Perret 2019; Zügner et al. 2022). Complete graphs can be cluttered with trivial edges from data noise, while predefined graphs are static and unable to learn from data features during training. As a result, they encounter **Challenge 2:** they may not fully utilize data features or capture hierarchies by optimizing objectives on static graphs.

To address the aforementioned challenges, we introduce continuous SE (CSE) in hyperbolic space for hierarchical clustering. CSE is defined on a graph and its hyperbolic space embedding, measuring the quality of the graph embedding in capturing the graph's hierarchy. We formulate the SE objective as the weighted average of LCAs' volumes of graph vertices. In hyperbolic space, graph embeddings can be analogous to partitioning trees (Chami et al. 2020). We quantify the CSE of graph embedding by approximating the

*Corresponding author.

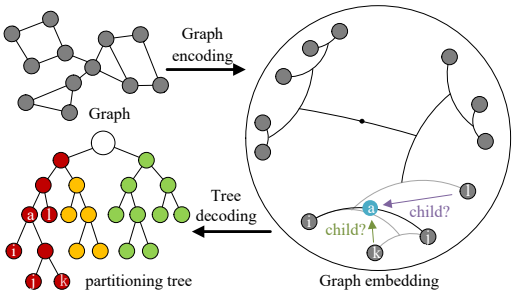


Figure 1: HypCSE overview. Graphs are encoded as hyperbolic embeddings by minimizing CSE. Partitioning trees are decoded from embeddings for hierarchical clustering.

volumes of LCAs on the analogous partitioning tree. Unlike DSI in LSEnet (Sun et al. 2024), which relaxes SE by level-wise assignment for flat partitioning clustering, our CSE relaxes SE by analogizing hyperbolic graph embeddings and binary partitioning trees for hierarchical clustering. Next, we optimize the graph embedding for hierarchical clustering by minimizing the CSE, as depicted in Figure 1. Given a constructed graph from input data, we encode it as a graph embedding in hyperbolic space and optimize the corresponding CSE via gradient descent. The optimized graph embedding is then decoded as a binary partitioning tree, serving as the hierarchical clustering result. Furthermore, we design a Hyperbolic CSE Neural Network (HypCSE) for CSE optimization on adaptive graphs updated during training. To avoid the use of predefined, inappropriate graphs, we employ a GSL strategy to learn more informative graphs. We construct an anchor graph as the anchor view and learn a new graph as the learner view, applying contrastive learning to guide the graph learner. The anchor graph is updated according to the learner graph through bootstrapping. This strategy helps learn anchor graphs with clear hierarchies and high discrimination among classes for CSE optimization.

The main contributions are summarized as follows: (1) We devise CSE by relaxing discrete SE to measure the quality of graph embeddings in hyperbolic space for capturing hierarchy. (2) We propose HypCSE via CSE for hierarchical clustering, where a GSL strategy is adopted to learn better graphs for CSE optimization. (3) Extensive experiments on 7 standard datasets demonstrate the superiority of HypCSE.

Preliminaries

Graph-Based Hierarchical Clustering

For a dataset $\mathbf{X}$ with n data points, graph G representing pairwise similarities (or dissimilarities) is denoted as $G = (V, E, \mathbf{W})$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices corresponding to data points in $\mathbf{X}$, E is the set of edges, and $\mathbf{W}$ consists of edge weights measuring pairwise similarities (or dissimilarities) between data points. A hierarchical clustering algorithm generates a dendrogram from $\mathbf{X}$ along with G. This dendrogram is an unweighted rooted tree $\mathcal{T}$ with n leaves corresponding n data points and internal nodes corresponding to nested clusters. For two leaves $\mathcal{T}_i$ and $\mathcal{T}_j$ in $\mathcal{T}$, their LCA is denoted as $\mathcal{T}_i \vee \mathcal{T}_j$.

Hyperbolic Space

Hyperbolic space is a space with negative curvature that has advantages in modeling hierarchical structure compared to flat Euclidean space (Peng et al. 2021; Sun et al. 2024, 2025b,a). Among the several isomorphic models of hyperbolic space, two commonly used ones are the Lorentz model and the Poincaré model. Specifically, a d-dimensional Lorentz model $\mathbb{L}^{\kappa,d}$ with curvature κ represents a manifold embeded in the $d+1$ dimensional Minkowski space, defined as $\mathbb{L}^{\kappa,d} = \{\mathbf{x} \in \mathbb{R}^{d+1} : \langle \mathbf{x}, \mathbf{x} \rangle_{\mathbb{L}} = \frac{1}{\kappa}\}$, where $\langle , \rangle_{\mathbb{L}}$ is the Minkowski inner product defined as $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbb{L}} = \mathbf{x}^T \mathbf{R}^{\mathbb{L}} \mathbf{y}$, $\mathbf{R}^{\mathbb{L}} \in \mathbb{R}^{(d+1)\times(d+1)}$ is a diagonal matrix with entries of 1s except for the first one being -1. Lorentz distance between points $\mathbf{x}$ and $\mathbf{y}$ is defined as $d_{\mathbb{L}} \operatorname{arcosh}(-\langle \mathbf{x}, \mathbf{y} \rangle)$. Lorentz norm of $\mathbf{x}$ is defined as $\|\mathbf{x}\|_{\mathbb{L}} = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle_{\mathbb{L}}}$. A d-dimensional Poincaré model $\mathbb{B}^{\kappa,d}$ with curvature κ is defined as $\mathbb{B}^{\kappa,d} = \{\mathbf{u} \in \mathbb{R}^d : \|\mathbf{u}\| < \frac{1}{\kappa}\}$, which can be given by projecting each point of Lorentz model $\mathbb{L}^{\kappa,d}$ onto the hyperplane $\mathbf{x}^{0,d} = 0$. For a point $\mathbf{x} = (x_0, x_1, \ldots, x_d) \in \mathbb{R}^{d+1}$ in the Lorentz model, the transformed point $\mathbf{u}$ in the Poincaré model is given by $\mathbf{u} = \frac{(x_1, \ldots, x_d)}{1+x_0}$.

Methodology

Definition 1 (Structural Entropy ((Li and Pan 2016))). Given an undirected weighted graph $G = (V, E, \mathbf{W})$ and an associated rooted tree $\mathcal{T}$, the SE of G on $\mathcal{T}$ is defined as

$$\mathcal{H}^{\mathcal{T}}(G) = \sum_{\alpha \in \mathcal{T}, \alpha \neq \lambda} \mathcal{H}^{\mathcal{T}}(G; \alpha) = \sum_{\alpha \in \mathcal{T}, \alpha \neq \lambda} -\frac{g_\alpha}{\mathcal{V}_G} \log_2 \frac{\mathcal{V}_\alpha}{\mathcal{V}_{\alpha^-}}, \tag{1}$$

where $\mathcal{H}^{\mathcal{T}}(G; \alpha)$ is the SE assigned to an non-root node α in $\mathcal{T}$. The rooted partitioning tree $\mathcal{T}$ of G forms a hierarchical clustering of data points. Each tree node $\alpha \in \mathcal{T}$ is associated with a vertex set T_α, where the root node λ is associated with $T_\lambda = V$ and each leaf node ν is associated with T_ν containing only one vertex in V. For each non-leaf node $\alpha \in \mathcal{T}$, the child nodes of α are associated with disjoint vertex subsets, whose union is T_α. The parent node of α is denoted as α^-. The notation g_α is the cut, i.e., the sum of graph edge weights with exactly one endpoint in T_α. Notations $\mathcal{V}_\alpha$ and $\mathcal{V}_G$ are the volumes, i.e., the sum of node degrees in T_α and T_G, respectively. The SE of graph G is the minimum one among all possible partitioning trees, which is defined as

$$\mathcal{H}(G) = \min_{\mathcal{T}} \mathcal{H}^{\mathcal{T}}(G), \quad \mathcal{T}^* = \arg_{\mathcal{T}} \min \mathcal{H}^{\mathcal{T}}(G), \tag{2}$$

where $\mathcal{T}^*$ is the optimal partitioning tree that best eliminates uncertainty in G by characterizing the hierarchical topology.

Lemma 2 (Minimum Structural Entropy ((Zhang, Wang, and Li 2021))). *Given an undirected weighted graph G, a binary partitioning tree $\mathcal{T}^*$ of the minimum structural entropy exists.*

Lemma 2 reveals that we can find the minimum SE of G by traversing all possible binary partitioning trees.

Lemma 3 (Connection to Graph-Based Clustering-Appendix A.1). *Given an weighted undirected graph $G = (V, E, \mathbf{W})$, $\rho^{\mathcal{T}}(G) = \mathcal{H}^{\mathcal{T}}(G)/\mathcal{H}^1(G)$ is the normalized*

SE of G on any possible rooted partitioning tree $\mathcal{T}$, and $\Phi(G) = \min_{S\subseteq V} \frac{\text{cut}(S)}{\min\{\mathcal{V}_S, \mathcal{V}_{V\setminus S}\}}$ is the graph conductance, the following inequality holds:

$$\rho^{\mathcal{T}}(G) \geq \Phi(G). \tag{3}$$

The proof is given in Appendix A. We note that graph conductance is a well-defined metric for assessing clustering quality on graphs and has been proven effective for graph-based hierarchical clustering (Cheng et al. 2006). Since the one-dimensional SE $\mathcal{H}^1(G)$ is a constant for a given G, and it serves as an upper bound for graph conductance (Lemma 3), minimizing SE in Definition (1) is likely to also reduce graph conductance. Furthermore, SE evaluates the quality of the partitioning tree globally, making it a suitable *global* objective for graph-based hierarchical clustering.

Continuous SE in Hyperbolic Space

We address **Challenge 1** by introducing CSE as the global objective in HypCSE. Both geodesics in hyperbolic space and shortest paths in trees can be viewed as paths with the shortest distances between points. This relationship allows us to derive hyperbolic LCA analogous to the discrete LCA (Chami et al. 2020). To achieve continuous optimization for HypCSE, we reformulate SE based on LCA and derive a differentiable objective CSE by the hyperbolic LCA.

Definition 4 (Structural Entropy via LCA)**.** Given an undirected weighted graph $G = (V, E, \mathbf{W})$ and an associated rooted tree $\mathcal{T}$, the SE of G on $\mathcal{T}$ can be defined as

$$\begin{aligned}\mathcal{H}^{\mathcal{T}}(G) = &\frac{2}{\mathcal{V}(G)} \sum_{(v_i,v_j)\in E} \mathbf{W}_{ij} \log_2 \mathcal{V}_{\mathcal{T}_i\vee\mathcal{T}_j} \\ &-\frac{1}{\mathcal{V}(G)} \sum_{v_i\in V} \mathcal{V}_{\mathcal{T}_i} \log_2 \mathcal{V}_{\mathcal{T}_i},\end{aligned} \tag{4}$$

where $\mathbf{W}_{ij}$ is the edge weight between pair (v_i, v_j), $\mathcal{T}_i$ is the leaf node in $\mathcal{T}$ that contains only one vertex v_i, and $\mathcal{V}_{\mathcal{T}_i\vee\mathcal{T}_j}$ is the volume of the LCA of leaf pair $(\mathcal{T}_i, \mathcal{T}_j)$. For a given graph G, the values of the second term in Eq. (4) are the same for all partitioning trees.

Theorem 5 (Equivalence ((Pan, Zheng, and Fan 2021)))**.** *$\mathcal{H}^{\mathcal{T}}(G)$ in Def. 4 is equivalent to Eq. (1) in Def. (1).*

LCA Volume in Hyperbolic Space. According to Eq. (4), the calculation of $\mathcal{H}^{\mathcal{T}}(G)$ requires finding the LCA of leaf node pairs and calculating the volume. Given the partitioning tree $\mathcal{T}$, the LCA $\mathcal{T}_i \vee \mathcal{T}_j$ of a leaf pair $(\mathcal{T}_i, \mathcal{T}_j)$ is the node on the shortest path between them and closest to the root λ. We have $\mathcal{V}_{\mathcal{T}_i\vee\mathcal{T}_j} = \sum_{v_k\in V} \mathcal{V}_{\mathcal{T}_k} \mathbb{I}[\{\mathcal{T}_k|\mathcal{T}_i, \mathcal{T}_j\}]$, where $\mathbb{I}[\cdot]$ is the indicator function and $\{\mathcal{T}_k|\mathcal{T}_i, \mathcal{T}_j\}$ means $\mathcal{T}_k$ is a descendant of $\mathcal{T}_i \vee \mathcal{T}_j$. When leaf nodes $\mathcal{T}_i$ and $\mathcal{T}_j$ (corresponding to graph vertices v_i, v_j) are embedded in hyperbolic space, denoted as $\mathbf{z}_i$ and $\mathbf{z}_j$, the shortest path between them is geodesic. Their hyperbolic LCA $\mathbf{z}_i \vee \mathbf{z}_j$ can be defined as the point closest to the origin on this geodesic. The hyperbolic LCA volume is calculated as $\mathcal{V}_{\mathbf{z}_i\vee\mathbf{z}_j} = \sum_{v_k\in V} \mathcal{V}_{\mathbf{z}_k} \mathbb{I}[\{\mathbf{z}_k|\mathbf{z}_i, \mathbf{z}_j\}]$, where $\{\mathbf{z}_k|\mathbf{z}_i, \mathbf{z}_j\}$ means $\mathbf{z}_k$ is a descendant of $\mathbf{z}_i \vee \mathbf{z}_j$.

Lemma 6 (Descendant via LCA-Appendix A.2)**.** *For leaves $\mathcal{T}_i, \mathcal{T}_j$ and $\mathcal{T}_k$ in rooted tree $\mathcal{T}$, $\mathcal{T}_k$ is the descendant of $\mathcal{T}_i \vee \mathcal{T}_j$ if and only if both the following statements hold:*

$$d_\lambda(\mathcal{T}_i \vee \mathcal{T}_j) \leq d_\lambda(\mathcal{T}_i \vee \mathcal{T}_k), \quad d_\lambda(\mathcal{T}_i \vee \mathcal{T}_j) \leq d_\lambda(\mathcal{T}_j \vee \mathcal{T}_k), \tag{5}$$

where $d_\lambda(\mathcal{T}_i)$ is the distance between $\mathcal{T}_i$ and the root λ in $\mathcal{T}$.

Lemma 7 (Distance of geodesic to origin (Chami et al. 2020))**.** *For two points $(\mathbf{x}, \mathbf{y}) \in \mathbb{B}^{1,d}$ where $\mathbf{x} \vee \mathbf{y}$ is the point on the geodesic connecting $\mathbf{x}$ and $\mathbf{y}$ that minimizes the distance to the origin $\mathbf{o}$, let $\mathbf{r}$ be the symmetric point to $\mathbf{x}$ with respect to the circle at infinity, and $\mathbf{o}_{\text{ref}}$ be the symmetric point to $\mathbf{o}$ with respect to the geodesic. We have:*

$$\begin{aligned} d_o^{\mathbb{B}}(\mathbf{x} \vee \mathbf{y}) &= \frac{\text{artanh}(\|\mathbf{o}_{\text{ref}}\|_2^2)}{2}, \\ \text{where } \mathbf{o}_{\text{ref}} &= \frac{\|\mathbf{r}\|_2^2 - 1}{\|\mathbf{o}_{\text{invref}} - \mathbf{r}\|_2^2} \cdot (\mathbf{o}_{\text{invref}} - \mathbf{r}) + \mathbf{r}, \\ \mathbf{o}_{\text{invref}} &= \frac{2\mathbf{x}^\top \cdot \mathbf{y}_{\text{inv}}}{\|\mathbf{x}\|_2^2} \cdot \mathbf{x} - \mathbf{x}, \;\; \mathbf{r} = \frac{\mathbf{x}}{\|\mathbf{x}\|_2^2}, \\ \text{and } \mathbf{y}_{\text{inv}} &= \frac{\|\mathbf{r}\|_2^2 - 1}{\|\mathbf{y} - \mathbf{r}\|_2^2} \cdot (\mathbf{y} - \mathbf{r}) + \mathbf{r}. \end{aligned} \tag{6}$$

CSE objective function. Given a graph G along with the partitioning tree $\mathcal{T}$, minimizing the SE is equivalent to minimizing the following cost function:

$$C_G^{\mathcal{T}} = \sum_{(v_i,v_j)\in E} \mathbf{W}_{ij} \log_2 \left(\mathcal{V}_{\mathcal{T}_i} + \mathcal{V}_{\mathcal{T}_j} + \sum_{k\in V}^{k\neq i,j} \mathcal{V}_{\mathcal{T}_k} \mathbb{I}[\{\mathcal{T}_k|\mathcal{T}_i, \mathcal{T}_j\}] \right). \tag{7}$$

Term $\mathbb{I}[\{\mathcal{T}_k|\mathcal{T}_i, \mathcal{T}_j\}]$ is the non-differential term in Eq. (7), which is an indicator for whether $\mathcal{T}_k$ is a descendant of $\mathcal{T}_i \vee \mathcal{T}_j$. According to Lemma 6, this indicator can be determined by the relationship among hyperbolic distances $d_\lambda(\mathcal{T}_i \vee \mathcal{T}_j)$, $d_\lambda(\mathcal{T}_i \vee \mathcal{T}_k)$, and $d_\lambda(\mathcal{T}_j \vee \mathcal{T}_k)$. The indicator outputs true when only $d_\lambda(\mathcal{T}_i \vee \mathcal{T}_j)$ is the smallest one. When leaves $\mathbf{Z}_{\mathbb{B}} = \{\mathbf{z}_1, ..., \mathbf{z}_n\}$ in $\mathcal{T}$ are embedded in the hyperbolic space of Poincaré model, the distance of LCA to the root $d_\lambda(\mathcal{T}_i \vee \mathcal{T}_j)$ can be approximated as the distance of geodesic to the origin $d_o^{\mathbb{B}}(\mathbf{z}_i \vee \mathbf{z}_j)$, and the indicator can be approximated by a softmax function. CSE objective is then:

$$\mathcal{L}_{\text{cse}}^{\mathbf{Z}_{\mathbb{B}}}(G) = \sum_{(v_i,v_j)\in E} \mathbf{W}_{ij} \log_2(\mathcal{V}_{\mathbf{z}_i} + \mathcal{V}_{\mathbf{z}_j} + \widehat{\mathcal{V}}_{\mathbf{z}_i\vee\mathbf{z}_j}), \tag{8}$$

$$\widehat{\mathcal{V}}_{\mathbf{z}_i\vee\mathbf{z}_j} = \sum_{k\in V, k\neq i.j} (\mathcal{V}_{\mathbf{z}_k}, 0, 0) \cdot \sigma_{t_1}\left(s_{ij}^o, s_{ik}^o, s_{jk}^o\right)^\top, \tag{9}$$

where $\sigma_{t_1}(\cdot)$ is the scaled softmax function: $\sigma_{t_1}(\mathbf{x})_i = e^{\mathbf{x}_i/t_1} / \sum_j e^{\mathbf{x}_j/t_1}$, $s_{ij}^o = r_1 - d_o^{\mathbb{B}}(\mathbf{z}_i \vee \mathbf{z}_j)$, and t_1 and r_1 are hyperparameters.

Hyperbolic Hierarchical Clustering

Hereafter, we elaborate on achieving hierarchical clustering in hyperbolic space by minimizing CSE objective $\mathcal{L}_{\text{cse}}^{\mathbf{Z}}$. As depicted in Figure 2 (I), the hyperbolic hierarchical clustering module consists of three steps: graph construction,

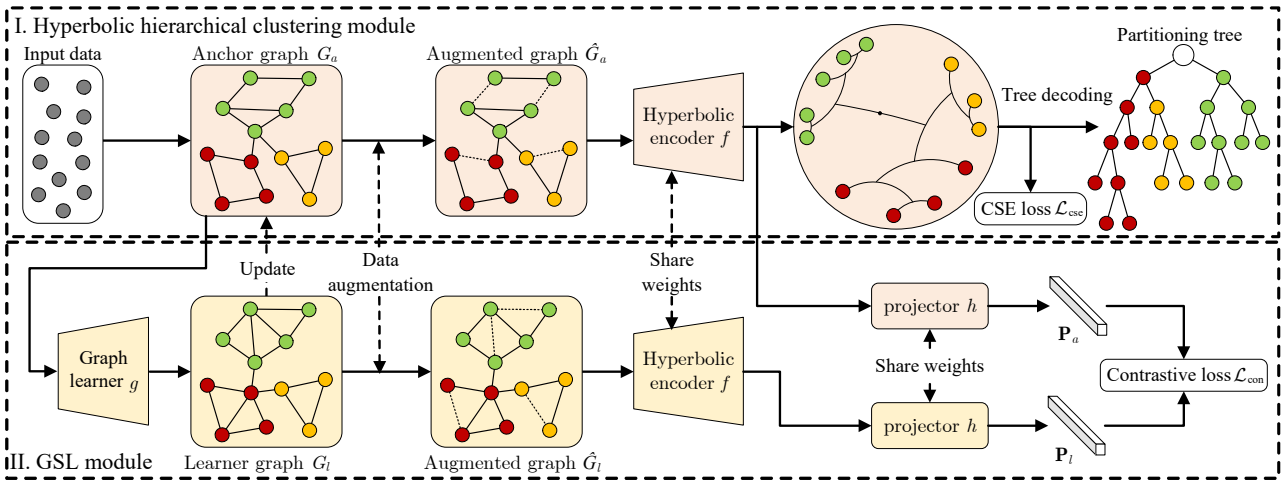


Figure 2: Framework of HypCSE. (I) In the hyperbolic hierarchical clustering module, we construct an anchor graph G_a from the input data, encode it using $f(\cdot)$, and decode it into a binary partitioning tree for hierarchical clustering. (II) In the GSL module, we learn a leaner graph G_l using graph learner $g(\cdot)$, update G_a from G_l, and guide $g(\cdot)$ via contrastive learning.

hyperbolic encoding, and partitioning tree decoding. Given that CSE is defined on graphs, for input data $\mathbf{X}$ containing n data points, we construct a similarity graph G and sparsify it to eliminate noise in data. First, we construct a weighted undirected graph $G = (V, E, \mathbf{W}, \mathbf{X})$ from $\mathbf{X}$. In practice, we measure the similarity between data points by Gaussian kernel with kernel width $\sigma = 1$ as edge weights and sparsify G by retaining the top $k = 10$ edges of each vertex. Next, we encode G via the Lorentz Convolution LConv (Chen et al. 2021) into an embedding $\mathbf{Z}$ in hyperbolic space, and minimize $\mathcal{L}_{\text{cse}}^{\mathbf{Z}}$ via gradient descent optimization. This embedding $\mathbf{Z}$ has a close correspondence to tree metrics (Sarkar 2011), where each vertex embedding $\mathbf{z}_i$ corresponds to a tree leaf and LCA of leaves form the partitioning tree (Chami et al. 2020). Finally, we heuristically decode $\mathbf{Z}$ into a binary partitioning tree for hierarchical clustering.

Hyperbolic graph embedding. Given a constructed attributed graph $G = (V, E, \mathbf{W}, \mathbf{X})$, we encode it into a hyperbolic embedding $\mathbf{Z} \in \mathbb{L}^{\kappa,d}$ via LConv (Chen et al. 2021). Two key components of LConv are the Lorentz linear layer LLinear and the Lorentz attention-based aggregation layer LAgg. For a graph vertex v_i with feature $\mathbf{x}_i$, its linear transform is defined as follows:

$$\text{LLinear}(\mathbf{x}_i) = \begin{bmatrix} \sqrt{\|\phi(\Theta\mathbf{x}_i, \mathbf{b})\|^2 - \frac{1}{\kappa}} \\ \phi(\Theta\mathbf{x}_i, \mathbf{b}) \end{bmatrix}, \tag{10}$$

where Θ and $\mathbf{b}$ are parameters, and ϕ is an operation function. The aggregation of $\mathbf{x}_i$ is defined as follows:

$$\begin{aligned} w_{ij} &= \frac{\exp(-\frac{1}{\sqrt{\dim(\mathbf{x}_i)}} d_{\mathbb{L}}^2(\mathbf{q}_i, \mathbf{k}_j))}{\sum_{l=1}^{n} \exp(-\frac{1}{\sqrt{\dim(\mathbf{x}_i)}} d_{\mathbb{L}}^2(\mathbf{q}_i, \mathbf{k}_l))}, \\ \text{LAgg}(\mathbf{x}_i) &= \frac{\sum_{v_j \in \mathcal{N}(v_i)} w_{ij} \mathbf{v}_j}{\sqrt{-\kappa} |\| \sum_{v_l \in \mathcal{N}(v_i)} w_{il} \mathbf{v}_l \|_{\mathbb{L}}|}, \end{aligned} \tag{11}$$

where $\dim(\mathbf{x}_i)$ is the dimension of $\mathbf{x}_i$, $\mathcal{N}(v_i)$ is the neighborhood of vertex v_i in G, $\mathbf{q}_i$, $\mathbf{k}_i$ and $\mathbf{v}_i$ are row vectors in the query set $\mathbf{Q}$, key set $\mathbf{K}$, and value set $\mathbf{V}$, respectively. The Lorentz convolution layer is defined as $\text{LConv}(\mathbf{X}|G) = \text{LAgg}(\text{LLinear}(\mathbf{X})) \in \mathbb{L}^{\kappa,d}$. In our Hyperbolic encoder $f(\cdot)$, we stack 3 layers of LConv to obtain the Lorentz embedding $\mathbf{Z}_{\mathbb{L}}$ of G. Afterwards, we transform $\mathbf{Z}_{\mathbb{L}}$ into Poincaré embedding $\mathbf{Z}_{\mathbb{B}}$ to facilitate $\mathcal{L}_{\text{cse}}^{\mathbf{Z}}$ minimization.

Hyperbolic tree decoding. To minimize SE, conventional algorithms use discrete optimization (Li and Pan 2016; Pan, Zheng, and Fan 2021) to output discrete encoding trees that best characterize the uncertainty of the hierarchical topology of graphs. HypCSE, however, minimizes the approximate CSE of graphs by optimizing $\mathcal{L}_{\text{cse}}$ on hyperbolic graph embeddings rather than explicit trees. For the Poincaré embedding $\mathbf{Z}_{\mathbb{B}} = \{\mathbf{z}_1, ..., \mathbf{z}_n\}$ encoded by LConv, we find a binary tree by a Single Linkage clustering-like (Gower and Ross 1969) decoding algorithm (Algorithm 1). We set each data point $\mathbf{z}_i$ as a cluster (tree leaves) and iteratively merge the two closest clusters C_α and C_β into a new cluster. The closeness of clusters is defined as:

$$\text{closeness}(C_\alpha, C_\beta) = \min_{\mathbf{z}_i \in C_\alpha, \mathbf{z}_j \in C_\beta} (d_o(\mathbf{z}_i \vee \mathbf{z}_j)). \tag{12}$$

To help tree leaves separate in the Poincaré disk, we normalize their embeddings to lie on the hyperbolic diameter (Chami et al. 2020). We merge clusters until all clusters are merged into one binary partitioning tree. With K-nearest neighbor graph construction and minimum spanning tree trick (Appendix B.2), the time complexity of Algorithm 1 can be reduced to $O(n \log n)$.

Graph Structure Learning

Minimizing SE on the constructed graph G guides the hyperbolic encoder $f(\cdot)$ in embedding G. However, we still encounter **Challenge 2:** optimization on the heuristically constructed static graph G neglects the significance of graph structure and overlooks the information in the original input data. To address this issue, we adopt a GSL technique

Algorithm 1: Tree decoding given embedding $\mathbf{Z}_{\mathbb{B}}$

Input: Embeddings of vertices $\mathbf{Z}_{\mathbb{B}} = \{\mathbf{z}_1, ..., \mathbf{z}_n\}$;
Output: Binary partitioning tree $\mathcal{T}$.
Initialize $\mathcal{T}$ to contain leaves $\{\mathbf{z}_1, ..., \mathbf{z}_n\}$ and root λ
Set each data point $\mathbf{z}_i$ as a separate cluster $C_i = \{\mathbf{z}_i\}$
Calculate the closeness of all cluster pairs
while number of clusters > 2 **do**
    Merge (C_α, C_β) into a new cluster $C_\gamma = C_\alpha \cup C_\beta$ conditioned on $\arg\min_{\alpha,\beta}\{\text{closeness}(C_\alpha, C_\beta)\}$
    for each remaining cluster C_δ **do**
        Calculate the closeness between C_γ and C_δ
    end for
end while
Set the remaining two clusters as the children of λ
return $\mathcal{T}$ (which is a binary tree)

(Liu et al. 2022) guided by contrastive learning in hyperbolic space to learn a better graph structure for SE minimization. As depicted in Figure 2 (II), the GSL module consists of two steps: graph learning and contrastive learning, as follows.

Graph Learning. We take the constructed graph $G = (V, E, \mathbf{X})$ as the anchor graph $G_a = (V, E_a, \mathbf{X})$, which provides stable guidance for GSL. To learn graph structure from $\mathbf{X}$, we generate a learner graph G_l via a GSL encoder. First, we apply a graph learner $g(\cdot)$ to learn vertex embeddings $\mathbf{E} = g(\mathbf{X}, E_a)$ for learner graph construction. Next, we construct the affinity matrix $\mathbf{A}_l$ of G_l by calculating the similarities between pairs of vertex embeddings. We then select p edges with the highest weights connected to each vertex to form the learner graph edge set E_l, resulting in the learner graph $G_l = \{V, E_l, \mathbf{X}\}$. To facilitate a more informative anchor graph G_a, we update its edge set E_a using E_l as:

$$E_a \leftarrow S_\tau(E_a) + S_{1-\tau}(E_l), \tag{13}$$

where $\tau \in (0, 1]$ is the decay rate, and $S_\tau(\cdot)$ represents a random sampling operator with rate τ. We choose a large τ and update G_a after each epoch, allowing it to gradually assimilate new and eliminate erroneous information. Graph G_a is then used to calculate $\mathcal{L}^{\mathbf{Z}}_{\text{cse}}(G_a)$ in Eq. (8).

Contrastive Learning. We adopt contrastive learning to guide the graph learner $g(\cdot)$ and learn more discriminative vertex features. After obtaining the anchor graph G_a and the learner graph G_l, we perform data augmentation by randomly removing edges and masking vertex features. These graphs are then embedded in hyperbolic space as $\mathbf{Z}_a$ and $\mathbf{Z}_l$ by the hyperbolic encoder $f(\cdot)$. These embeddings are passed to a projector $h(\cdot)$ with 2 layers of LLinear:

$$\mathbf{P}_a = h(\mathbf{Z}_a),\ \mathbf{P}_l = h(\mathbf{Z}_l). \tag{14}$$

Contrastive learning aims to learn representations that can differentiate between similar and dissimilar data points. It usually builds upon similarities between data point representation pairs (Chen et al. 2020). Unlike representations in Euclidean space, the similarities between vertex features in this context are difficult to define. Motivated by (Ge et al. 2023), we instead minimize the hyperbolic distance of two views from the same vertex and maximize the distance from different vertices. For two representations $\mathbf{P}_a = \{\mathbf{p}_a^1, \ldots, \mathbf{p}_a^n\}$ and $\mathbf{P}_l = \{\mathbf{p}_l^1, \ldots, \mathbf{p}_l^n\}$, the hyperbolic contrastive loss is calculated as follows:

$$\mathcal{L}_{\text{con}} = \frac{1}{2n}\sum_{i=1}^{n}\left[L(\mathbf{p}_l^i, \mathbf{p}_a^i) + L(\mathbf{p}_a^i, \mathbf{p}_l^i)\right],$$
$$L(\mathbf{p}_l^i, \mathbf{p}_a^i) = -\log\frac{\exp\left\{[r_2 - d_{\mathbb{L}}(\mathbf{p}_l^i, \mathbf{p}_a^i)]/t_2\right\}}{\sum_{k=1}^{n}\exp\left\{[r_2 - d_{\mathbb{L}}(\mathbf{p}_l^i, \mathbf{p}_a^k)]/t_2\right\}}, \tag{15}$$

where r_2 and t_2 are hyperparameters.

Overall Framework

We apply several scalability strategies to improve HypCSE's efficiency, reducing the time complexity to $O(n \log n)$ (Appendix B). To generate more balanced partitioning trees, we ensure that the learned leaf embeddings in $\mathbf{Z}_{\mathbb{L}}$ are scattered around the origin by minimizing the distance between the centroid of leaves and the origin. We introduce the centroid loss $\mathcal{L}_{\text{cen}}$ based on Lorentz distance (Law et al. 2019) as:

$$\mathcal{L}_{\text{cen}} = d_o^{\mathbb{L}}\left(\frac{\sum_{j=1}^{n}\mathbf{z}_j}{\sqrt{-\kappa}|\|\sum_{i=1}^{n}\mathbf{z}_i\|_{\mathbb{L}}|}\right), \tag{16}$$

where $d_o^{\mathbb{L}}(\mathbf{z}_i)$ is the distance $\mathbf{z}_i$ to origin in Lorentz model. In summary, the overall objective is formulated as:

$$\mathcal{L}_{\text{HypCSE}} = \mathcal{L}_{\text{cse}} + \eta_1\mathcal{L}_{\text{con}} + \eta_2\mathcal{L}_{\text{cen}}, \tag{17}$$

where the hyperparameters η_1 and η_2 are simply set as 1.

Experiments

Experimental Setup

We adopt two metrics, including Dendrogram Purity (DP) (Heller and Ghahramani 2005; Chami et al. 2020) and Structural Entropy (SE) (Li and Pan 2016), for hierarchical clustering performance evaluation. DP is a holistic measure of partitioning trees, defined as the average purity score of LCAs of leaf pairs with the same ground truth labels. SE of a partitioning tree and its corresponding graph quantifies the amount of uncertainty remaining in this graph, where a lower SE indicates a higher quality of the tree, eliminating more uncertainty in this graph. We calculate the SEs of the partitioning trees generated by all algorithms in the constructed anchor graphs without performing updates. In HypCSE, the hyperbolic encoder and projector are based on the Lorentz model with curvature $\kappa = -1$ and optimized via Riemannian Adam (Becigneul and Ganea 2019) in Geoopt (Kochurov, Karimov, and Kozlukov 2020), while the GSL encoder is in Euclidean space and optimized via Adam (Kingma 2014). We consistently set $t_1 = 1000$, $r_1 = 2$, $t_2 = 1$, and $r_2 = 0$, across all datasets, with the sole exception of PenDigits, for which the temperature hyperparameters are $t_1 = 1$ and $t_2 = 1000$. The default value for τ is 0.9999. Further implementation details are in Appendix D. We conduct all experiments 5 times and report the mean values. Source code is available at GitHub[1].

Datasets and Baselines. We conduct experiments on 7 clustering datasets from the UCI Machine Learning Repository (Markelle, Rachel, and Kolby 2024), whose size ranges

[1]https://github.com/SELGroup/HypCSE

Methods	Zoo		Iris		Wine		Br. Cancer		OpticalDigits		Spambase		PenDigits	
	DP↑	SE↓	DP↑	SE↓	DP↑	SE↓	DP↑	SE↓	DP↑	SE↓	DP↑	SE↓	DP↑	SE↓
SingleLinkage	97.6	2.037	81.2	3.483	67.9	3.909	85.1	4.977	73.3	2.839	58.9	7.180	70.0	6.125
BKM	64.2	2.179	82.4	3.939	86.1	3.698	95.7	5.057	50.8	3.364	65.6	6.898	69.1	5.135
HDBSCAN	96.4	2.357	76.6	4.161	53.5	4.680	83.3	5.617	58.5	3.710	57.8	8.011	64.1	7.478
HCSE	97.3	1.929	89.7	3.593	71.1	3.819	94.2	4.319	81.5	3.011	55.2	6.599	76.9	6.877
SpecWRSC	95.4	2.228	83.2	3.172	93.5	3.441	95.1	4.512	85.9	2.900	55.1	8.817	65.3	7.190
DPClusterHSBM	93.6	2.264	82.9	2.997	89.5	3.404	92.9	4.169	81.0	2.517	61.0	**5.194**	not converge	
UFit	93.3	2.496	81.5	3.236	78.9	3.670	95.0	4.318	69.7	3.186	59.9	6.737	70.0	6.386
HypHC	96.8	2.010	76.0	3.485	88.7	3.692	96.5	5.549	33.5	4.468	75.4	8.895	11.7	10.69
FPH	89.9	2.227	85.3	3.806	92.8	**3.102**	92.6	**3.581**	81.0	5.707	54.8	7.660	69.6	8.192
HypCSE	**97.9**	**1.822**	**95.1**	**2.957**	**93.9**	3.496	**96.8**	4.342	**86.4**	**2.336**	**75.5**	6.668	**81.4**	**4.704**

Table 1: Hierarchical clustering quality measured in DP (%) and SE. Bold: the best and underline: the runner-up performance.

Base	GL	CL	Zoo	Iris	Wine	Br.	Opt.	Spam.
✓			97.8	89.7	88.8	96.0	86.2	70.1
✓	✓		97.8	89.7	88.8	96.5	86.3	72.1
✓		✓	97.7	94.1	93.4	96.6	85.9	75.4
✓	✓	✓	**97.9**	**95.1**	**93.9**	**96.8**	**86.4**	**75.5**

Table 2: Experimental results (DP %) of ablation study.

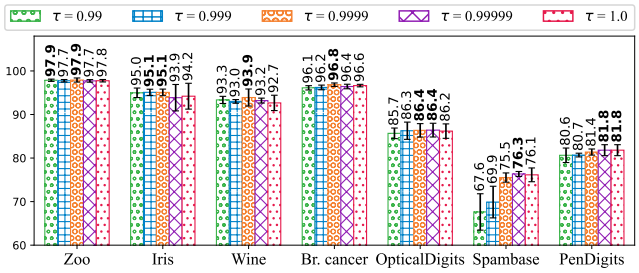


Figure 3: Parameter sensitivity on decay rate τ (DP %).

from 101 to 10,992. We compare HypCSE against 6 discrete hierarchical clustering methods, including SingleLinkage (Gower and Ross 1969), BKM (Moseley and Wang 2017), HDBSCAN(McInnes et al. 2017), SE-based Hierarchical Clustering (HCSE) (Pan, Zheng, and Fan 2021), SpecWRSC (Laenen, Manghiuc, and Sun 2023), and DPClusterHSBM (Imola et al. 2023). We also compare HypCSE against 3 continuous methods, including UFit (Chierchia and Perret 2019), HypHC (Chami et al. 2020), and FPH (Zügner et al. 2022). We adopt the same graphs as the proposed HypCSE for HCSE, SpecWRSC, and FPH since they are defined on graphs and assume the graphs are given. We adopt the graph construction methods in the original papers for other graph-based methods, i.e., DPClusterHSBM, UFit, and HypHC.

Hierarchical Clustering Quality

In Table 1, we report the performance of discrete and continuous methods for hierarchical clustering across 7 real-world datasets. HypCSE outperforms its discrete baseline HCSE in both metrics and achieves the best performance in DP across all datasets. This outcome verifies that the continuous optimization of a relaxed CSE objective in HypCSE is effective for hierarchical clustering. Regarding the SE metric, HypCSE achieves the best performance on 4 datasets and runner-up performance on 1 dataset. DPClusterHSBM achieves top-tier SE performance with the lowest SE on Spambase and the second lowest on 4 datasets. However, the SE metric is heavily dependent on the quality of the constructed similarity graph. Consequently, it might not be a reliable indicator when the similarities between data points are not accurately measured. On the Br. Cancer dataset, although HypCSE achieves the highest DP score, it performs poorly in the SE metric, indicating that the constructed graphs on this dataset fail to fully capture the class-discriminative features. HypCSE addresses this by learning an improved graph structure via the GSL module. We also report Dasgupta's costs of trees from all methods in Appendix E. The ranks of each method in Dasgupta's cost are similar to SE, indicating the consistency between 2 metrics.

Further Analysis

Ablation Study. To verify the effectiveness of two key components in the GSL module, we conduct an ablation study and report the experimental results in Table 2. The base model represents the hyperbolic hierarchical clustering module. Two key components in the GSL module are the Graph Learning (GL) and the Contrastive Learning (CL) components. We remove GL by replacing the learner graph G_l with the anchor graph G_a and removing the graph learner $g(\cdot)$. From Table 2, we find that both GL and CL components improve the overall performance. Without the guidance of the CL component, the GL component has a minimal effect on its own. The CL component improves overall performance by learning more discriminative features.

Parameter Sensitivity. We further investigate the parameter sensitivity of HypCSE. The graph learning decay rate τ controls the update speed of the anchor graph G_a, where a smaller τ leads to faster updates. We report the DP scores of HypCSE across different τ values, as shown in Figure 3. HypCSE showed stable performance across most datasets, with the notable exception of the Spambase dataset. This may indicate an ambiguous hierarchical structure within Spambase, which complicates graph structure

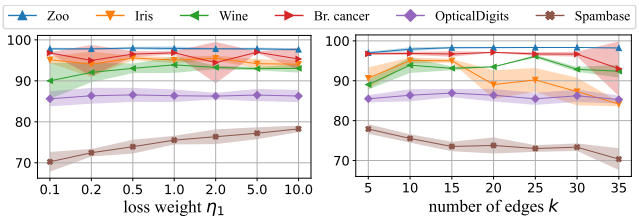


Figure 4: Parameter sensitivity on η_1 and k (DP %).

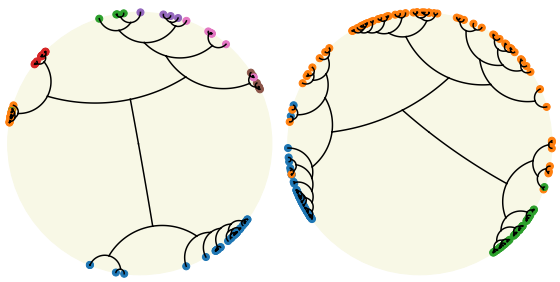
Methods	Zoo	Iris	Glass	Seg.
LP	41.4	76.7	46.8	65.3
HypHC-TS	84.8±3.5	84.4±1.7	50.6±2.6	64.1±0.9
HypHC-ETE	87.9±3.8	85.6±0.8	54.4±2.9	67.7±3.4
HypCSE	**88.3**±3.8	**91.8**±3.3	**54.7**±5.5	**78.8**±2.3

Table 3: Results (ACC %) of similarity-based classification.

learning. While the optimal τ varies by dataset, we observe that $\tau = 0.9999$ yields the best performance across 5 datasets, leading us to adopt it as the default value. We also investigate the sensitivity of contrastive loss weight η_1 and the number of edges k when constructing G_a, and report the DP scores in Figure 4. We find that η_1 rarely affects model performance, except for the Spambase dataset. On Spambase, HypCSE performs better with larger η_1 values, indicating the significant role of the contrastive loss $\mathcal{L}_{\text{con}}$ for this dataset. Similarly, the influence of k for the overall performance is also small, except for the Iris and Spambase datasets. On these datasets, performance drops as more edges are retained, suggesting an increase in noise. We simply set $\eta_1 = 1$ and $k = 10$ for all datasets. More parameter sensitivity analysis can be found in Appendix E.2.

Flexibility Analysis. Compared to discrete hierarchical clustering methods, continuous methods offer greater flexibility, as they can be combined with other gradient descent optimization-based methods. Following (Chami et al. 2020), we evaluate the performance of HypCSE with a classification loss for the similarity-based classification task. We split 4 datasets from the UCI ML Repository into training, test, and validation sets (30/60/10% splits), where similarities of all data points are available during training, and the test and validation set labels are only available during testing and validation. To ensure that only similarities are available for classification, we utilize the similarities of each data point to all other points as the data feature and reduce the feature dimension to 10 via Principal Component Analysis. We compare HypCSE against HypHC End-to-End learning with a classification loss (HypHC-ETE), HypHC embed-then-classify Two-Step approach (HypHC-TS), and a similarity-based semi-supervised learning method, Label Propagation (LP). The experimental results are reported in Table 3. Similar to HypHC, the proposed HypCSE can also be jointly optimized with the classification loss. HypCSE achieves the highest ACC on all datasets, demonstrating its flexibility in jointly integrating with other machine learning pipelines.

Visualization. We visualize the partitioning trees generated by HypCSE on a Poincaré disk and normalize the tree leaves to lie on the disk border. The Zoo dataset has 7 classes and the Wine dataset has 3 classes, each with a dominant class (characterized by a large number of data points) in colors blue and orange, respectively. For both partitioning trees, most leaves from the same class are correctly grouped into the same subtrees. For Zoo, several leaves of the class in pink are mis-clustered into nearby subtrees. For Wine, the leaves of the class in orange are too large in number, so some of them are mis-clustered into the other two subtrees. In all, the visualization of trees verifies the effectiveness of HypCSE in graph embedding and hierarchical clustering.



(a) Zoo (b) Wine

Figure 5: Visualization of partitioning trees in space $\mathbb{B}^2$.

Related Work

Hierarchical clustering algorithms group data points into nested clusters organized as a dendrogram, and they can be categorized into discrete and continuous optimization methods (Zügner et al. 2022). Conventional discrete algorithms include bottom-up agglomerative and top-down divisive methods (Ran et al. 2023). In contrast, continuous methods learn dendrograms via gradient descent and offer advantages in flexibility, allowing for joint optimization with other pipelines (Chami et al. 2020). Before Dasgupta's cost was proposed (Dasgupta 2016), most existing methods lacked a global objective function (Chierchia and Perret 2019). SE (Li and Pan 2016) also serves as a global cost objective for hierarchical clustering (Pan, Zheng, and Fan 2021) by quantifying information in graphs using partitioning trees.

Conclusion

In this paper, we introduce HypCSE, a continuous approach for structure-enhanced hierarchical clustering from the perspective of structural entropy. We develop continuous structural entropy (CSE) by reformulating the classic structural entropy objective using the lowest common ancestor on trees and approximating it in hyperbolic space. This differentiable formulation enables HypCSE to minimize CSE while simultaneously optimizing other objectives via gradient descent, offering significant flexibility. To avoid relying on predefined and potentially suboptimal graphs, HypCSE dynamically updates graph structures under the guidance of hyperbolic contrastive learning. Extensive experiments on 7 real-world datasets demonstrate the superiority of HypCSE.

Acknowledgments

The corresponding author is Yicheng Pan. This work is partly supported by National Key R&D Program of China (2021YFB3500700), the NSFC through grants 62322202, 62441612, and 62202164, CCF-DiDi GAIA collaborative Research Funds for Young Scholars through grant 202527, National Key Laboratory under grant 241-HF-D07-01, the Fundamental Research Funds for the Central Universities, and State Key Laboratory of Complex & Critical Software Environment (CCSE-2024ZX-20).

References

Becigneul, G.; and Ganea, O.-E. 2019. Riemannian Adaptive Optimization Methods. In *Proceedings of the ICLR*, 1–11.

Chami, I.; Gu, A.; Chatziafratis, V.; and Ré, C. 2020. From trees to continuous embeddings and back: Hyperbolic hierarchical clustering. In *Proceedings of the NeurIPS*, volume 33, 15065–15076.

Charpentier, B.; and Bonald, T. 2019. Tree sampling divergence: an information-theoretic metric for hierarchical graph clustering. In *Proceedings of the IJCAI*, 2067–2073.

Chen, L.; and Li, S. C. 2023. Incorporating cell hierarchy to decipher the functional diversity of single cells. *Nucleic acids research*, 51(2): e9–e9.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the ICML*, 1597–1607. PMLR.

Chen, W.; Han, X.; Lin, Y.; Zhao, H.; Liu, Z.; Li, P.; Sun, M.; and Zhou, J. 2021. Fully hyperbolic neural networks. *arXiv preprint arXiv:2105.14686*.

Cheng, D.; Kannan, R.; Vempala, S.; and Wang, G. 2006. A divide-and-merge methodology for clustering. *ACM Transactions on Database Systems (TODS)*, 31(4): 1499–1525.

Chierchia, G.; and Perret, B. 2019. Ultrametric fitting by gradient descent. In *Proceedings of the NeurIPS*, 1–12.

Ciaramella, A.; Nardone, D.; and Staiano, A. 2020. Data integration by fuzzy similarity-based hierarchical clustering. *BMC bioinformatics*, 21: 1–15.

Cohen-Addad, V.; Kanade, V.; Mallmann-Trenn, F.; and Mathieu, C. 2019. Hierarchical clustering: Objective functions and algorithms. *Journal of the ACM*, 66(4): 1–42.

Dasgupta, S. 2016. A cost function for similarity-based hierarchical clustering. In *Proceedings of the STOC*, 118–127.

Ge, S.; Mishra, S.; Kornblith, S.; Li, C.-L.; and Jacobs, D. 2023. Hyperbolic contrastive learning for visual representations beyond objects. In *Proceedings of the CVPR*, 6840–6849.

Gower, J. C.; and Ross, G. J. 1969. Minimum spanning trees and single linkage cluster analysis. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 18(1): 54–64.

Heller, K. A.; and Ghahramani, Z. 2005. Bayesian hierarchical clustering. In *Proceedings of the ICML*, 297–304.

Imola, J.; Epasto, A.; Mahdian, M.; Cohen-Addad, V.; and Mirrokni, V. 2023. Differentially private hierarchical clustering with provable approximation guarantees. In *Proceedings of the ICML*, 14353–14375. PMLR.

Kingma, D. P. 2014. Adam: A method for stochastic optimization. In *Proceedings of the ICLR*, 1–11.

Kochurov, M.; Karimov, R.; and Kozlukov, S. 2020. Geoopt: Riemannian optimization in pytorch. In *37th ICML GRLB Workshop*, 1–6.

Laenen, S.; Manghiuc, B. A.; and Sun, H. 2023. Nearly-optimal hierarchical clustering for well-clustered graphs. In *Proceedings of the ICML*, 18207–18249. PMLR.

Law, M.; Liao, R.; Snell, J.; and Zemel, R. 2019. Lorentzian distance learning for hyperbolic representations. In *Proceedings of the ICML*, 3672–3681. PMLR.

Li, A. 2024. *Science of Artificial Intelligence: Mathematical Principles of Intelligence (In Chinese)*. Beijing: Science Press.

Li, A.; and Pan, Y. 2016. Structural information and dynamical complexity of networks. *IEEE TIT*, 62(6): 3290–3339.

Liu, Y.; Zheng, Y.; Zhang, D.; Chen, H.; Peng, H.; and Pan, S. 2022. Towards unsupervised deep graph structure learning. In *Proceedings of the ACM Web Conference*, 1392–1403.

Markelle, K.; Rachel, L.; and Kolby, N. 2024. "The UCI Machine Learning Repository".

McInnes, L.; Healy, J.; Astels, S.; et al. 2017. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, 2(11): 205.

Monath, N.; Kobren, A.; Krishnamurthy, A.; and McCallum, A. 2017. Gradient-based hierarchical clustering. In *Proceedings of the NeurIPS workshop*, 1–6.

Monath, N.; Zaheer, M.; Silva, D.; McCallum, A.; and Ahmed, A. 2019. Gradient-based hierarchical clustering using continuous representations of trees in hyperbolic space. In *Proceedings of the KDD*, 714–722.

Moseley, B.; and Wang, J. R. 2017. Approximation bounds for hierarchical clustering: average linkage, bisecting K-means, and local search. In *Proceedings of the NeurIPS*, 3097–3106.

Pan, Y.; Zheng, F.; and Fan, B. 2021. An Information-theoretic Perspective of Hierarchical Clustering. *arXiv preprint arXiv:2108.06036*, 1–21.

Peng, W.; Varanka, T.; Mostafa, A.; Shi, H.; and Zhao, G. 2021. Hyperbolic deep neural networks: A survey. *IEEE TPAMI*, 44(12): 10023–10044.

Ran, X.; Xi, Y.; Lu, Y.; Wang, X.; and Lu, Z. 2023. Comprehensive survey on hierarchical clustering algorithms and the recent developments. *Artificial Intelligence Review*, 56(8): 8219–8264.

Sarkar, R. 2011. Low distortion delaunay embedding of trees in hyperbolic plane. In *International symposium on graph drawing*, 355–366. Springer.

Sun, L.; Huang, Z.; Peng, H.; Wang, Y.; Liu, C.; and Philip, S. Y. 2024. LSEnet: Lorentz Structural Entropy Neural Network for Deep Graph Clustering. In *Proceedings of the ICML*, 1–10.

Sun, L.; Huang, Z.; Zhang, M.; and Yu, P. S. 2025a. Deeper with Riemannian Geometry: Overcoming Oversmoothing and Oversquashing for Graph Foundation Models. In *Advances in the 39th Neural Information Processing Systems (NeurIPS)*.

Sun, L.; Huang, Z.; Zhou, S.; Wan, Q.; Peng, H.; and Yu, P. S. 2025b. RiemannGFM: Learning a Graph Foundation Model from Riemannian Geometry. In *Proceedings of the ACM on Web Conference 2025 (WWW)*, 1154–1165. ACM.

Wang, D.; and Wang, Y. 2020. An improved cost function for hierarchical cluster trees. *Journal of Computational Geometry*, 11(1): 283–331.

Yan, J.; Luo, L.; Deng, C.; and Huang, H. 2021. Unsupervised hyperbolic metric learning. In *Proceedings of the CVPR*, 12465–12474.

Zhang, Y. W.; Wang, M. B.; and Li, S. C. 2021. SuperTAD: robust detection of hierarchical topologically associated domains with optimized structural information. *Genome biology*, 22: 1–20.

Zügner, D.; Charpentier, B.; Ayle, M.; Geringer, S.; and Günnemann, S. 2022. End-to-end learning of probabilistic hierarchies on graphs. In *Proceedings of the ICLR*, 1–23.