Gradient-based Hierarchical Clustering using Continuous Representations of Trees in Hyperbolic Space

Nicholas Monath[*]
nmonath@cs.umass.edu
CICS, UMass Amherst

Manzil Zaheer
manzilz@google.com
Google Research

Daniel Silva
dansilva@google.com
Google Research

Andrew McCallum
mccallum@cs.umass.edu
CICS, UMass Amherst

Amr Ahmed
amra@google.com
Google Research

ABSTRACT

Hierarchical clustering is typically performed using algorithmic-based optimization searching over the discrete space of trees. While these optimization methods are often effective, their discreteness restricts them from many of the benefits of their continuous counterparts, such as scalable stochastic optimization and the joint optimization of multiple objectives or components of a model (e.g. end-to-end training). In this paper, we present an approach for hierarchical clustering that searches over continuous representations of trees in hyperbolic space by running gradient descent. We compactly represent uncertainty over tree structures with vectors in the Poincaré ball. We show how the vectors can be optimized using an objective related to recently proposed cost functions for hierarchical clustering [16, 49]. Using our method with a mini-batch stochastic gradient descent inference procedure, we are able to outperform prior work on clustering millions of ImageNet images by 15 points of dendrogram purity. Further, our continuous tree representation can be jointly optimized in multi-task learning applications offering a 9 point improvement over baseline methods.

CCS CONCEPTS

• **Computing methodologies → Cluster analysis**.

KEYWORDS

Clustering, Hierarchical clustering, Gradient-based Clustering

ACM Reference Format:
Nicholas Monath, Manzil Zaheer, Daniel Silva, Andrew McCallum, and Amr Ahmed. 2019. Gradient-based Hierarchical Clustering using Continuous Representations of Trees in Hyperbolic Space. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330997

[*]Work done as an intern at Google.

1 INTRODUCTION

Hierarchical clustering is a ubiquitous and often-used tool for data analysis [44, 57], visualization [23, 43] and mining of meaningful representations of data [8]. In hierarchical clustering, data points are arranged as the leaves of a multi-layered tree structure with internal nodes representing meaningful and potentially overlapping sub-clusters of the data.

Hierarchical clustering is often used downstream as a component of larger systems [23, 37, 43]. It can discover tree structured representations that are used in tasks such as personalization and recommendation [55]. Moreover, it is also used on its own to solve coreference and record linkage tasks [12, 15, 32, 50] in which a set of entity mentions are clustered to discover entities.

Hierarchical clusterings are typically found using discrete algorithmic methods that search over discrete tree structures [18, 27, 28, 30, 51, 54]. For example, hierarchical agglomerative clustering greedily merges sub-trees to form a complete dendrogram. Hierarchical clustering algorithms strive to optimize a particular cost, objective, or probabilistic model that designates which hierarchical partitions of the data are more favorable than others. For example, Moseley and Wang [34] give a cost, which is akin to Dasgupta's [16] and is well approximated by hierarchical agglomerative clustering with average linkage, and Adams et al. [1] give a MCMC-based inference procedure for a nested stick-breaking objective.

While these algorithms are often highly effective in practice [16, 28, 34], their discreteness inherently restricts them from several key advantages of continuous optimization such as scalability and joint optimization in down-stream applications. Continuous models are typically amenable to stochastic / mini-batch optimization, allowing for scalability to massive datasets. For example, flat clustering using mini-batch K-means performs very well and scales to massive amounts of data due to its use of stochastic optimization [42] . Furthermore, these gradient-based methods can often be implemented to efficiently run on specialized hardware such as GPUs. Importantly, it is natural for gradient-based optimization methods to be combined for the joint optimization of multiple objectives or components of a model for end-to-end training [46, 52]. In these joint optimization settings, signal from the underlying task can help inform the clustering algorithm and vice-versa.

In this paper, we present a new approach for hierarchical clustering, utilizing a embedded representation of tree structures in hyperbolic space, specifically the Poincaré ball. Our approach, *gradient-based hyperbolic hierarchical clustering* (gHHC), represents each node of a discrete tree structure using as a continuous vector. Child-parent relationships in the tree structure are based on the relative position and norms of the embedded node representations. The negative curvature of the hyperbolic space is widely known to accurately capture parent-child relationships [19, 35, 40]. We use the norm of vectors to model depth in the tree, requiring child nodes to have a larger norm than their parents. The root is near the origin of the space and the leaves near the edge of the ball. We present an objective function that is differentiable with respect to our tree node embeddings and perform hierarchical clustering by optimizing this objective using stochastic gradient descent. This objective has connections to recently proposed cost functions for hierarchical clustering [16, 49]. Note the distinction between this objective and previous work on learning representations in hyperbolic space [19, 35, 40], which are given a tree or graph structure to embed in hyperbolic space, and our method that *discovers* a meaningful tree structure using a hierarchical clustering objective.

A key feature of our approach is scalability to large datasets using mini-batch optimization and we present an efficient optimization algorithm that scales to datasets of millions of points. We show that our method outperforms state-of-the-art approaches on a clustering task of ImageNet ILSVRC images [28] by 15 points of dendrogram purity. Further, we apply our method to a multi-task learning application and jointly optimize the clustering of the tasks and the task specific regressors, resulting in an improvement of 9 points over baseline methods.

2 HYPERBOLIC GEOMETRY

Hyperbolic geometry is a non-Euclidean geometry, which drops the parallel line postulate while keeping the remaining four of the five of the postulates of Euclidean geometry. The resulting space has constant negative curvature. As a consequence, for a fixed dimension, the volume of any ball in such hyperbolic space grows exponentially with its radius rather than polynomially as in the Euclidean space (Figure 1). Another ramification of dropping the parallel line postulate is that a small perturbation of a point x by a vector v in the space is no longer the simple linear map $x + v$ as is in the case of Euclidean space. Instead, one has to carefully derive the perturbation map (formally known as the retraction map or exponential map), which would be important when performing optimization with gradient-descent over parameters lying in the hyperbolic space.

The recent attention to hyperbolic spaces can be primarily attributed to the exponential growth of volume in a ball with its radius, which allows for a parsimonious representation of hierarchical data. To elaborate, consider a complete binary tree. At level l, the tree would have 2^l nodes, i.e. the number of nodes grows exponentially with level, or, in other words, the distance to the root of the tree. In hyperbolic geometry this kind of tree structure can be modeled easily (with fixed dimensions) as nodes with increasing level can be arranged radially in a ball in hyperbolic space. Moreover, a linear time algorithm was proposed by Sarkar [41] for embedding the tree into hyperbolic space, while approximately maintaining the tree distance. This type of construction is feasible as hyperbolic annulus volume (and even the circle length) grows exponentially with its radius. This type of construction would be infeasible in Euclidean space where the volume growth is only polynomial. Such desirable properties has led to use of hyperbolic space in many applications [6, 26, 29, 31].

Formally, we consider the Poincaré ball model of hyperbolic space, which corresponds to a Riemannian manifold with a particular metric tensor (for more details see [21, 45]). The Poincaré ball model is defined on the set $\mathbb{D} = \{x \in \mathbb{R}^d : \|x\| < 1\}$. The induced distance between any two points $x, y \in \mathbb{D}$ can be derived as [35]:

$$d_{\mathbb{D}}(x, y) = \cosh^{-1}\left(1 + 2\frac{\|x - y\|^2}{(1 - \|x\|^2)(1 - \|y\|^2)}\right). \tag{1}$$

Then it follows that the induced norm of a point $x \in \mathbb{D}$ is:

$$\|x\|_{\mathbb{D}} = d_{\mathbb{D}}(x, 0) = \cosh^{-1}\left(\frac{1 + \|x\|^2}{1 - \|x\|^2}\right) = 2\tanh^{-1}(\|x\|). \tag{2}$$

We will use these properties of Poincaré ball in developing our proposed method.

3 GRADIENT-BASED HIERARCHICAL CLUSTERING

In this section, we present our proposed approach for discovering hierarchical clusterings using gradient descent. Our approach consists of three components: (1) a collection of continuous vector embeddings (in hyperbolic space) of the nodes of a discrete tree, (2) a mapping from the continuous embeddings to discrete trees, and (3) and objective and corresponding gradient-based optimization produced to update the position of these vectors given a dataset. We refer to our approach as *gradient-based hyperbolic hierarchical clustering* (gHHC).

3.1 Continuous Tree Representation

A hierarchical clustering, $\mathcal{T}$, is a tree structured partitioning of a dataset $X = \{x_1, \ldots, x_N\}$. Each data point x_i sits at a leaf node of the hierarchy. Internal nodes are typically thought of as representing a (flat) cluster containing their descendant leaves. Formally,

Definition 1. *Hierarchical clustering [30]. A hierarchical clustering, $\mathcal{T}$, of a dataset $\{x_i\}_{i=1}^N$, is a set of clusters $T_0 \triangleq \{x_i\}_{i=1}^N \in \mathcal{T}$ and for each $T_i, T_j \in \mathcal{T}$ either $T_i \subset T_j$, $T_j \subset T_i$ or $T_i \cap T_j = \emptyset$. For any cluster $T \in \mathcal{T}$, if $\exists T'$ with $T' \subset T$, then there exists a set $\{T_i\}_{i=1}^k$ of disjoint clusters such that $\bigcup_{i=1}^k T_i = T$.*

The hierarchical clustering refers to a tree consisting of nodes $\mathcal{N}$. Each node corresponding to a cluster $T_j \in \mathcal{T}$. In a slight abuse of notation, we will use T_j to refer to both the node and its corresponding cluster.

gHHC embeds a discrete tree structure with a continuous one as a set of vectors $Z = \{z_1, \ldots, z_k\},\ z_j \in \mathbb{D}^d$ in the d dimensional Poincaré ball. Each vector $z_j \in Z$ corresponds to a particular internal node internal node $T_j \in \mathcal{T}$. Leaf nodes in $\mathcal{T}$ are represented by their corresponding data points and do not have parameters in Z. In this paper, we primarily focus on data with unit norm that sits at the edge of the ball. Data that does not sit in the ball can be embedded into the space using a learned or random projection or by setting the norm of the vectors to be 1. A discrete tree structure can be defined in terms of its child-parent edges. Following previous work [31, 35, 40, 41], we use the norm and relative positions of vectors to

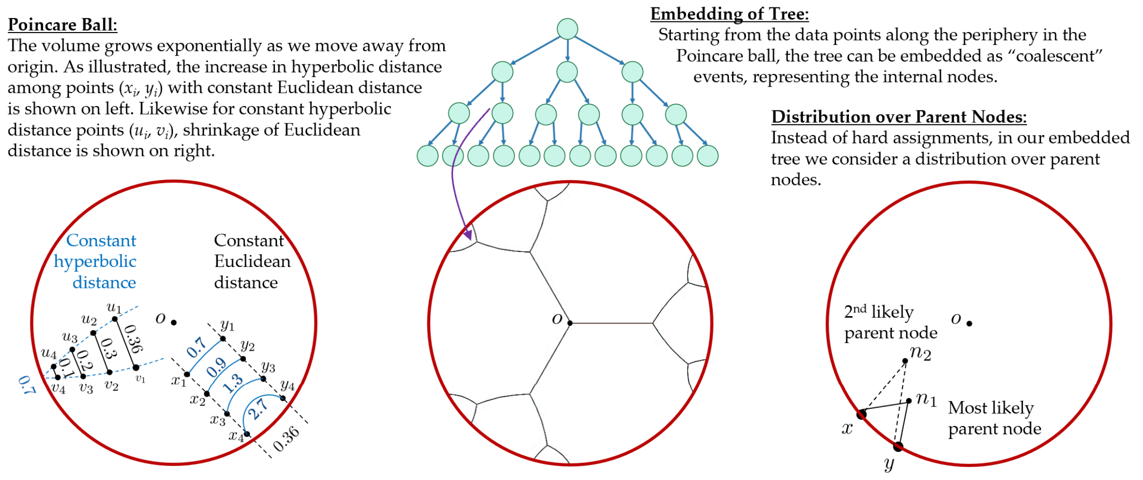
Poincare Ball:
The volume grows exponentially as we move away from origin. As illustrated, the increase in hyperbolic distance among points (x_i, y_i) with constant Euclidean distance is shown on left. Likewise for constant hyperbolic distance points (u_i, v_i), shrinkage of Euclidean distance is shown on right.
Constant hyperbolic distance
Constant Euclidean distance
o
0.7
0.9
1.3
2.7
0.36
0.3
0.2
0.1
0.36
0.7
Embedding of Tree:
Starting from the data points along the periphery in the Poincare ball, the tree can be embedded as "coalescent" events, representing the internal nodes.
o
Distribution over Parent Nodes:
Instead of hard assignments, in our embedded tree we consider a distribution over parent nodes.
2nd likely parent node
n_2
n_1
Most likely parent node
x
y
o

Figure 1: Unique features of gHHC: continuous representations of trees in the Poincaré ball and uncertainty over ancestors.

model the child-parent relationships in the tree structure. Vectors near the origin of the space are meant to indicate nodes closer to the root of the tree structure, while nodes closer to the edge of the ball are meant to indicate nodes closer to the leaf level of the tree. Section 2 reviewed a key property of the Poincaré ball, which is that there is exponentially more space moving away from the origin of the ball, just as how the number of descendants of a node in a discrete tree can grow exponentially. Child-parent relationships are modeled by the distance between two embedded nodes in the space and by the norm of the nodes. Parent nodes are those that are both nearby to their children in the space and have smaller norms than their children. As in previous work [35, 40], a child to parent dissimilarity function (not a distance metric) is used that encourages children to have a smaller norm than their parent:

$$d_{cp}(T_c, T_P) = d_{\mathbb{D}}(z_c, z_p)(1 + \max\{||z_p||_{\mathbb{D}} - ||z_c||_{\mathbb{D}}, 0\}). \quad (3)$$

In words, if the parent has smaller norm than the child, the dissimilarity between the child and parent nodes is their distance in hyperbolic space. Otherwise, the dissimilarity is the distance weighted by 1 plus the difference between the norm of the parent and the child.

This child-parent dissimilarity function can be used to extract a discrete tree from the embedded continuous representation of nodes. A discrete tree is predicted by having each data point and internal node select a parent under the condition that the parent must have a smaller norm:

$$\text{Parent}(T_c) = \underset{\substack{T_p \in \mathcal{N}, \\ ||z_p|| < ||z_c||}}{\text{argmin}} \; d_{cp}(T_c, T_p). \quad (4)$$

This allows gHHC to create trees with non-binary branching factor. Binary trees are more expressive in that they represent a strictly larger set of *tree consistent partitions*, which are a set of roots of disjoint subtrees that give a valid flat clustering of the dataset [5, 22, 28]. However, trees with n-ary or non-parametric branching factors [5, 27, 54] have a lower memory/parameter footprint as they contain fewer internal nodes and can provide more interpretable trees than binary trees, which might contain needlessly complicated structure, especially near the leaf level.

The continuous embedding representation of a tree structure allows for uncertainty to be represented in child-parent relationships. A particular child node might have multiple candidate parents that are nearby in the Poincaré ball. This powerful feature of our representation allows for the embeddings to encode uncertainty over alternative tree structures.

While our work focuses on the unnormalized child-parent dissimilarity function d_{cp}, we can use this to model a distribution over tree structures by using d_{cp} to define a distribution over parent nodes. The probability that a child node c has a parent p is inversely proportional to their child-parent dissimilarity.

$$P_{\text{par}}(T_p|T_c, Z) \propto \exp(-d_{cp}(T_C, T_P)). \quad (5)$$

The probability of a tree structure can then be computed by the product of all of the parent-child relationships in the tree

$$P(\mathcal{T}|Z) \propto \prod_{T_p \in \mathcal{T}} \; \prod_{T_C \in \texttt{chldrn}(T_p)} P_{\text{par}}(T_p|T_c, Z). \quad (6)$$

A minor point to note is that $P(\mathcal{T}|Z)$ would be computed in a way that considers only parent nodes that must have at least two children–effectively pruning any node that has a single child according to the definition in Eq. 4.

In this paper, we use a fixed number of internal node embeddings, fixing the size of the set Z. Some of these embeddings will not contribute meaningful tree structure, by either having no descendant data points or having only a single child. the nodes could be pruned in a post-processing step. In experiments, we use a number of internal nodes that is less than that of a binary tree structure.

3.2 Hierarchical Clustering Objectives

Given a dataset X, we would like to devise a hierarchical clustering cost function, $C_{\text{gHHC}}(X, Z)$, that we can use to optimize the placement of the internal node embeddings, Z. The cost function should be both faithful to the structure of the data and amenable to stochastic optimization using gradient descent.

Our objective is inspired by recent work on cost functions for hierarchical clustering [9–11, 13, 14, 16, 34, 49]. These objectives measure the quality of a particular discrete tree structure arrangement of a dataset given the pairwise similarities of the points. Our objective is aligned with the intuition of these recently proposed cost functions– data points that are highly similar should be near by one another in the tree structure and dissimilar points should be placed in separate branches, only having common ancestors near the root of the tree.

Given three data points x_i, x_j, and x_k, such that x_i and x_j are the most similar (least dissimilar) pair among the three possible pairs, our objective will encourage trees to represent a tree consistent partition that keeps x_i and x_j separate from x_k. In other words, we would like x_i and x_j to be merged closer to the leaves than x_k, so that, the least common ancestor of x_i and x_j is a descendant of the least common ancestor of all three nodes.

We make a few mild assumptions about X. As mentioned in the previous section, we assume that data point in X sits in $\mathbb{D}^d$. Second, we will assume that our dataset has a pairwise measure of similarity between data points: $w : X \times X \to \mathbb{R}^+$. Such an assumption is common throughout the clustering literature [3, 16, 28, 34]. For notation convenience, we will refer to the similarity between x_i and x_j as w_{ij} and will use $\mathbf{x}_{ij}$ to represent the pair x_i, x_j and $\mathbf{x}_{ijk}$ to represent the triple x_i, x_j, x_k.

We will first review two hierarchical clustering cost functions in the literature: Dasgupta's cost [16] and its recent extension by Wang and Wang [49]. Then we present the cost function we optimize for our model, $C_{\text{gHHC}}(X, \mathcal{T})$.

Dasgupta [16] presents a well-motivated cost function for hierarchical clustering that encourages similar points to be nearby in the tree structure. This cost has garnered much recent interest [9–11, 13, 14, 17, 34, 39, 49]. The cost function is:

$$\mathrm{C}_D(\mathcal{T}) = \sum_{\mathbf{x}_{ij} \in X^2} w_{i,j} |\texttt{lvs}\ (\texttt{lca}\ (\mathbf{x}_{ij}))|, \tag{7}$$

where $\texttt{lvs}\ (n)$ gives a set of leaf node descendants of n, and $\texttt{lca}\ (\mathbf{x}_{ij}) = \texttt{lca}\ (x_i, x_j)$ gives the least common ancestor of x_i and x_j in the discrete tree $\mathcal{T}$. In words, Dasgupta's cost says that one pays, for each pair of data points, the similarity of the pair times the number of leaves of the pair's least common ancestor. This precisely incentivizes the aforementioned intuition. However, this objective is not amenable for stochastic gradient methods due to its discrete nature.

Wang and Wang [49] recently proposed a related cost function based on a ranking of candidate configurations of the tree. The cost is defined over triples of data points x_i, x_j, x_k and incentivizes the most similar pair among the three to have a least common ancestor closer to the leaf level than the least common ancestor of all three points. The cost per triplet is

$$triC_{\mathcal{T}}(\mathbf{x}_{ijk}) = w_{i,j} + w_{i,k} + w_{j,k} - w_{\texttt{ord}(i,j,k,\mathcal{T})}. \tag{8}$$

$w_{\texttt{ord}(i,j,k,\mathcal{T})}$ is defined as the similarity of the pair of points that has the deepest least common ancestor among the three possible pairs or is zero if the three have the same least common ancestor. In words, if all of tree points x_i, x_j, x_k have the same least common ancestor, the cost is the sum of all of the pairwise similarities between the three points. Otherwise, if two points have a least common ancestor that is deeper in the tree than the least common ancestor of all three, then the cost is the sum of all of the pairwise similarities between the three points *except* that of the pair with the deeper least common ancestor. Formally, we write $w_{\texttt{ord}(i,j,k,\mathcal{T})}$ as follows, letting $\texttt{desc}(n, m)$ evaluate to true if n is an descendant of m in $\mathcal{T}$

$$\begin{aligned} w_{\texttt{ord}(i,j,k,\mathcal{T})} = {} & w_{i,j}\mathbb{I}[\texttt{desc}(\texttt{lca}\ (\mathbf{x}_{ij}), \texttt{lca}\ (\mathbf{x}_{ik}))] \\ & + w_{i,k}\mathbb{I}[\texttt{desc}(\texttt{lca}\ (\mathbf{x}_{ik}), \texttt{lca}\ (\mathbf{x}_{ij}))] \\ & + w_{j,k}\mathbb{I}[\texttt{desc}(\texttt{lca}\ (\mathbf{x}_{jk}), \texttt{lca}\ (\mathbf{x}_{ik}))]. \end{aligned}$$

The cost of a tree is the defined as:

$$C_{\text{Wang}}(\mathcal{T}, X) = \frac{\sum_{\mathbf{x}_{ijk} \in X^3} triC_{\mathcal{T}}(\mathbf{x}_{ijk})}{\sum_{\mathbf{x}_{ijk} \in X^3} minC(\mathbf{x}_{ijk})}, \tag{9}$$

where $minC$ represents the lowest cost possible for the triplet:

$$minC(\mathbf{x}_{ijk}) = \min\{w_{i,j} + w_{i,k},\ w_{i,j} + w_{j,k},\ w_{i,k} + w_{j,k}\}.$$

The cost represents the ratio between the cost of the given tree and the cost of an optimistic tree that could correctly order every triple of points. The authors show that the tree which optimizes $C_D(\mathcal{T})$ also optimizes $C_{\text{Wang}}(\mathcal{T})$. Ideally, we would hope to optimize the expected $C_{\text{Wang}}(\mathcal{T})$ under the distribution over discrete trees induced by our embedded continuous tree representation, $P(\mathcal{T}|Z)$. Noticing that the denominator of $C_{\text{Wang}}(\mathcal{T})$ is constant for a given dataset, consider:

$$\begin{aligned} & \mathbb{E}_{P(\mathcal{T}|Z)}\mathbb{E}_{(\mathbf{x}_{ijk})}\left[triC_{\mathcal{T}}(\mathbf{x}_{ijk})\right] \\ & = \mathbb{E}_{P(\mathcal{T}|Z)}\mathbb{E}_{(\mathbf{x}_{ijk})}\left[w_{i,j} + w_{i,k} + w_{j,k} - w_{\texttt{ord}(i,j,k,\mathcal{T})}\right]. \end{aligned} \tag{10}$$

Now, suppose $w_{ij} > \max(w_{ik}, w_{jk})$. We would like x_i and x_j to have a different least common ancestor of x_i, x_j, x_k and specifically one that is the deepest among least common ancestors of any pair of the three points. By the definition of $w_{\texttt{ord}(i,j,k,\mathcal{T})}$, we can upperbound Equation. 10 by only subtracting $w_{i,j}$ from the cost of all three pairwise similarities if x_i and x_j in fact have a least common ancestor that is a descendant of the least common ancestor of all three points, i.e.,:

$$\begin{aligned} \mathbb{E}_{P(\mathcal{T}|Z)}\mathbb{E}_{(\mathbf{x}x_{ijk})}\big[& w_{i,j} + w_{i,k} + w_{j,k} \\ & - w_{ij}\mathbb{I}[\texttt{desc}(\texttt{lca}\ (\mathbf{x}_{ij}), \texttt{lca}\ (\mathbf{x}_{ik}))]\big]. \end{aligned} \tag{11}$$

Unfortunately, optimizing this quantity directly is computationally challenging with our model and so we design a related objective. We use a geometric heuristic to provide an approximate distribution over least common ancestors for both x_i and x_j and, similarly, a distribution over least common ancestors for all three points x_i, x_j, and x_k. The distribution over least common ancestors for both x_i and x_j is:

$$P_{\text{lca}}(n|\mathbf{x}_{ij}) \propto \exp(-\max\{d_{cp}(x_i, n), d_{cp}(x_j, n)\}). \tag{12}$$

We would like to encourage x_i, x_j, and x_k to have a least common ancestor that is different than the least common ancestor of x_i, x_j. And so, in the distribution over least common ancestor for all three points, we give 0 probability mass to the most likely least common ancestor of x_i and x_j:

$$P_{\text{lca}}(n|\mathbf{x}_{ijk}) \propto \exp(-\xi_{ijk,n})\mathbb{I}[n \neq \operatorname*{argmax}_{n'} P_{\text{lca}}(n'|\mathbf{x}_{ij})] \tag{13}$$

$$\xi_{ijk,n} = \max\{d_{cp}(x_i, n), d_{cp}(x_j, n), d_{cp}(x_k, n)\}. \tag{14}$$

Our objective is to minimize the expected distance between x_i and the embeddings of the nodes that are likely to be a least common ancestor of x_i and x_j and increase the distance between x_i and nodes that are likely to be a least common ancestor of x_i, x_j, and x_k.

We model this by the ranking objective: $d_{cp}(x_i, n)(P_{\text{lca}}(n|x_i, x_j) - P_{\text{lca}}(n|x_i, x_j, x_k))$. Similar to Bayesian Personalized Ranking [38], we optimize the sigmoid of the aforementioned score. We optimize the analogous value for x_j. We minimize the distance between x_k and nodes that are likely to be a least common ancestor of x_i, x_j, and x_k and maximize the distance between x_k and the nodes likely to be a least common ancestor of x_i and x_j: $\sigma(d_{cp}(x_k, n)(P_{\text{lca}}(n|x_i, x_j, x_k) - P_{\text{lca}}(n|x_i, x_j)))$. Overall our cost is:

$$C_{\text{gHHC}}(X, Z) = \sum_{\mathbf{x}_{ijk} \in X^3} \sum_{n \in \mathcal{N}} \Big(\sigma\Big(d_{cp}(x_i, n)(P_{\text{lca}}(n|\mathbf{x}_{ij}) - P_{\text{lca}}(n|\mathbf{x}_{ijk}))\Big) \tag{15}$$

$$+ \sigma\Big(d_{cp}(x_j, n)(P_{\text{lca}}(n|\mathbf{x}_{ij}) - P_{\text{lca}}(n|\mathbf{x}_{ijk}))\Big)$$

$$+ \sigma\Big(d_{cp}(x_k, n)(P_{\text{lca}}(n|\mathbf{x}_{ijk}) - P_{\text{lca}}(n|\mathbf{x}_{ij}))\Big)\Big),$$

where $\sigma(y) = \frac{1}{1+e^{-y}}$ is the sigmoid function. This objective encourages trees that merge more similar points together closer to the leaves and less similar things higher up in the tree. It encourages at least one node to be an ancestor for x_i and x_j and not for x_k while encouraging another to serve as a possible ancestor for x_k as well as the other two. Furthermore, our objective is amenable to stochastic gradient descent with respect to Z by sampling triples of points (x_i, x_j, x_k).

3.3 Child-Parent Margin Objective

To encourage child-parent nodes to be well separated and not have equal norms, we use a margin-based version of the objective at inference time. We add a margin γ in d_{cp}:

$$d_{cp}(T_c, T_P; \gamma) = d_{\mathbb{D}}(z_c, z_p)(1 + \max\{||z_p||_{\mathbb{D}} - ||z_c||_{\mathbb{D}} + \gamma, 0\}), \tag{16}$$

For each pair of child-parent nodes, we attempt to minimize the distance between children and parents while maintaining the margin between their norms:

$$C_{cp:\text{marg}}(Z, \gamma) = \sum_{T_c \in \mathcal{N}} d_{cp}(T_c, \mathsf{Parent}(T_c); \gamma). \tag{17}$$

We alternate between optimizing this objective and the C_{gHHC} objective in our gradient-based inference procedure.

4 GRADIENT-DESCENT BASED INFERENCE

Given the objective in C_{gHHC} (Eq. 15), and a dataset X, we can perform hierarchical clustering by optimizing the objective with respect to the representation of the tree structure Z using gradient descent. As the objective is an expectation over triples of data points x_i, x_j, x_k from X, it can be optimized using mini-batch stochastic gradient descent. This allows the model to scale to massive datasets that do not fit in memory and can even allow for distributed optimization. Each gradient adjusts the placement of internal nodes by adjusting the relative distances to the data points x_i, x_j, x_k.

The internal node parameters of our model Z, sit in hyperbolic space. And so, for this reason, they cannot be optimized with standard Euclidean methods, but rather can be optimized using Riemannian gradient descent [7].

4.1 Gradient-Descent on Riemannian Manifold

Since the Poincaré ball has a Riemannian manifold structure, we need employ stochastic Riemannian optimization methods such as RSGD [7] or RSVRG [53]. Here we provide only a very basic intuition of these methods and outline the final results that we use. In every step of gradient descent, we slightly perturb the current point in the direction of steepest descent. Since our optimization variables lie in the Poincaré ball, we have to use the proper retraction operator $\Re_{(.)}$, unlike Euclidean space where a simple subtraction of scaled gradient would work, as pointed out in Section 2. That is, the gradient update would be of the form:

$$Z_{t+1} = \Re_{Z_t}(-\eta \nabla_R C_{\text{gHHC}}(Z_t)), \tag{18}$$

where η is the learning rate and $\nabla_R C_{\text{gHHC}}(Z_t)$ is the Riemannian gradient. For the retraction operator, following [35], we use the following first order approximation:

$$\Re_Z(g) = \begin{cases} z + g, & \text{if } \|z + g\| < 1, \\ (z + g)/\|z + g\| - \epsilon, & \text{else.} \end{cases} \tag{19}$$

Finally, the Riemannian gradient $\nabla_R C_{\text{gHHC}}(Z_t)$ can be expressed in terms of the Euclidean gradient by using the Riemannian metric tensor for the Poincare ball. We directly state the result here:

$$\nabla_R C_{\text{gHHC}}(Z_t) = \frac{(1 - \|Z\|^2)^2}{4} \nabla C_{\text{gHHC}(Z_t)} \tag{20}$$

Figure 2 shows how gHHC trees progress through SGD inference on a toy dataset.

4.2 Practical Considerations

Rather than sampling triples uniformly at random to optimize C_{ghhc} (Eq. 15), we sample triples x_i, x_j, x_k such that x_j is one of the K nearest neighbors of x_i and x_k is another point sampled at random. To encourage exploration of the space of internal nodes during optimization, we add Gumbel noise to distances so that nodes that might not be the closest will sometimes be selected.

We also find having a good initialization of the model improves performance as is the case with all clustering methods. We initialize the models by first creating a set of leaf nodes. These leaf nodes can either be selected at random from the data points or with an approximate farthest first / k-means seeding method [2]. Then we build an initial hierarchy over these leaf *internal* using hierarchical agglomerative clustering. We heuristically embed the nodes discovered by HAC back into the Poincaré ball by representing them as the average of their descendants and scaling the representation by a logarithmically decreasing factor based on the order of mergers in HAC (i.e., the node that was merged in the j^{th} of N rounds of HAC has norm $\frac{\log(N-j)}{\log N}$). This strategy is very efficient and should not be confused with running HAC over the whole dataset, as the number of leaf nodes in our method is fixed. To extract a discrete tree structure, we use the parent assignment in Equation 4. We assign an ordering to the nodes and in the case of a two nodes having the same norm, we break a tie using the ordering. We also perform an additional discrete tree extraction trick. If data points X and internal nodes M select a particular internal node n as their parent, we create a new node n' and assign n' to be a child of n along with the internal nodes M and set X to be the children of n'. We do this so that there is always a tree consistent partition that matches the data point to parent node assignments.

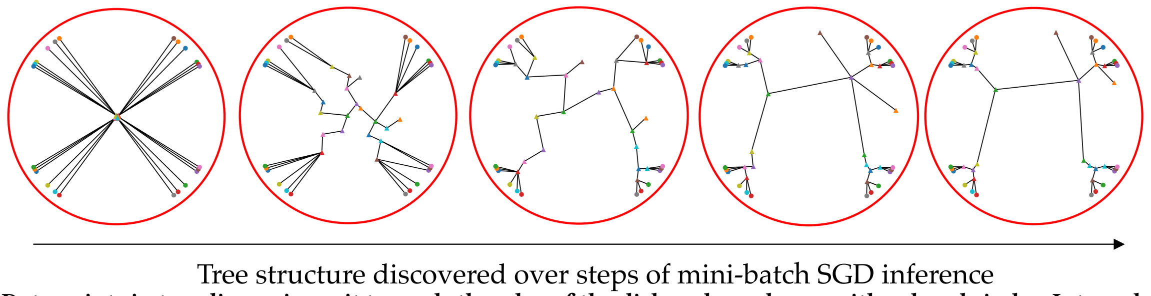
Tree structure discovered over steps of mini-batch SGD inference

Figure 2: Data points in two dimensions sit towards the edge of the disk and are shown with colored circles. Internal nodes are shown with colored triangles. We show how gHHC over gradient steps moves internal nodes and rectifies incorrect clustering. The dangling internal nodes correspond to nodes not used by the model and are pruned during post-processing.

5 END-TO-END OPTIMIZATION

In this section we demonstrate the efficacy of using our continuous representation of tree structures to perform end-to-end optimization in downstream applications. Recall that our optimization problem seeks to find an embedding of the internal tree nodes $Z_{\mathcal{T}}$ in the hyperbolic space that minimizes the cost $C_{gHHC}(X, Z)$ given in Eq.15 while assuming that the embedding of the data points X, $Z_{\mathcal{X}}$, is fixed over the boundary of the hyperbolic disk. In order to learn an embedding $Z_{\mathcal{X}}$, which needn't be constrained to be over the boundary of the hyperbolic disk, we need to define a problem-specific cost function, $C_{\text{problem}}(\mathcal{X})$, and jointly optimize it with $C_{gHHC}(X, Z)$ as follows:

$$\min_{Z_{\mathcal{T}}, Z_{\mathcal{X}}} C_{gHHC}(\mathcal{T}, X) + C_{\text{problem}}(\mathcal{X}). \tag{21}$$

Multi-task learning: In this case we are given a set of regression (classification) problems that are somehow related. We let X_i denotes the dataset of task i. The goal is to arrange the tasks in a tree structure and regularize the regression weights over the tree. In this case $C_{\text{problem}}(\mathcal{X}) = \sum_i loss(X_i; Z_{X_i})$, where Z_{X_i} in this case represents the regression (classification) weights of task i. As such, when optimizing Eq.21 using the above problem specific cost, we learn both the regression weights and the tree structure over tasks. We note here that using the optimization algorithm in Section 4, we sample three data points (d_i, d_j, d_k) from three regression problems (i, j, k) and compute the gradient of Eq.21. In this case, this will result in updates to the regression weights of each problem, $(Z_{X_i}, Z_{X_j}, Z_{X_k})$, as well as to the internal tree structure. It should be noted that from Eq.15, the representation of each regression weight is constrained by the location of its parent, which enforces the desired multi-task regularization effect.

Representation learning: In this application, we want to learn an embedding of words in the hyperbolic space and jointly discover a tree structured clustering of the words and so use the GloVe objective [36] for $C_{problem}(\mathcal{X})$.

6 EXPERIMENTS

We provide qualitative and quantitative evaluations of our method. We evaluate our method over several hierarchical clustering datasets, We demonstrate the efficacy of our method in a downstream multi-task learning application using end-to-end optimization. We qualitatively evaluate the performance of our method on representation learning using a word-embedding task.

6.1 Hierarchical Clustering Evaluation

We evaluate the performance of our method of using continuous trees for hierarchical clustering against state-of-the-art hierarchical clustering methods that search over the discrete space of trees. Following previous work [22, 28], we evaluate the quality of our hierarchical clusterings using dendrogram purity (DP). Given a ground truth flat clustering $C^\star$ of a dataset X, dendrogram purity of a tree structure $\mathcal{T}$ is:

$$\text{DP}(\mathcal{T}) = \frac{1}{|\mathcal{P}^\star|} \sum_{C^\star \in \mathcal{C}^\star} \sum_{(x_i, x_j) \in C^\star \times C^\star} \texttt{pur}(\texttt{lca}(x_i, x_j), C^\star), \tag{22}$$

where $\mathcal{P}^\star = \{(x_i, x_j) | C^\star(x_i) = C^\star(x_j)\}$, $C^\star(x)$ denotes the ground truth cluster membership of x and $\texttt{pur}(n, C^\star)$ is the purity of the cluster represented by n with respect to the cluster C (i.e. the fraction of n's descendant leaves are in the ground truth cluster $C^\star$). In words, this is the average purity of the least common ancestors of pairs of points belonging to the same ground truth cluster. Trees with high DP scores contain nodes that are similar to clusters in the ground truth flat partition.

We follow the experimental setup of Kobren et al. [28] and evaluate our method against the following approaches: **PERCH** [28] is a state-of-the-art large scale clustering algorithm that incrementally builds a tree structure by inserting points as a sibling of their nearest neighbor and performing local tree re-arrangements; **BIRCH** [54] is a top-down hierarchical clustering algorithm with a dynamically growing tree structure; **Hierarchical K-Means (HKMeans)** is a recursive application of Lloyd's algorithm; **Hierarchical Agglomerative Clustering (HAC)** is a widely used and exceedingly performant method that builds a trees structure in a bottom-up way by recursively merging the two sub-trees with the highest similarity value according to its linkage function.

To demonstrate the effectiveness of our approach, we evaluate the performance of our model on three classic hierarchical clustering benchmark datasets as well as on three large scale datasets [28]: **Glass** samples of glass[1]; **Spambase** spam emails [2]; **Digits** samples of handwritten digits [3]; **CovType** forest cover types; **ALOI** (Amsterdam Library of Object Images) contains images and is used as an extreme classification benchmarks; **ImageNet ILSVRC 2012**

[1]https://archive.ics.uci.edu/ml/datasets/glass+identification
[2]https://archive.ics.uci.edu/ml/datasets/spambase
[3]https://archive.ics.uci.edu/ml/datasets/optical\+recognition+of+handwritten+digits

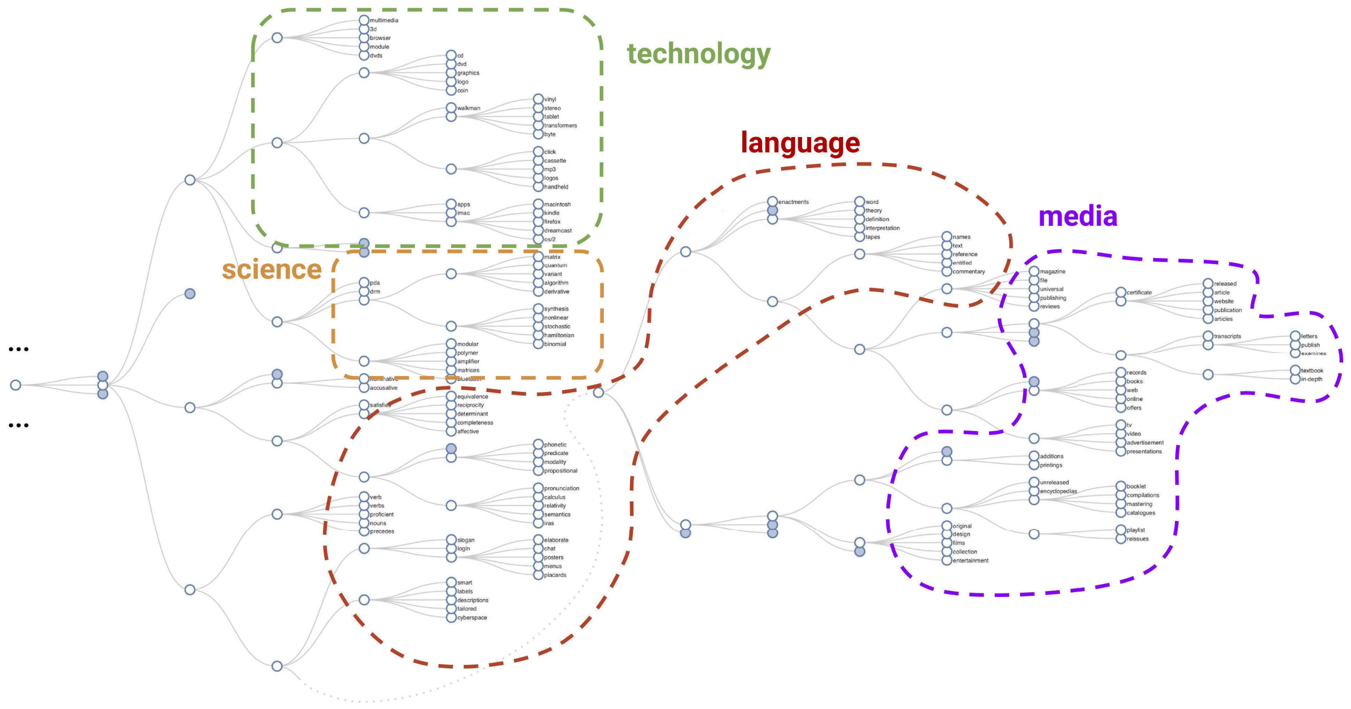

Figure 3: Sampled sub-tree of the learned hierarchical tree using GloVe. We can clearly identify groups of semantically similar words, such as 'science', 'technology', 'language', and 'media'. On a higher level, we also observe that 'science' and 'technology' sub-trees, which are semantically very close, merge into a larger sub-tree. The same happens for 'language' and 'media'. Some nodes were collapsed and leaf nodes were sub-sampled to allow for better visualization of the tree.

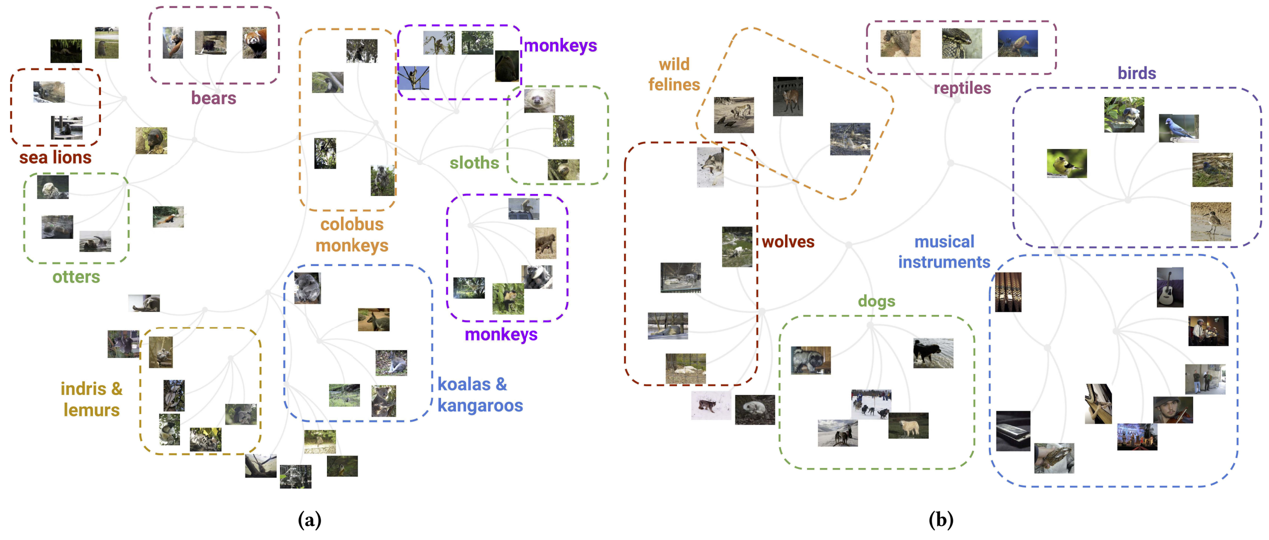

Figure 4: Sampled sub-trees of the learned hierarchical tree on the ImageNet dataset. Both images above show sampled sub-tree of the full learned tree with 1-million leaf nodes. In 4a we can observe both how photos of the same animals were clustered together into the lower sub-trees and that as we go up in the sub-trees, the dendrogram merges together sub-trees of similar animal species. In 4b we see clear clusters of specific animals and animal groups, such as 'dogs', 'wolves' and 'birds'. It's interesting to notice, for instance, how the clusters for 'dogs' and 'wolves' are close together, and how among the different clusters of animal groups, the one closest to 'musical instruments' is the 'birds' cluster.

	Glass	**Digits**	**Spambase**	**CovType**	**ALOI**	**ImageNet ILSVRC**
# Ground truth clusters	6	10	2	7	1000	1000
# Data points	214	200	4601	500K	108K	1.3M
Dimensionality	10	64	57	54	128	2048
gHHC (This Paper)	0.463 ± 0.0016	0.675 ± 0.015	0.614 ±0.009	0.444 ± 0.005	**0.462 ± 0.0004**	**0.367 ± 0.0001**
Perch	0.474 ± 0.017	0.614 ± 0.033	0.611 ± 0.0131	0.45 ± 0.004	0.44 ± 0.004	0.21 ± 0.017
BIRCH	0.429 ± 0.013	0.544 ± 0.054	0.595 ± 0.013	0.44 ± 0.002	0.32 ±0.002	0.11 ± 0.006
HKMeans	**0.508 ± 0.008**	0.586 ± 0.054	0.626 ± 0.00	0.44 ± 0.001	0.44 ± 0.001	0.11 ± 0.003
HAC-Centroid	0.47	0.594	0.628	-	-	-
HAC-Avg	0.501	**0.7836**	**0.629**	-	-	-

Table 1: Dendrogram Purity. Results for competing approaches from [28] using each algorithm's optimal setting. Bold indicates the best performing method. On small-scale problems (first three datasets) HAC performs very well. As the number of ground truth clusters, dimensionality and data points increases, our algorithm outperforms state of the art methods.

representations of each image from the last layer of the Inception neural network.

For all datasets, we use the same settings of the hyperparameters of gHHC. We use: a Poincaré ball of the same dimension d as the original space of the data, a learning rate of 0.01, a batch size of 100, training episodes of length 5000. For the Glove dataset, Adam [25] is to optimize the embedding of points in $\mathbb{D}^d$. Similarly for the Digits dataset, which requires learning a representation in hyperbolic space. For the first three, we use a number of internal nodes = 64, for CovType we use 5000, and for the ALOI and ImageNet we use 40K and 20K internal nodes respectively. For baselines we report the best results obtained using hyperparameter settings from [28].

Table 1 presents the results for this experiment. As evident, HAC is optimal for small-scale problems, however, its quadratic complexity prevents is from scaling to the rest of the large scale datasets. Furthermore, among all scalable clustering methods, our method consistently performs competitively, and significantly outperforms state of the art over the ImageNet dataset by around 15 points. We should note here that for CovType and Spambase, there is no significant winner. For ALOI, PERCH used a complete binary tree while our method used only 40K internal nodes. For ImageNet, both methods used a reduced number of nodes, PERCH uses 25K while our used 20K nodes. We hypothesize that our increased performance is due to gHHC's ability to update internal nodes in mini-batch fashion without making the incremental hard decisions that are difficult with a small number of internal nodes.

6.2 Multi-Task Learning

In this section, we show the efficacy of the end-to-end optimization of regression weights and tree structure in a hierarchical multi-task learning problem as described in Section 5. We use the school dataset which is the standard testbed for multi-task regression. It consists of the examination scores of 15362 students from 139 secondary schools in London during the years 1985, 1986 and 1987. Thus there are 139 tasks, and each example has the year of the examination, four school-specific and three student-specific attributes. We follows the experimental setup and split in [56]. We replace each categorical attribute with one binary variable for each possible attribute value resulting in 27 attributes per example. We evaluate the performance using the measure of percentage explained variance which is defined as the percentage of one minus nMSE (normalized mean squared error) [56]. We compare against state the art Multi-task relationship learning (MTRL [56]) that learns the structure of inter-task relationship and against a single-task learning (STL) baseline that solves each regression problem independently. As shown in Table 2, we outperform the baselines.

Method	Explained Variance (higher better)
STL	23.5
MTRL [56]	29.9
gHHC (this paper)	**38.4**

Table 2: Results over the school dataset.

6.3 Qualitative Evaluation

We examine, qualitatively, the quality of the tree structures learned by our method. We show two scenarios, one in which the input representation is fixed over the ImageNet dataset (Figure 4) and one in which the input representation is jointly learned with the tree structure as described in Section 5 using GloVe [36] (Figure 3). In both cases, we see that our method produces meaningful structure.

7 RELATED WORK

Hierarchical clustering is a widely studied problem theoretically, in machine learning, and in applications. Apart from the work on Dasgupta's cost and related costs [49], there has been much work on probabilistic models that have been used to describe the quality of hierarchical clusterings. Much of this work uses Bayesian non-parametric models to describe tree structures [1]. There has also been some work using discriminative graphical models to measure the quality of a clustering [50]. These cost functions come with their own inductive biases and the optimization of them with similar techniques to this paper could be interesting future work. Gradient-based methods are prevalent in flat clustering such as stochastic and mini-batch k-means [42].

Hyperbolic geometry has received much recent interest in the machine learning community. It has been studied in the embedding of taxonomies and graphs [19, 35, 40, 48]. It has also been used to visualize hierarchical clustering [4, 31]. Recent work [47] uses hyperbolic space to discover, in an unsupervised way, the relationships between words. However, to the best of our knowledge, no work addressed learning latent tree in an unsupervised fashion in the hyperbolic space as we do in this paper.

Differentiable clustering methods have also been used as regularizes in deep auto-encoders as well as deep supervised models that jointly optimize flat clustering objectives with their models core objective. Goyal et al. [20] similarly uses a nested Chinese Restaurant Process model jointly with an autoencoder for videos. Other problems traditionally solved with clustering, such as within-document coreference have used problem specific representations of clusterings that support gradient-based learning of similarity between data points and assignment of points to clusters [33]. Work on extreme classification also learns tree structures but typically use labeled data to discover these structures [24].

8 CONCLUSION

In this paper, we presented a novel hierarchical clustering algorithm that uses gradient-based optimization and a continuous representation of trees in the Poincaré ball. We showed its ability to perform hierarchical clustering on large scale data. We also showed how our model can be jointly optimized with multi-task regression. In future work, we hope to explore gradient-based optimization of tree structures in deep latent-variable models.

ACKNOLWEDGEMENTS

We thank Ari Kobren for his helpful discussions and work on earlier versions of gradient-based hierarchical clustering as well as Nishant Yadav, Javier Burroni, and the anonymous reviewers for their helpful comments and feedback. This work was supported in part by the Center for Data Science and the Center for Intelligent Information Retrieval, in part by the National Science Foundation under Grant No. NSF-1763618. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

REFERENCES

[1] R. P. Adams, Z. Ghahramani, and M. I. Jordan. 2010. Tree-Structured Stick Breaking for Hierarchical Data.
[2] O. Bachem, M Lucic, H. Hassani, and A. Krause. 2016. Fast and provably good seedings for k-means. *NeurIPS.*
[3] M.F. Balcan, A. Blum, and S. Vempala. 2008. A discriminative framework for clustering via similarity functions. *STOC.*
[4] J. Bingham and S. Sudarsanam. 2000. Visualizing large hierarchical clusters in hyperbolic space. *Bioinformatics.*
[5] C. Blundell, Y. W. Teh, and K. A. Heller. 2010. Bayesian rose trees. *UAI.*
[6] F. Boguná, M.and Papadopoulos and D. Krioukov. 2010. Sustaining the internet with hyperbolic mapping. *Nature communications.*
[7] S. Bonnabel. 2013. Stochastic gradient descent on Riemannian manifolds. *IEEE Trans. Automat. Control.*
[8] P. F. Brown, P. V Desouza, R. L Mercer, V. J D. Pietra, and J. C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics.*
[9] M. Charikar and V. Chatziafratis. 2017. Approximate hierarchical clustering via sparsest cut and spreading metrics. *SODA.*
[10] M. Charikar, V. Chatziafratis, and R. Niazadeh. 2019. Hierarchical Clustering better than Average-Linkage. *SODA.*
[11] M. Charikar, V. Chatziafratis, R. Niazadeh, and G. Yaroslavtsev. 2019. Hierarchical Clustering for Euclidean Data. *AISTATS.*
[12] K. Clark and C. D Manning. 2016. Improving coreference resolution by learning entity-level distributed representations. *ACL.*
[13] V. Cohen-Addad, V. Kanade, and F. Mallmann-Trenn. 2017. Hierarchical clustering beyond the worst-case. *NeurIPS.*
[14] V. Cohen-Addad, V. Kanade, F. Mallmann-Trenn, and C. Mathieu. 2018. Hierarchical clustering: Objective functions and algorithms. *SODA.*
[15] A. Culotta, P. Kanani, R. Hall, M. Wick, and A. McCallum. 2007. Author disambiguation using error-driven machine learning with a ranking loss function. *IIWeb.*
[16] S. Dasgupta. 2016. A cost function for similarity-based hierarchical clustering. *STOC.*
[17] E. Emamjomeh-Zadeh and D. Kempe. 2018. Adaptive hierarchical clustering using ordinal queries. *SODA.*
[18] H. Fichtenberger, M. Gillé, M. Schmidt, V. Schwiegelshohn, and C. Sohler. 2013. BICO: BIRCH meets coresets for k-means clustering. *ESA.*
[19] O.E. Ganea, G. Bécigneul, and T. Hofmann. 2018. Hyperbolic Entailment Cones for Learning Hierarchical Embeddings. *ICML.*
[20] P. Goyal, Z. Hu, X. Liang, C. Wang, and E. P. Xing. 2017. Nonparametric variational auto-encoders for hierarchical representation learning. *ICCV.*
[21] V. Guillemin and A. Pollack. 2010. *Differential topology.*
[22] K. A Heller and Z. Ghahramani. 2005. Bayesian hierarchical clustering. *ICML.*
[23] J. Himberg, A. Hyvärinen, and F. Esposito. 2004. Validating the independent components of neuroimaging time series via clustering and visualization. *Neuroimage.*
[24] Y. Jernite, A. Choromanska, and D. Sontag. 2017. Simultaneous learning of trees and representations for extreme classification and density estimation. *ICML.*
[25] D. P Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. *ICLR.*
[26] R. Kleinberg. 2007. Geographic routing using hyperbolic space. *INFOCOM.*
[27] D. A Knowles and Z. Ghahramani. 2011. Pitman-Yor diffusion trees. *UAI.*
[28] A. Kobren, N. Monath, A. Krishnamurthy, and A. McCallum. 2017. A hierarchical algorithm for extreme clustering. *KDD.*
[29] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguná. 2010. Hyperbolic geometry of complex networks. *Physical Review E.*
[30] A. Krishnamurthy, S. Balakrishnan, M. Xu, and A. Singh. 2012. Efficient active algorithms for hierarchical clustering. *ICML.*
[31] J. Lamping and R. Rao. 1994. Laying out and visualizing large trees using a hyperbolic space. *UIST.*
[32] H. Lee, M. Recasens, A. Chang, M. Surdeanu, and D. Jurafsky. 2012. Joint entity and event coreference resolution across documents. *EMNLP.*
[33] K. Lee, L. He, M. Lewis, and L. Zettlemoyer. 2017. End-to-end neural coreference resolution. *EMNLP.*
[34] B. Moseley and J. Wang. 2017. Approximation Bounds for Hierarchical Clustering: Average Linkage, Bisecting K-means, and Local Search. *NeurIPS.*
[35] M. Nickel and D. Kiela. 2017. Poincaré embeddings for learning hierarchical representations. *NeurIPS.*
[36] J. Pennington, R. Socher, and C. D. Manning. 2014. GloVe: Global Vectors for Word Representation. *EMNLP.*
[37] L. Ratinov and D. Roth. 2009. Design challenges and misconceptions in named entity recognition. *ACL.*
[38] S. Rendle, Z. Freudenthaler, C.and Gantner, and L. Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. *UAI.*
[39] S. Roy and S. Pokutta. 2016. Hierarchical clustering via spreading metrics. *NeurIPS.*
[40] F. Sala, C. De Sa, A. Gu, and C. Ré. 2018. Representation tradeoffs for hyperbolic embeddings. *ICML.*
[41] R. Sarkar. 2011. Low distortion delaunay embedding of trees in hyperbolic plane. *GD.*
[42] D. Sculley. 2010. Web-scale k-means clustering. *WWW.*
[43] J. Seo and B. Shneiderman. 2002. Interactively exploring hierarchical clustering results [gene identification]. *Computer.*
[44] T. Sørlie, C. M Perou, R. Tibshirani, T. Aas, S. Geisler, H. J.sen, T. Hastie, M. B Eisen, M. Van De Rijn, S. S Jeffrey, et al. 2001. Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications. *PNAS.*
[45] M. Spivak. 1979. A comprehensive introduction to differential geometry, Publish or Perish.
[46] E. Strubell, P. Verga, D. Andor, D. Weiss, and A. McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. *EMNLP.*
[47] Alexandru Tifrea, Gary Becigneul, and Octavian-Eugen Ganea. 2019. Poincare Glove: Hyperbolic Word Embeddings. *ICLR.*
[48] T. D. Q. Vinh, Y. Tay, S. Zhang, G. Cong, and X.-L. Li. 2018. Hyperbolic Recommender Systems. *arxiv.*
[49] D. Wang and Y. Wang. 2018. An Improved Cost Function for Hierarchical Cluster Trees. *arXiv.*
[50] M. Wick, S. Singh, and A. McCallum. 2012. A discriminative hierarchical model for fast coreference at large scale. *ACL.*
[51] D. H Widyantoro, T. R Ioerger, and J. Yen. 2002. An incremental approach to building a cluster hierarchy. *ICDM.*
[52] A. R Zamir, A. Sax, W. Shen, L. J Guibas, J. Malik, and S. Savarese. 2018. Taskonomy: Disentangling task transfer learning. *CVPR.*
[53] H. Zhang, S. J Reddi, and S. Sra. 2016. Riemannian SVRG: Fast stochastic optimization on Riemannian manifolds. *NeurIPS.*
[54] T. Zhang, R. Ramakrishnan, and M. Livny. 1996. BIRCH: A new data clustering algorithm and its applications. *SIGMOD.*
[55] Y. Zhang, A. Ahmed, V. Josifovski, and A. Smola. 2014. Taxonomy discovery for personalized recommendation. *ICDM.*
[56] Y. Zhang and D.-Y. Yeung. 2010. A Convex Formulation for Learning Task Relationships in Multi-Task Learning citation. *UAI.*
[57] Y. Zhao and G. Karypis. 2002. Evaluation of hierarchical clustering algorithms for document datasets. *CIKM.*