A GEOMETRY-AWARE ALGORITHM TO LEARN HIERARCHICAL EMBEDDINGS IN HYPERBOLIC SPACE

Zhangyu Wang *
Alibaba Inc.
Hangzhou, China 310000
zhangyu.wzy@alibaba-inc.com

Lantian Xu
Carnegie Mellon University
Pittsburgh, PA, USA 15213
lxu2@andrew.cmu.edu

Zhifeng Kong
University of California San Diego
La Jolla, CA, USA 92092
z4kong@eng.ucsd.edu

Weilong Wang
Purdue University
West Lafayette, IN, USA, 47906
wang4167@purdue.edu

Xuyu Peng, Enyang Zheng
Alibaba Inc.
Hangzhou, China 310000
{xijiu.pxy, enyang.zhengey}@alibaba-inc.com

ABSTRACT

Hyperbolic embeddings are a class of representation learning methods that offer competitive performances when data can be abstracted as a tree-like graph. However, in practice, learning hyperbolic embeddings of hierarchical data is difficult due to the different geometry between hyperbolic space and the Euclidean space. To address such difficulties, we first categorize three kinds of illness that harm the performance of the embeddings. Then, we develop a geometry-aware algorithm using a dilation operation and a transitive closure regularization to tackle these illnesses. We empirically validate these techniques and present a theoretical analysis of the mechanism behind the dilation operation. Experiments on synthetic and real-world datasets reveal superior performances of our algorithm.

1 INTRODUCTION

Learning data representation is important in machine learning as it provides a metric space that reveals or preserves inherent data structure (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017; Hoff et al., 2002; Grover & Leskovec, 2016; Perozzi et al., 2014; Nickel et al., 2011; Bordes et al., 2013; Riedel et al., 2013). Hyperbolic embeddings, a class of hierarchy representation methods, have shown competitive performances when data can be abstracted as a graph (Chamberlain et al., 2017; Davidson et al., 2018; Ganea et al., 2018a; Gu et al., 2018; Tifrea et al., 2018).

*The author’s corresponding email is zhangyuwang@ucsb.edu. The work is done during an internship at Alibaba Inc.
In this work, we focus on the following embedding task. Let $D$ be a dataset incorporated with a set of hierarchical relations represented as edges in a tree-like graph $G$. The goal is to learn an embedding $\Theta$ in the hyperbolic space by drawing positive and negative samples of edges from the graph such that $\Theta$ preserves the edge relationships, which are reflected by the order of similarity between data pairs. The formal problem statement is presented in Section 2.

Theoretically, hyperbolic space, such as the Poincaré Ball model, benefit from high representational power due to their negative curvatures (Nickel & Kiela 2017; Sala et al. 2018). This observation has motivated research on solving real-world problems in hyperbolic space. For datasets with an observed structure, hyperbolic space can embed the data and preserve the structure with arbitrarily low distortion (Nickel & Kiela 2017; Chamberlain et al. 2017; Nickel & Kiela 2018; Ganea et al. 2018b; Chami et al. 2019c). For datasets with a latent structure, especially those obeying the power-law, hyperbolic space can provide a natural metric such that finer concepts are embedded into areas allowing more subtlety (Tifrea et al. 2018; Leimeister & Wilson 2018; Le et al. 2019).

Despite the theoretical advantages of hyperbolic embeddings, learning such representation in practice is difficult. Specifically, the following fundamental difficulties have not been well-studied in the literature. (1) Many properties of the Euclidean space do not transfer to hyperbolic space. For example, the latter generally do not have the scale or shift-invariance in the sense of preserving similarity orders. (2) Many nice properties exclusive to hyperbolic space may improve learning. However, it is unclear how to design algorithms to effectively incorporate these properties. (3) Optimization in hyperbolic space is (i) expensive due to a more sophisticated distance measure and (ii) unstable because gradient descent is performed on hyperbolic manifolds.

In this paper, we analyze these difficulties and provide a set of solutions to them. First, we define bad cases as improper relationship between nodes and edges. We then categorize them into capacity illness, inter-subtree illness, and intra-subtree illness. Formal definitions and intuitive visualizations are presented in Section 3. We present a theoretical analysis of local capacity, capacity illness, and their relationship in Section 4. We then develop an algorithm that reduces these illness in Section 5. The algorithm involves a dilation operation during the learning process, adding transitive closure edges of data to positive samples, and a re-weighting strategy. We conduct experiments on synthetic and real world datasets in Appendix 6. The results show that our algorithm achieves superior performances under various evaluation metrics.

2 Preliminaries

A hyperbolic space $H^d$ is a $d$-dimensional Riemannian manifold with a constant negative sectional curvature. In this paper, we focus on the Poincaré ball model. Let $B = B^d$ denote the $d$-dimensional Poincare ball. The distance between any two points $B_1, B_2 \in B^d$ is defined as

$$d(B_1, B_2) = \text{arcosh} \left( 1 + 2 \frac{||u - v||^2}{(1 - ||u||^2)(1 - ||v||^2)} \right),$$

where $u$ and $v$ are the Euclidean vectors of $B_1$ and $B_2$. In the rest of the paper, we denote the Poincaré distance by $d(\cdot, \cdot)$ and the Euclidean distance by $||\cdot||$. Given a set of points $V = \{x_i\}_{i=1}^n$ and the relation set $E \subseteq [n]^2$, the goal is to learn an embedding $f : V \rightarrow B$ that preserves the inherent structure. To achieve this goal, we define and minimize the following loss function $L$. For $(i, j) \in E$, define $N(x_i, x_j)$ as the set of negative samples of $(i, j)$. Let $\Theta = \{\theta_i\}_{i=1}^n$, where each $\theta_i \in B^d$ is the embedding of $x_i$. Define $d(x_i, x_j) = d(\theta_i, \theta_j)$. Then, the loss function is defined as

$$L(\Theta) = - \sum_{(i, j) \in E} \log \frac{e^{-d(x_i, x_j)}}{\sum_{x' \in N(x_i, x_j) \cup \{x_j\}} e^{-d(x_i, x')}} = - \sum_{(i, j) \in E} L_{i, j}(\Theta).$$

This objective can be optimized via Riemannian gradient descent (Nickel & Kiela 2017).

3 Illness

In this section, we formally define illness that harms the performance of hyperbolic embeddings and is hard to optimize. Let $\overrightarrow{AB}$ be a ground-truth edge in $G$, and $\overrightarrow{AB'}$ be the inferred edge from the
Algorithm 1 Geometry-Aware Algorithm

for $i = 1$ to $N_{\text{epoch}}$
    Compute local capacity according to equation (5)
    if local capacity is not sufficient then perform the dilation operation in equation (6)
    end if
    if $i \leq N_{\text{tc}}$ then loss $\leftarrow \mathcal{L}_{\text{tc}}(\Theta)$ according to equation (7)
    else loss $\leftarrow \mathcal{L}(\Theta)$ according to equation (2)
    end if
    Apply Riemannian gradient descent over loss
end for

hyperbolic embeddings, where $B' \neq B$. We call this situation illness with respect to $A$. Let $C$ be the nearest common ancestor of $B$ and $B'$. We categorize three kinds of illness according to the pairwise relationships among $B$, $B'$, and $C$. Formally, we define capacity illness, intra-subtree illness, and inter-subtree illness in Definition 1.

**Definition 1 (Categories of Illness).** We define the illness to be capacity illness if $B$ is the parent of $B'$. We define the illness to be an intra-subtree illness if $B$ is the ancestor but not the parent of $B'$. We define the illness to be an inter-subtree illness if $C \neq B$.

It is straightforward to see that the union of capacity illness and intra-subtree illness are exactly situations where $C = B$. Therefore, the above three kinds of illness are a partition of all illness. We visualize these three kinds of illness in Figure 3 in the appendix.

4 **LOCAL CAPACITY**

We define local capacity below and theoretically relate it to capacity illness.

**Definition 2 (Local Capacity).** Given a geodesic space $(\mathcal{X}, d)$ and a geodesic ball $S_d$ centered at $A \in \mathcal{X}$ with radius $r$. The local capacity of $(A, r)$ is defined as

$$\max \{|C| : C \in S_d; \forall C_1, C_2 \in C, C_1 \neq C_2, d(C_1, C_2) > d(C_1, A) \lor d(C_2, A)\}.$$  \hspace{1cm} (3)

Given a geodesic space $(\mathcal{X}, d)$ and a geodesic ball $S_d$ centered at $A \in \mathcal{X}$ with radius $r$. The local capacity of $(A, r)$ is defined as

$$\max \{|C| : C \in S_d; \forall C_1, C_2 \in C, C_1 \neq C_2, d(C_1, C_2) > d(C_1, A) \lor d(C_2, A)\}.$$  \hspace{1cm} (4)

For not very small $r$ and large $d$ we have the following bounds:

$$2^d e^{\frac{d^2 r}{2}} \geq A(d, \theta_r) \geq \sqrt{2\pi \log \frac{2}{\sqrt{3}} \cdot 2^{1-d} e^{\frac{d^2 r}{2}-1}}$$  \hspace{1cm} (5)

where $\theta_r = \arcsin (1/(2 \cosh(r/2)))$. Full derivations are in Appendix B.

5 **THE ALGORITHM**

In this section, we build a geometry-aware algorithm (Algorithm 1) targeting the three categories of illness by proposing the dilation operation and the transitive closure regularization.

**Dilation.** We define a mapping $g : B \rightarrow B$ as a $k$-dilation if for any $A \in B$:

$$d(O, g(A)) = k \cdot d(O, A).$$  \hspace{1cm} (6)

Notably, $g$ can be computed explicitly. For instance, a 2-dilation can be formulated as $g(A) = \frac{2}{1+\|A\|^2} A$. The dilation operation rescales the embedded structure so that each point is pushed to a location with sufficient local capacity. Given $A \in B$ with degree $k$, this operation helps increase the distance between $A$ and its $k$-nearest neighbor($r_A$), thus increase the local capacity of $(A, r_A)$. 

3
Figure 1: Visualizations of two-dimensional embeddings of a synthetic balance tree (156 nodes, 155 edges) learned by the baseline Poincaré embedding algorithm in Nickel & Kiela (2017) and our geometry-aware algorithm, respectively. Both algorithms are trained for 3000 epochs. The lines refer to ground-truth edges and the points refer to the learned hyperbolic embeddings. The red lines indicate bad cases where the embeddings fail to reconstruct these ground-truth edges.

Figure 2: Visualization of the learning process of two-dimensional Poincaré embeddings with (1) the baseline algorithm in the upper row, and (2) the geometry-aware algorithm in the lower row. The dataset, baseline model, and plotting settings are identical as in Figure 1. We plot intra-subtree and inter-subtree illness consistently existing throughout the entire 3000 epochs.

**Transitive closure regularization.** It contains the following two operations.

*Adding transitive closure edges.* The transitive closure edges $E_{tc}$ are edges between nodes and their non-parent ancestors. These edges are also considered as positive samples in addition to $E$ in the objective in equation 2. The purpose of adding these auxiliary edges is to push the subtrees apart so they are less likely to overlap in the Poincaré ball.

*Re-weighting.* We modify the weights of transitive closure edges to prevent overfitting in early (the first $N_{tc}$) epochs. Let $\eta_{tc}$ be a real number between 0 and 1. Then, the objective becomes

$$L_{tc}(\Theta) = L(\Theta) + \eta_{tc} \sum_{(i,j) \in E_{tc}} L_{i,j}(\Theta).$$

(7)

It is noteworthy that these operations are not admissible in the Euclidean space, where the local capacity of any $(A,r) \in \mathbb{R}^d \times \mathbb{R}$ is a constant with respect to $d$.

6 **Experiments**

We compare our algorithm to the baseline model (Nickel & Kiela, 2017) on the synthetic dataset in Figure 1 and Figure 2. Our method not only achieves perfect MAP (0.998) but also yields better reconstructed geometry. We do extensive experiments on multiple real-world datasets of various scales and characteristics in Appendix D. Results show our algorithm consistently outperform the baseline algorithms (Nickel & Kiela, 2017, 2018), especially on extremely bushy datasets.

---

1 Nickel & Kiela (2017)

2 Nickel & Kiela (2018)

3 Appendix D
7 Conclusion

In this paper, we analyze three categories of illness and develop a geometry-aware algorithm that targets at reducing these illnesses and improving performance. Our algorithm shows superior performance over baseline models on both synthetic and real world datasets.

References


A RELATED WORK

Learning in hyperbolic space is initially proposed by [Nickel & Kiela (2017)]. It is the most related work to our paper. Their method outperforms the Euclidean counterpart in low dimensions as to the task of learning embeddings for edge reconstruction. However, since their algorithm is directly adapted from the Euclidean space, it does not naturally leverage potentially useful geometrical properties of hyperbolic space (see Section 2). As a consequence, there remain many bad cases even after convergence (see Figure 1a).

A series of work directly incorporate properties of hyperbolic space via optimization ([Wilson & Leimeister, 2018; Bonnabel, 2013; Absil et al., 2009; Afsari et al., 2013]). Specifically, Nickel & Kiela (2018) conduct training in the Lorentz space with a closed-form expression of the geodesics on the hyperbolic manifold. However, since the learning objective equation 2 is highly non-convex, obtaining more accurate gradients does not completely solve the problem.

Another group of work either implement hyperbolic versions of commonly used neural network modules (Ganea et al., 2018c; Gulcehre et al., 2018; Chami et al., 2019b), or design models specifically tailored for hyperbolic space (Vulić & Mrkšić, 2017; Le et al., 2019; Cho et al., 2019; Leimeister & Wilson, 2018; Weber et al., 2020; Chami et al., 2019a). These methods are task-specific and thus expensive to deploy in downstream applications.

Apart from the above learning approaches, Sala et al. (2018) presents a combinatorial algorithm that achieves better performance than [Nickel & Kiela, 2017; 2018] with even lower dimensions. The core idea is to extend the 2-dimensional results of Sarkar (2011) to arbitrary dimensions. However, this algorithm suffers from three vital weaknesses: (1) it requires complete information of the graph; (2) it is sensitive to addition/removal of data; and (3) most critically, it involves discrete operations and thus does not have gradients. Therefore, in scenarios where complete information is unavailable, the graph dynamically changes, or joint learning is needed, this approach does not suffice.

In this paper, we endorse the importance of leveraging geometrical properties in learning unsupervised hyperbolic embeddings. Based on this intention, we develop a geometry-aware algorithm that improves embedding performances, which, to the best of our knowledge, is original.

B ILLNESS

According to Definition 2, for $A \in V$ and a radius $r$, if $|\{C : C$ is a child of $A, d(A, C) \leq r\}|$ exceeds the local capacity of $(A, r)$, then capacity illness must exist. To obtain bounds on local...
capacity, we first bound the $r$-packing number of a geodesic sphere centered at point $A \in B$ with radius $r$.

First, the packing number of the geodesic sphere centered at the origin $O$ is the same as that of $A$. To see why we use a result from Paupert (2016):

$$\text{Isom}(B) = \text{M"ob}(B).$$  \hfill (8)

That is, the group of isometries from $B$ to itself coincide with the group of all M"obius transformations preserving $\overline{B}$. In the Poincaré ball model, $\text{M"ob}(B)$ is generated by inversions in generalized spheres $S'$ such that $S' \perp \partial B$, therefore once we extend $\overline{OA}$ to $C$ with $||OC||^2 - 1 = ||OC|| \cdot ||CA||$, then by taking the restriction in $\overline{B}$ of the inversion in generalized sphere $S'$ centered at $C$ we get an isometry from $\overline{B}$ to itself which maps $A$ to $O$. Specifically, it is an isometry between any geodesic sphere centered at $A$ and the geodesic sphere centered at the origin $O$ (with the same radius $r$).

Then, we compute the $r$-packing number of the geodesic sphere $S_r$ centered at $O$ with Poincaré radius $r$. For $B_1, B_2 \in S_r$, let $u = \overline{OB_1}, v = \overline{OB_2}$ and $r$ be the Poincaré norm of $u$. Then, as long as the angle $\theta$ between $u$ and $v$ satisfies

$$\theta \geq \theta_r = 2 \arcsin \left( \frac{1}{2 \cosh(r/2)} \right),$$  \hfill (9)

we have $d(B_1, B_2) \geq d(B_i, O), i = 1, 2$. For not very small $r, \theta_r \approx 2e^{-r/2}$. Then, the $r$-packing problem is equivalent to evaluating the size of the largest spherical code of angle $\theta_r$ in dimension $d$, defined as $\mathcal{A}(d, \theta_r)$. According to Jenssen et al. (2018), we have

$$\mathcal{A}(d, \theta) \geq (1 + o(1)) \frac{c_\theta \cdot d}{s_d(\theta)},$$  \hfill (10)

where $c(\theta) = \log \frac{\sin^2(\theta)}{\sqrt{(1 - \cos \theta)(1 + 2 \cos \theta)}} \approx \log \frac{2}{\sqrt{3}}$ for small $\theta$, $s_d(\theta) = (1 + o(1)) \frac{\sin^{d-1} \theta}{\sqrt{2 \pi d \cos \theta}}$. Specifically, when $d \leq 16$ with small $\theta$, Shannon (1959) provides a better bound:

$$\mathcal{A}(d, \theta) \geq \frac{1}{s_d(\theta)} = (1 + o(1)) \frac{\sqrt{2\pi d} \cdot \cos \theta}{\sin^{d-1} \theta}.$$  \hfill (11)

To sum up, for not very small $r$ we have the following lower bounds under different dimensions:

$$\mathcal{A}(d, \theta_r) \geq \begin{cases} \pi e^{\frac{r}{2}} & d = 2 \\ \sqrt{2\pi d} \cdot 2^{1-d} e^{\frac{d-1}{2}} r & 3 \leq d \leq 16 \\ \sqrt{2\pi \log \frac{2}{\sqrt{3}} \cdot d^2} \cdot \frac{1}{2^{d-d} e^{\frac{d-1}{2}}} r & d \geq 17 \end{cases}.$$  \hfill (12)

As for the upper bound, according to Kabatiansky & Levenshtein (1978),

$$\mathcal{A}(d, \theta) \leq e^{\phi(\theta)d(1+o(1))},$$  \hfill (13)

where $\phi(\theta) > -\log \sin \theta$ is a certain function. Therefore,

$$\mathcal{A}(d, \theta_r) \leq \begin{cases} \pi e^{\frac{r}{2}} & d = 2 \\ 2^d e^{\frac{r}{2}} & d \geq 3 \end{cases}.$$  \hfill (14)

Note that by considering the extension of radius, one can define local capacity on geodesic balls instead of spheres, which leads to the same conclusion.
D EXPERIMENTAL SETTINGS AND OVERVIEW

We apply our algorithm to both a synthetic dataset and real-world datasets on the graph reconstruction task. We evaluate the performance by mean average precision (MAP) and Mean Rank (MR) defined below. For \( A \in V \) with degree \( \deg(A) \) and neighborhood \( \mathcal{N}_A = \{ B_1, \ldots, B_{\deg(A)} \} \), let \( R_{A,B_i} \) be the smallest subset of \( V \) containing \( B_i \) and all points closer to \( A \) than \( B_i \). Then, the MAP is defined as

\[
\text{MAP}(f) = \frac{1}{|V|} \sum_{A \in V} \frac{1}{\deg(A)} \sum_{i=1}^{\left| \mathcal{N}_A \right|} \frac{\left| \mathcal{N}_A \cap R_{A,B_i} \right|}{\left| R_{A,B_i} \right|}.
\]

and the MR is defined as

\[
\text{MR}(f) = \frac{1}{|V|} \sum_{A \in V} \sum_{i=1}^{\left| \mathcal{N}_A \right|} (\left| R_{A,B_i} \right| - i).
\]

In addition, we report the number of three kinds of illness defined in Definition 1 after the algorithm converges.

We run baseline algorithms \cite{NickelKiela2017, NickelKiela2018} and our algorithm on a synthetic dataset, Yelp Challenge (Tree) (see Table 2), WordNet Verbs (Tree) (see Table 3), WordNet Nouns (Tree) (see Table 4), Commodity Catalog (Tree) (see Table 5) and WordNet Nouns (Closure) (see Table 6). We report the reconstruction Mean Rank, MAP, number of capacity errors, number of intra-subtree errors and number of inter-subtree errors respectively.

A key point to notice is that we focus on tree datasets instead of general DAG or transitive closures of trees. In terms of the objective (reducing MAP and MR), learning a tree structure is much harder because the size of the neighborhood set is as few as one. Experiments validate this statement: we apply the baseline algorithm \cite{NickelKiela2017} to the same WordNet Noun Hierarchy dataset \cite{NickelKiela2017} where the transitive closure edges are removed. The performances in terms of MAP and MR significantly drop compared to the numbers reported in \cite{NickelKiela2017} (See Table 2).
E Synthetic Dataset Experiments

Figure 4: Number of different kinds of illness under different $\eta_{tc}$. Base denotes the baseline algorithm.

Figure 5: Performances of our algorithm under different hyperparameter settings on the synthetic tree. Base denotes the baseline algorithm, GA is our algorithm where DL is the dilation operation, RW is the re-weighting strategy followed by the threshold epoch number $N_{tc}$.

We synthesize a balanced tree $T$ of 5 layers including the root node. Each non-leaf node in $T$ has 5 children. The edges are directed, pointing from children to parents. We illustrate our algorithm on this synthetic dataset.

We compare our algorithm (Algorithm 1) to the baseline Poincaré embedding algorithm (Nickel & Kiela, 2017). In both algorithms, we set the dimension to be 2, learning rate to be 0.5, batch size to be 50, and the number of negative samples $m$ to be 50.

The comparison between learning procedures is presented in Figure 2. As demonstrated, our algorithm learns visually more balanced embeddings with less illness. Quantitative results of the comparison, including the number of illness, MAP and MR, are presented in Figure 4 and Figure 5. We compare
different $N_{tc}$ in Figure 5 and different $\eta_{tc}$ in Figure 4. Our algorithm produces less illness than the baseline algorithm and achieves the highest MAP and MR.

Figure 4 shows our explorations on how to use transitive closure edges. They tend to increase the overall effective gradient magnitude and draw vertices of a same subtree tightly together. Therefore, this could help reduce capacity illness quickly (See Figure 4(a)) and eliminate inter-subtree illness (See Figure 4(c)). However, it might also confuse the ground-truth tree edges with the added ones, thus increasing intra-subtree illness (See Figure 4(b)). When we assign weights to the transitive closure edges, this side effect is mitigated: $\eta_{tc} = 0.2$ yields the best results.

Figure 5 shows our explorations on how to use dilation and reweighting. These two operations should be applied after certain epochs of training so that 1) the subtrees are pushed relatively far from each other to ensure the dilation operation will push vertices in the appropriate directions, and 2) before the vertices are already pushed to places with sufficient capacity. Empirically we find this threshold epoch number $N_{tc} = 300$ yields the best results.

---

A similar idea is shown in the burn-in stage of (Nickel & Kiela, 2017).
F.1 Dataset Statistics

Table 1 displays statistics of several datasets used in our experiments (including the Yelp Challenge Dataset, the Commodity Catalog Dataset, and the WordNet dataset). We report the number of nodes $|\mathcal{V}|$, the number of edges $|\mathcal{E}|$, maximum degree, variance of all degrees, and tree depths. We make subjective remarks to their size and characteristics.

| Dataset                          | $|\mathcal{V}|$ | $|\mathcal{E}|$ | Max deg | Std of deg | Depth | Size     | Characteristics          |
|---------------------------------|--------------|---------------|----------|------------|-------|----------|--------------------------|
| Yelp Challenge (T)              | 1587         | 1586          | 194      | 7.9        | 4     | small    | shallow                 |
| Commodity Catalog (T)           | 134812       | 134811        | 726      | 26.4       | 6     | large    | shallow; bushy           |
| WordNet Verbs (T)               | 13542        | 13541         | 361      | 9.1        | 12    | medium   | deep                    |
| WordNet Nouns (T)               | 82115        | 82114         | 666      | 6.6        | 15    | large    | deep                    |
| WordNet Nouns (C)               | 82115        | 769130        | 82114    | 913.6      | 15    | large    | deep; dense             |

Table 1: Dataset statistics (T: Tree; C: Closure)

F.2 Yelp Challenge Dataset (Tree) Results

The results for Yelp Challenge (Tree) are illustrated in Table 2. In all algorithms, we set the learning rate to be 1.0, batch size to be 10, and the number of negative samples $m$ to be 50.

<table>
<thead>
<tr>
<th></th>
<th>Dim</th>
<th>MR</th>
<th>MAP</th>
<th>Capacity</th>
<th>Intra</th>
<th>Inter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>27.612</td>
<td>0.107</td>
<td>1440</td>
<td>82</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2017)</td>
<td>1.681</td>
<td>0.855</td>
<td>292</td>
<td>45</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2018)</td>
<td>2</td>
<td>1.520</td>
<td>0.894</td>
<td>163</td>
<td>47</td>
<td>74</td>
</tr>
<tr>
<td>GA-DL (Ours)</td>
<td>1.351</td>
<td>0.926</td>
<td>86</td>
<td>65</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>GA-DL-RW (Ours)</td>
<td>1.202</td>
<td>0.914</td>
<td>162</td>
<td>54</td>
<td>37</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Dim</th>
<th>MR</th>
<th>MAP</th>
<th>Capacity</th>
<th>Intra</th>
<th>Inter</th>
<th>YELP CHALLENGE Reconstruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>11.320</td>
<td>0.326</td>
<td>1236</td>
<td>9</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2017)</td>
<td>1.063</td>
<td>0.988</td>
<td>1</td>
<td>0</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2018)</td>
<td>5</td>
<td>1.101</td>
<td>0.984</td>
<td>1</td>
<td>11</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>GA-DL (Ours)</td>
<td>1.062</td>
<td>0.987</td>
<td>3</td>
<td>9</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GA-DL-RW (Ours)</td>
<td>1.030</td>
<td>0.989</td>
<td>3</td>
<td>9</td>
<td>18</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Dim</th>
<th>MR</th>
<th>MAP</th>
<th>Capacity</th>
<th>Intra</th>
<th>Inter</th>
<th>yelp challenge reconstruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>1.528</td>
<td>0.910</td>
<td>183</td>
<td>1</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2017)</td>
<td>1.042</td>
<td>0.990</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2018)</td>
<td>10</td>
<td>1.064</td>
<td>0.987</td>
<td>0</td>
<td>9</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>GA-DL (Ours)</td>
<td>1.051</td>
<td>0.989</td>
<td>1</td>
<td>5</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GA-DL-RW (Ours)</td>
<td>1.015</td>
<td>0.993</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Yelp Challenge Dataset

https://www.yelp.com/dataset/documentation/main
F.3 WordNet Verbs Dataset (Tree) Results

The results for WordNet Verbs (Tree) are illustrated in Table 3. In all algorithms, we set the learning rate to be 1.0, batch size to be 10, and the number of negative samples \( m \) to be 50.

<table>
<thead>
<tr>
<th>Dim</th>
<th>MR</th>
<th>MAP</th>
<th>Capacity</th>
<th>Intra</th>
<th>Inter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>66.302</td>
<td>0.124</td>
<td>7366</td>
<td>3778</td>
<td>1879</td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2017)</td>
<td>8.088</td>
<td>0.521</td>
<td>4826</td>
<td>2582</td>
<td>1683</td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2018)</td>
<td>2</td>
<td>6.875</td>
<td>0.448</td>
<td>5459</td>
<td>1769</td>
</tr>
<tr>
<td>GA-DL (Ours)</td>
<td>8.319</td>
<td>0.510</td>
<td>5023</td>
<td>2361</td>
<td>1908</td>
</tr>
<tr>
<td>GA-DL-RW (Ours)</td>
<td>3.559</td>
<td>0.563</td>
<td>4543</td>
<td>2741</td>
<td>1399</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dim</th>
<th>MR</th>
<th>MAP</th>
<th>Capacity</th>
<th>Intra</th>
<th>Inter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>21.245</td>
<td>0.295</td>
<td>10383</td>
<td>1273</td>
<td>277</td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2017)</td>
<td>2.329</td>
<td>0.854</td>
<td>235</td>
<td>2321</td>
<td>454</td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2018)</td>
<td>5</td>
<td>2.389</td>
<td>0.853</td>
<td>285</td>
<td>2277</td>
</tr>
<tr>
<td>GA-DL (Ours)</td>
<td>2.195</td>
<td>0.878</td>
<td>211</td>
<td>1443</td>
<td>920</td>
</tr>
<tr>
<td>GA-DL-RW (Ours)</td>
<td>1.722</td>
<td>0.841</td>
<td>378</td>
<td>1306</td>
<td>1893</td>
</tr>
</tbody>
</table>

Table 3: WordNet Verbs

F.4 WordNet Verbs Nouns (Tree) Results

The results for WordNet Nouns (Tree) are illustrated in Table 4. In all algorithms, we set the learning rate to be 1.0, batch size to be 50, and the number of negative samples \( m \) to be 50.

<table>
<thead>
<tr>
<th>Dim</th>
<th>MR</th>
<th>MAP</th>
<th>Capacity</th>
<th>Intra</th>
<th>Inter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>151.321</td>
<td>0.235</td>
<td>46563</td>
<td>21942</td>
<td>2993</td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2017)</td>
<td>71.271</td>
<td>0.322</td>
<td>48844</td>
<td>17293</td>
<td>438</td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2018)</td>
<td>74.625</td>
<td>0.230</td>
<td>50950</td>
<td>18777</td>
<td>311</td>
</tr>
<tr>
<td>GA-DL (Ours)</td>
<td>19.313</td>
<td>0.481</td>
<td>35119</td>
<td>21158</td>
<td>444</td>
</tr>
<tr>
<td>GA-DL-RW (Ours)</td>
<td>2.869</td>
<td>0.697</td>
<td>13394</td>
<td>23907</td>
<td>27</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dim</th>
<th>MR</th>
<th>MAP</th>
<th>Capacity</th>
<th>Intra</th>
<th>Inter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>23.113</td>
<td>0.278</td>
<td>23978</td>
<td>31018</td>
<td>15745</td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2017)</td>
<td>41.014</td>
<td>0.324</td>
<td>49152</td>
<td>17276</td>
<td>154</td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2018)</td>
<td>38.395</td>
<td>0.241</td>
<td>50986</td>
<td>18690</td>
<td>100</td>
</tr>
<tr>
<td>GA-DL (Ours)</td>
<td>7.900</td>
<td>0.754</td>
<td>10647</td>
<td>13547</td>
<td>5551</td>
</tr>
<tr>
<td>GA-DL-RW (Ours)</td>
<td>2.738</td>
<td>0.722</td>
<td>9483</td>
<td>24844</td>
<td>32</td>
</tr>
</tbody>
</table>

Table 4: WordNet Nouns
F.5 Commodity Catalog (Tree) Results

The results for Commodity Catalog (Tree) are illustrated in Table 5. In all algorithms, we set the learning rate to be 1.0, batch size to be 10, and the number of negative samples $m$ to be 50.

<table>
<thead>
<tr>
<th>Dim</th>
<th>MR</th>
<th>MAP</th>
<th>Capacity</th>
<th>Intra</th>
<th>Inter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>163.044</td>
<td>0.024</td>
<td>132276</td>
<td>1480</td>
<td>700</td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2017)</td>
<td>86.759</td>
<td>0.063</td>
<td>130479</td>
<td>2001</td>
<td>174</td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2018)</td>
<td>50.393</td>
<td>0.082</td>
<td>127373</td>
<td>1968</td>
<td>238</td>
</tr>
<tr>
<td>GA-DL (Ours)</td>
<td>5.405</td>
<td>0.745</td>
<td>49837</td>
<td>1971</td>
<td>301</td>
</tr>
<tr>
<td>GA-DL-RW (Ours)</td>
<td>2.951</td>
<td>0.683</td>
<td>56922</td>
<td>7711</td>
<td>26</td>
</tr>
</tbody>
</table>

Table 5: Commodity Catalog

The Commodity Catalog (Tree) dataset is generated from real-world e-commerce data. It is extremely bushy. According to Sala et al. (2018), the hyperbolic space is capable of embed even extremely bushy trees. However, we find the baseline algorithm in Nickel & Kiela (2017) cannot fully exert such capability. This is because a bushy tree requires large local capacity, but the baseline algorithm takes very long time to reach such capacity and instead easily gets stuck at local optimum. Experimentally, the baseline algorithm does not perform well on such bushy dataset, while our algorithm with simple dilation yields significantly better results.

F.6 WordNet Nouns (Closure) Results

The results for WordNet Nouns (Closure) are illustrated in Table 6. In all algorithms, we set the learning rate to be 1.0, batch size to be 50, and the number of negative samples $m$ to be 50.

<table>
<thead>
<tr>
<th>Dim</th>
<th>MR</th>
<th>MAP</th>
<th>Capacity</th>
<th>Intra</th>
<th>Inter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>59.133</td>
<td>0.052</td>
<td>131806</td>
<td>1676</td>
<td>289</td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2017)</td>
<td>67.283</td>
<td>0.071</td>
<td>130212</td>
<td>1998</td>
<td>147</td>
</tr>
<tr>
<td>Nickel &amp; Kiela (2018)</td>
<td>36.836</td>
<td>0.112</td>
<td>124366</td>
<td>1964</td>
<td>196</td>
</tr>
<tr>
<td>GA-DL (Ours)</td>
<td>2.167</td>
<td>0.978</td>
<td>1871</td>
<td>1861</td>
<td>390</td>
</tr>
<tr>
<td>GA-DL-RW (Ours)</td>
<td>2.015</td>
<td>0.881</td>
<td>19902</td>
<td>5525</td>
<td>26</td>
</tr>
</tbody>
</table>

Table 6: WordNet Nouns Closure

---

3Noticeably, since the tree is bushy, adding transitive closure edges would largely increase intra-class illness, which makes MAP drop.

4Results of Nickel & Kiela (2017) are based on their official implementation and hyperparameters in [https://github.com/facebookresearch/poincare-embeddings](https://github.com/facebookresearch/poincare-embeddings).