LEARNING WEIGHTED PRODUCT SPACES REPRESENTATIONS FOR GRAPHS OF HETEROGENEOUS STRUCTURES

Tuc Nguyen
FPT Software, AIC
Hanoi, Vietnam
tucnv2@fsoft.com.vn

Dung D. Le
College of Engineering and Computer Science
VinUniversity
Hanoi, Vietnam
dung.ld@vinuni.edu.vn

The-Anh Ta
FPT Software, AIC
Hanoi, Vietnam
anhtt71@fsoft.com.vn

ABSTRACT

Graph-like data are heterogeneous with different types of nodes and edges. Heterogeneous structure in a graph means it has various structures such as trees, circles, and grid-like. In representation learning, it is important to learn embedding spaces whose geometry matches the underlying structure of the data. In the literature, an active research direction aims at using product spaces, which consists of Euclidean and non-Euclidean manifolds to represent data of varying curvatures. However, real-world data are usually heterogeneous and consist of a mixture of varying structures, requiring the representation learning process to flexibly select and combine the member spaces accordingly. Since previous works only consider a combination of embedding spaces with equal weights, in this paper, we propose a data-driven method to learn the embeddings in a weighted product space for graph data. Specifically, our model utilizes the topological information of the input graph to learn the weight of each component of the product spaces. Experiments on synthetic and real-world datasets show that our models produce better representations in terms of distortion measures, and perform better on tasks such as word similarity learning.

1 INTRODUCTION

Representation learning aims to find parameterizations of data distribution manifolds of low intrinsic dimension from data samples into high dimensional feature spaces (Bengio et al., 2013). Euclidean embedding spaces have been traditionally used in most representation learning models (Balazevic et al., 2019; Mikolov et al., 2013b). However, due to their uniform geometric structure, it has been realized that Euclidean spaces cannot faithfully represent various types of structure data such as tree-like (Nickel & Kiela, 2017) or circle-like (Wilson et al., 2014) graphs. Thus, increasing attention has been paid to feature embedding into non-Euclidean spaces (Wilson et al., 2014; Nickel & Kiela, 2017; Chami et al., 2019; Meng et al., 2019; Ganea et al., 2018). Recent works (Gu et al., 2018; Skopek et al., 2020) also show promising results when learning representation in product spaces of components with different geometries. This is mainly motivated by the fact that real-world graphs usually have varying patterns and complicated geometry, rather than being uniformly structured, while models with embedding spaces of the single geometric type usually fail in capturing the underlying mixed structures leading to higher distortion in representation (Gu et al., 2018).

Since existing models (Gu et al., 2018; Skopek et al., 2020) use product spaces with equally weighted components, training data samples are used to fit the learnable parameters in all these spaces evenly after the component spaces are determined. We argue that this approach limits the robustness of models when learning data of mixed geometric structures since all components in product spaces are updated equally, even if the input data have a dominating geometric type compared to others. As a result, it is difficult to appropriately adjust the geometry of product spaces to that of the input graph.

To address this problem, (Zhang et al., 2021) recently proposes a method called Switch Spaces which learns to choose a combination of K components from the total of N spaces with input specification. Switch Spaces starts with two embedding points, then learns to choose the top K spaces whose combination is best suited for a specific scoring function. However, with such a gating mechanism (Zhang et al., 2021) could be seen as a similarity learning between two graph nodes, thus Switch Spaces cannot capture the overall relation of points from input data.

To deal with this problem, we propose a new data-driven approach with a new scoring mechanism for each component in the product spaces that enables automatic learning of the soft weights for each component space based on the overall structure of the entire input data. More precisely, we introduce a new soft gating mechanism for automatic component spaces weighting and learning the weights of all component spaces from the input graph data. Our method allows us to construct general product spaces with weighted signatures based on given input data, which is capable of learning better representation from data with varying geometric structures.

Our main contributions are summarized in the following. *Firstly,* to our best knowledge, this work is the first one that discusses the problem of learning weighted product spaces from the entire input graph. *Secondly,* we propose a new approach to learn embedding product spaces for representation learning via a soft gating mechanism in Section 2. *Thirdly,* we perform experiments on synthetic and real-world datasets to validate our approach in Section 3.

2 Proposed Method

In this section, we describe our new approach for adaptively learning weighted product spaces for representation learning. Our goal is to construct product space models that make data with certain patterns be handled by suitable subsets of component spaces. To this end, we introduce a new soft weight gating mechanism. The main idea is to weigh the score for each component in embedding product spaces with specialization to enforce the curvature of the embedding to be approximate to the graph curvature.

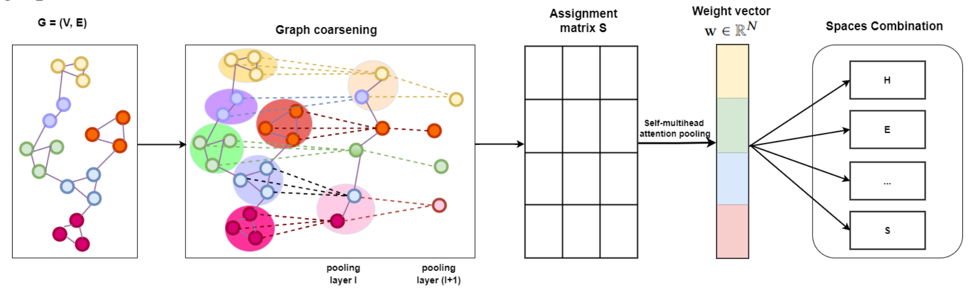

Figure 1: Illustration of weighting mechanism in our proposed method. From left to right, for an input graph $G = (V, E)$, we use a graph coarsening technique to pool G to produce assignment matrix $S_{n_l \times n_{l+1}}$, where $n_{l+1} = N$ is the number of components in the product space; self multi-head attention is then used to capture weight vector $\mathbf{w} \in \mathbb{R}^N$ which is then used to define soft weighted average distortion over all the nodes of G. We implement the coarsening module with DiffPool in this paper.

Problem formulation. Given three types of geometry Euclidean ($\mathbb{E}$), Hyperbolic ($\mathbb{H}$), and Spherical ($\mathbb{S}$). Let $M_1, M_2, \ldots, M_N$ be N component spaces where $M_i \in \{\mathbb{E}, \mathbb{H}, \mathbb{S}\}$ and $\dim M_i = b_i$. The goal is to learn from the input data the weighting scores $\mathbf{w} = (w_1, \ldots, w_N) \in \mathbb{R}^N$ for each component space such that the embedding of data into $\mathcal{P} = w_1 M_1 \times w_2 M_2 \times \cdots \times w_N M_N$ will have lowest geometric distortion.

Weighting scores and graph coarsening. To find the soft weight for each component space, we use the DiffPool architecture from (Ying et al., 2018) to distill topology information from the graph. In particular, let $S^{(l)} \in \mathbb{R}^{n_l \times n_{l+1}}$ denotes the cluster assigment matrix at pooling layer l. Then

Table 1: $D_{avg}(\downarrow)$ results on the synthetic datasets.

Dim	Method	Best model	**Cycle**	**Tree**	**Ring of trees**
d = 3	Single	$\mathbb{H}^3$	0.163	0.0461	0.0896
		$\mathbb{S}^3$	0.009	0.16	0.11
	Product	$\mathbb{H}^2 \times \mathbb{S}^1$	0.11	0.055	0.0632
	Ours	$w_1\mathbb{H}^2 \times w_2\mathbb{S}^1$	**0.104**	0.051	0.058
		$w_1\mathbb{H}^1 \times w_2\mathbb{S}^2$	0.116	**0.046**	**0.0514**
d = 5	Single	$\mathbb{H}^5$	0.132	0.042	0.09
		$\mathbb{S}^5$	0.011	0.14	0.13
	Product	$\mathbb{H}^2 \times \mathbb{S}^3$	0.115	0.07	0.052
		$\mathbb{H}^2 \times \mathbb{S}^2 \times \mathbb{E}$	0.12	0.092	0.047
	Ours	$w_1\mathbb{H}^2 \times w_2\mathbb{S}^3$	0.11	**0.058**	0.054
		$w_1\mathbb{H}^2 \times w_2\mathbb{S}^2 \times w_3\mathbb{E}$	**0.095**	0.086	**0.0414**

$S^{(l)}$ gives a soft assignment of each node at layer l to a cluster at the next layer $l+1$. We treat the number of cluster as a hyperparameter and $n_{l+1} = N$. Intuitively, each row of $S^{(l)}$ shows a soft margin of each node to each component space in $\mathcal{P}$. After the final pooling layer, we apply a multi-head attention block to get $\mathbf{w} \in \mathbb{R}^N$ which is the average weight of N spaces in $\mathcal{P}$. We refer to (Ying et al., 2018) for more detailed information on DiffPool.

Objective function and optimization. Let $\mathbf{w} \in \mathbb{R}^N$ is the weight vector of N spaces are captured from A.3. Our total average distortion objective function is defined as:

$$\mathcal{L}(x) = \sum_{1 \le i < j \le n} \left| \left(\frac{d_{\mathcal{P}}(x_i, x_j)}{d_G(X_i, X_j)} \right)^2 - 1 \right| + \lambda \mathcal{L}_{\text{aux}}, \tag{1}$$

where $d^2_{\mathcal{P}}(x_i, x_j) = \sum_{k=1}^{N} w_k \operatorname{dist}^2(x_i^k, x_j^k)$, d_G is the graph distance and $\mathcal{L}_{\text{aux}}$ is a combinations of the link prediction ($\mathcal{L}_{\text{LP}}$) loss and entropy regularization ($\mathcal{L}_e$). More precisely, $\mathcal{L}_{\text{LP}} = \left\| A^{(l)} - S^{(l)} S^{(l)^T} \right\|_F$ at each layer l, where $\| \cdot \|_F$ denotes the Frobenius norm, and $\mathcal{L}_e = \frac{1}{n} \sum_{i=1}^{n} H(S_i)$, where $H(S_i)$ is the entropy of the row i^{th} in matrix S. Minimizing $\mathcal{L}_e$ makes the output cluster assignment for each node close to a one-hot vector so that the membership for each cluster is clearly defined. Our objective function $\mathcal{L}(x)$ in (1) is optimized with the Riemannian gradient descent algorithm from (Gu et al., 2018, Algorithm 1).

3 Experiments

3.1 Embeddings of synthetic and real-world datasets.

Metrics and baselines. We use the average distortion D_{avg} and mean average precision mAP metrics whose formulas can be found in A.4 to measure the quality of graph embeddings. Pure product spaces (Gu et al., 2018) is used as the state-of-the-art baselines for comparison with our models.

Results on synthetic datasets. We first perform experiments of finding the best matching embedding spaces in total dimensions $d = 3$ and $d = 5$ for the three synthetic datasets Cycle, Tree, and Ring of trees introduced in Gu et al. (2018), each of them is a graph of 40 nodes with canonical geometry. The average distortion results are reported in Table 1. Overall, our best models significantly outperform best models from Gu et al. (2018) in terms of D_{avg}. For $d = 3$, our method improves upon the pure product model by 18.6% and 16.3% on Ring of trees and Tree datasets, respectively. When $d = 5$, our models perform better than the models of Gu et al. (2018) on Cycle and Ring of trees by 17.3% and 11.9%, respectively.

Results on real-world datasets. We evaluate the quality of embeddings of our models on two real-world datasets with embeddings into spaces of 10 and 50 total dimensions. The first dataset is *Cs PhDs*, a graph of computer science Ph.D advisor-advisee relationships (de Nooy et al., 2011). The second dataset is *Power*, a power distribution network (Watts & Strogatz, 1998).

Table 2: $D_{avg}(\downarrow)$ and $mAP(\uparrow)$ results of on the Cs PhDs and Power datasets.

Dim	Method	Best model	Cs PhDs		Power	
			D_{avg}	mAP	D_{avg}	mAP
d = 10	Single	$\mathbb{H}^{10}$	0.0502	0.9310	0.0388	0.8442
	Product	$(\mathbb{H}^2)^5$	0.0357	0.9694	0.0396	0.8739
		$\mathbb{H}^5 \times \mathbb{S}^5$	0.0529	0.9041	0.0323	0.8850
	Ours	$w(\mathbb{H}^2)^5$	**0.0301**	**0.9699**	0.0423	0.854
		$w(\mathbb{S}^2)^5$	0.055	0.8361	0.0402	**0.894**
		$w_1\mathbb{H}^5 \times w_2\mathbb{S}^5$	0.0494	0.9231	**0.0231**	0.8842
d = 50	Single	$\mathbb{H}^{50}$	0.091	0.881	0.0531	0.845
	Product	$(\mathbb{H}^{10})^5$	0.0657	0.913	0.0501	0.869
		$(\mathbb{H}^{10})^2 \times (\mathbb{S}^{10})^2 \times \mathbb{E}^{10}$	0.057	0.907	0.0495	0.856
	Ours	$w(\mathbb{H}^{10})^5$	0.062	**0.932**	0.0479	0.871
		$w(\mathbb{S}^5)^{10}$	0.062	0.87	0.0648	**0.871**
		$w_1(\mathbb{H}^{10})^3 \times w_2(\mathbb{S}^{10})^2$	0.059	0.9312	**0.0426**	0.862
		$w_1(\mathbb{H}^{10})^2 \times w_2(\mathbb{S}^{10})^2 \times w_3\mathbb{E}^{10}$	**0.046**	0.924	0.05	0.86

Results on the Cs PhDs and Power datasets are reported in Tables 2. On both dimensions and both datasets, our best models learn better embeddings than the product spaces models of (Gu et al., 2018). On the Cs PhDs dataset, our best models improve upon the best models of the pure product space approach (Gu et al., 2018) by 15.6% for $d = 10$, and 19.3% for $d = 50$ in terms of average distortion, respectively. Similarly, on the Power dataset, our models outperform best models of (Gu et al., 2018) by 28.4% and 13.9% on D_{avg} for $d = 10, 50$, respectively. We provide more experimental results on the quality of embeddings in the Appendix.

For the total embedding dimension $d = 10$, our method finds that the optimal weighted product spaces to embed the Power dataset is $0.83\,\mathbb{H}^5 \times 0.16\,\mathbb{S}^5$. The rate between hyperbolic and spherical components is $0.83 : 0.16 \approx 5 : 1$ which is only possible to capture by our approach.

An heuristics way to estimate the component in the product of model space. The component in the pool of the product spaces is chosen based on the distortion value of a single space on the synthetic and benchmark dataset. For example, Table 1 witness the best distortion of $\mathbb{S}^3$ on the Cycle dataset. Thus, we can choose the product of model spaces that has spherical properties as a component in the product of model spaces.

3.2 PERFORMANCE ON THE WORD SIMILARITY TASK

We evaluate our models on applications that require the understanding of the underlying manifold structure with a downstream task of learning word embeddings on the Word Similarity (WS-353) benchmark dataset as in previous works (Gu et al., 2018; Leimeister & Wilson, 2018). Our implementation is based on the hyperbolic skip-gram embeddings from (Leimeister & Wilson, 2018).

Setup. We use the standard skip-gram model (Mikolov et al., 2013a) and extend the loss function to a generic objective suitable for arbitrary manifolds, which is a variant of the objective used in (Leimeister & Wilson, 2018). More precisely, given a word u and a target w with label $y = 1$ if w is a context word for u, and $y = 0$ if it is a negative sample, the model is $P(y \mid w, u) = \sigma\left((-1)^{1-y}\left(-\cosh\left(d\left(\alpha_u, \gamma_w\right)\right) + \theta\right)\right)$.

Word similarity. We measure the Spearman rank correlation ρ between our scores and annotated ratings on the word similarity datasets WS-353 ((Finkelstein et al., 2001)) [1]. The results are reported in Table 3. In general, our models outperform both the hyperbolic word embeddings of (Leimeister & Wilson, 2018) and the pure product space models of (Gu et al., 2018) on both total dimension settings.

[1]https://github.com/mfaruqui/eval-word-vectors

Table 3: Spearman rank correlation on the WS-353 dataset.

Dim	Method	Best model	Spearman rank
d = 10	Single	$\mathbb{H}^{10}$	0.4412
	Product	$\mathbb{H}^5 \times \mathbb{H}^5$	0.4489
	Ours	$w_1\mathbb{H}^5 \times w_2\mathbb{H}^5$	**0.4510**
d = 50	Single	$\mathbb{H}^{50}$	0.6389
	Product	$(\mathbb{H}^{25})^2$	0.6421
		$(\mathbb{H}^{10})^5$	0.6531
	Ours	$w_1(\mathbb{H}^{25}) \times w_2\mathbb{H}^{25}$	0.6506
		$w(\mathbb{H}^{10})^5$	**0.6612**

4 CONCLUSION

Real-world data usually have complicated geometric structures which are difficult to capture by embedding into spaces of uniform curvature. We introduce a data-driven method of weighted product spaces for learning better representation. Our models improve both the embedding quality and downstream task performance compared to previous works.

REFERENCES

Ivana Balazevic, Carl Allen, and Timothy Hospedales. Multi-relational poincaré graph embeddings. *Advances in Neural Information Processing Systems*, 32:4463–4473, 2019.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems*, 32:4868–4879, 2019.

Ines Chami, Albert Gu, Vaggos Chatziafratis, and Christopher Ré. From trees to continuous embeddings and back: Hyperbolic hierarchical clustering. *arXiv preprint arXiv:2010.00402*, 2020a.

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. Low-dimensional hyperbolic knowledge graph embeddings. *arXiv preprint arXiv:2005.00545*, 2020b.

Wouter de Nooy, Andrej Mrvar, and Vladimir Batagelj. Exploratory social network analysis with pajek: Revised and expanded second edition. *Cambridge: Cambridge University Press. Dong, S. B.(1977). A Block–Stodola eigensolution technique for large algebraic systems with non-symmetrical matrices. International Journal for Numerical Methods in Engineering*, 11(2):247–267, 2011.

Francesco Di Giovanni, Giulia Luise, and Michael Bronstein. Heterogeneous manifolds for curvature-aware graph embedding. *arXiv preprint arXiv:2202.01185*, 2022.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pp. 406–414, 2001.

Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *arXiv preprint arXiv:1805.09112*, 2018.

Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. Learning mixed-curvature representations in product spaces. In *International Conference on Learning Representations*, 2018.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Matthias Leimeister and Benjamin J Wilson. Skip-gram word embeddings in hyperbolic space. *arXiv preprint arXiv:1809.01498*, 2018.

Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. *arXiv preprint arXiv:1910.12892*, 2019.

Yu Meng, Jiaxin Huang, Guangyuan Wang, Chao Zhang, Honglei Zhuang, Lance Kaplan, and Jiawei Han. Spherical text embedding. *Advances in Neural Information Processing Systems*, 32:8208–8217, 2019.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun (eds.), *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013a.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013b.

Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30:6338–6347, 2017.

Peter Petersen. *Riemannian geometry*, volume 171. Springer, 2006.

Ondrej Skopek, Octavian-Eugen Ganea, and Gary Bécigneul. Mixed-curvature variational autoencoders. In *International Conference on Learning Representations*, 2020.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 583–591, 2018.

Lucas Vinh Tran, Yi Tay, Shuai Zhang, Gao Cong, and Xiaoli Li. Hyperml: A boosting metric learning approach in hyperbolic space for recommender systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 609–617, 2020.

Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393 (6684):440–442, 1998.

Thomas James Willmore. *An introduction to differential geometry*. Courier Corporation, 2013.

Richard C Wilson, Edwin R Hancock, Elżbieta Pekalska, and Robert PW Duin. Spherical and hyperbolic embeddings of data. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2255–2269, 2014.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

Shuai Zhang, Yi Tay, Wenqi Jiang, Da-cheng Juan, and Ce Zhang. Switch spaces: Learning product spaces with sparse gating. *arXiv preprint arXiv:2102.08688*, 2021.

Table 4: Notations.

Notation	Explanations
f	an embedding
$D(f)$	the distortion fidelity measure of the embedding f
$mAP(f)$	the mean average precision fidelity measure of the embedding f
G	a graph with V nodes and E edges
n	is the number of nodes in the graph G
a, b	nodes in a graph
$\mathcal{N}_a$	neighborhood around node a in a graph
$\mathcal{R}_{a,b}$	the smallest set of closest points to node a in an embedding f that contains node b
M	is an Riemanninan manifold when equipped with a metric g
p	$p \in M$ is a point in a manifold M
T_pM	the tangent spaces of point p in M
g	is an metric defining an inner product on T_pM
$\mathbb{E}^d$	d-dimensional Euclidean space
$\mathbb{S}^d$	d-dimensional Spherical space
$\mathbb{H}^d$	d-dimensional Hyperbolic space
$\mathcal{P}$	product manifold including spherical, euclidean and hyperbolic components
$\mathrm{Exp}_x(v)$	the exponential map at point x of tangent vector v
d_E	metric distance between two points in Euclidean space
d_H	metric distance between two points in Hyperbolic space
d_S	metric distance between two points in Spherical space
d_G	metric distance between two points in a graph G
$K(x, y)$	the sectional curvature for a subspace spanned by linearly independent $x, y \in T_pM$
J	is the diagonal matrix which is used in the Minkowski inner product
$\mathrm{acosh}()$	is the inverse hyperbolic cosine function
$\arccos()$	is the inverse cosine function
$A \in \mathbb{R}^{n \times n}$	is a graph adjacency matrix of graph G
X	is initial embedding for all the node in graph G
GNN	a graph neural network model with inputs are A, X
Z	the embedding matrix which is achieved from a GNN model
Z'	the coarsened embedding matrix which is achieved from a GNN model
$\|\|p\|\|_2$	l_2 norm of vector p
$S^{(l)}_{n_l \times n_{l+1}}$	is a soft assignment matrix at layer l
H	is the entropy function
$\deg(a)$	is the degree of node a in undirected graph G
$\mathbf{w} = (w_1, \ldots, w_N) \in \mathbb{R}^N$	is the weighted score to each of the given components in the product of model space.

A Appendix

A.1 Related work

In recent years, learning representation in non-Euclidean spaces has gained increasing interest. For example, hyperbolic representation learning has shown state of the art performance on a number of tasks such as taxonomic entities modeling (Nickel & Kiela, 2017), network embeddings (Nickel & Kiela, 2017), recommender systems (Vinh Tran et al., 2020), question answering (Tay et al., 2018), knowledge graphs completion (Balazevic et al., 2019; Chami et al., 2020b), graph-related classification (Chami et al., 2019; Liu et al., 2019), similarity-based hierarchical Clustering (Chami et al., 2020a). Hyperbolic space is better for the embedding of trees and excels in modeling hierarchical structures (Chami et al., 2020a), while spherical space is a more suitable choice for directional similarity modeling and data with cyclical structure (Meng et al., 2019; Wilson et al., 2014). Recent works have applied spherical spaces in text embeddings (Meng et al., 2019), texture mapping (Wilson et al., 2014), time-warping functions embedding (Wilson et al., 2014). As discussed above, different spaces have their own special geometric feature, and the choice of embedding spaces depends on the characteristics of the data.

The quality of such feature embeddings crucially depends on whether the geometry of the space matches that of the graph. Euclidean spaces are often a poor choice for many types of real-world graph data, where one encounters hierarchical structure and power-law degree distribution due to negative curvature are linked to negative curvature (Chami et al., 2020a). In particular, recent works (Chami et al., 2020b; Gu et al., 2018; Zhang et al., 2021) show that hyperbolic spaces and more general manifolds, such as products of constant-curvature spaces and matrix manifolds, are advantageous to approximate better capture from graphs. A product space is a direct product of embedding spaces with heterogeneous curvature (e.g., Euclidean space, hyperbolic space, spherical space, etc.) (Zhang et al., 2021; Gu et al., 2018). Each component of product space has a constant curvature while the curvature of product space is the sum of curvatures of components which is still constant (Di Giovanni et al., 2022), which makes it possible to capture a wider range of curvatures with lower distortion than a single space. Theoretically, product space is well-suited for modeling real-world data with a mixture of structures. Empirically, product spaces have demonstrated their efficacy in graph reconstruction (Gu et al., 2018), wording embedding with low dimensions (Gu et al., 2018).

A.2 Background

Embeddings. For metric spaces U, V equipped with distances d_U, d_V, an embedding is an injective continuous map $f : U \to V$. The quality of an embedding is measured by various fidelity measures. A standard measure is the *average distortion* $D_{\text{avg}}(f)$. The distortion of a pair of points a, b is $\left|\left(\frac{d_V(f(a),f(b))}{d_U(a,b)}\right)^2 - 1\right|$, and $D_{\text{avg}}(f)$ is the average distortion over all pairs of points.

Distortion is a global metric; it considers the explicit value of all distances. At the other end of the global-local spectrum of fidelity measures are the *mean average precision* (mAP), which applies to unweighted graphs. Let $G = (V, E)$ be a graph and node $a \in V$ have neighborhood $\mathcal{N}_a = \{b_1, \cdots, b_{\text{deg(a)}}\}$, where $\deg(a)$ is the degree of a. In the embedding f, define R_{a,b_i} to be the smallest ball around $f(a)$ that contains b_i which means R_{a,b_i} is the smallest set of nearest points required to retrieve the i^{th} neighbor of a in f. Thus, define $\text{mAP}(f) = \frac{1}{|V|}\sum\limits_{a\in V}\frac{1}{\deg(a)}\sum\limits_{i=1}^{|\mathcal{N}_a|}\frac{|\mathcal{N}_a \cap R_{a,b}|}{|R_{a,b_i}|}$. $mAP(f)$ is a ranking-based measure for local neighborhoods; it does not track explicit distances like a distortion measure. We see that, $mAP(f) \leq 1$ (higher is better) while $D_{avg} \geq 0$ (lower is better) (Gu et al., 2018).

Riemannian manifolds. Let M be a smooth manifold, $p \in M$ be a point, and T_pM be the tangent space to the point p. If M is equipped with a Riemannian metric g, then the pair (M, g) is called a Riemannian manifold. The shortest-distance paths on manifolds are called geodesics. To compute distance functions on a Riemannian manifold, the metric tensor g is integrated along the geodesic. This is a smoothly varying function (in p) $g_p(.,.) : T_pM \times T_pM \to \mathbb{R}$ that induces geometric notions such as length and angle by defining an inner product on the tangent space. For example, the norm of $v \in T_pM$ is defined as $||v||_g := g_p(v, v)^{\frac{1}{2}}$. In Euclidean space $\mathbb{R}^d$, each tangent space $T_p\mathbb{R}^d$ is canonically identified with $\mathbb{R}^d$, and the metric tensor g^E is simply the usual inner product. We refer to (Willmore, 2013; Petersen, 2006) for a formal introduction of Riemannian geometry.

Product manifolds. Consider a sequence of smooth manifolds $M_1, M_2, \cdots, M_k$. The product manifold is defined as the Cartesian product $M = M_1 \times M_2 \times \cdots \times M_k$. We write points $p \in M$ through their coordinates $p = (p_1, \ldots, p_k), p_i \in M_i$ and similarly a tangent vector $v \in T_pM$ can be written $(v_1, \ldots, v_k) : v_i \in T_{p_i}M_i$. If the M_i are equipped with metric tensor g_i, then the product M is also Riemannian with metric tensor $g_p(u, v) = \sum\limits_{i=1}^{k} g_{i p_i}(u_i, v_i)$.

Geodesics and distances. Gradient descent algorithms on manifolds requires a notion of taking a step. This step can be performed in the tangent space and transferred to the manifold via the exponential map $\mathrm{Exp_p} : T_pM \to M$ (Willmore, 2013). In a product manifold P, for tangent vectors $v = (v_1, \ldots, v_k)$ at $p = (p_1, \ldots, p_k) \in P$, the exponential map simply decomposes, as do squared distances:

$$\operatorname{Exp}_p(v) = \left(\operatorname{Exp}_{p_1}(v_1), \ldots, \operatorname{Exp}_{p_k}(v_k)\right), \quad d_{\mathcal{P}}^2(x, y) = \sum_{i=1}^{k} d_i^2(x_i, y_i) \tag{2}$$

In other words, the shortest path between points in the product travels along the shortest paths in each component simultaneously. Note the analogy to Euclidean products $\mathbb{R}^d \equiv \left(\mathbb{R}^1\right)^d$.

Hyperbolic and spherical models. We use the hyperboloid model of hyperbolic space with points in $\mathbb{R}^{d+1}$. Let $J \in \mathbb{R}^{(d+1)\times(d+1)}$ be the diagonal matrix with $J_{00} = -1$ and $J_{ii} = 1 : i > 0$. For

$p, q \in \mathbb{R}^{d+1}$, the Minkowski inner product is $\langle p, q\rangle_* := p^T Jq = -p_0q_0 + p_1q_1 + \ldots + p_dq_d$, and the corresponding norm is $||p||_* = \langle p, p\rangle_*^{\frac{1}{2}}$. For any $K > 0$, the hyperboloid $\mathbb{H}_K^d$ is defined as the subset $\{p \in \mathbb{R}^{d+1} : \|p\|_* = -K^{1/2}, p_0 > 0\}$. When the subscript K is omitted, it is taken to be 1. The hyperbolic distance on $\mathbb{H}^d$ is $d_H(p, q) = acosh(-\langle p, q\rangle_*)$ where $acosh()$ is the inverse hyperbolic cosine function (Willmore, 2013; Petersen, 2006).

Similarly, spherical space $\mathbb{S}_K^d$ is most easily defined when embedded in $\mathbb{R}^{d+1}$. The manifold $\mathbb{S}_K^d$ is defined as the subset $p \in \mathbb{R}^{d+1} : ||p||_2 = K^{\frac{1}{2}}$, with metric tensor g^S is induced by inner product on the tangent space at each point only. The spherical distance on $\mathbb{S}^d$ is $d_S\langle p, q\rangle = arccos(\langle p, q\rangle)$ where $arccos()$ is the inverse cosine function (Willmore, 2013; Petersen, 2006).

A.3 WEIGHTED PRODUCT MANIFOLDS AND GRAPH COARSENING.

We aim to obtain the weight for each model which reflects how much the component space contributes to the product of the model spaces. For example, in the case of the Ring of tree dataset which has a lot of tree structures and one ring, we can think that it will be better if we evaluate the score for the hyperbolic model is bigger than the spherical model. In this work, to get the soft weight information for all the spaces, we use the DiffPool (Ying et al., 2018) architecture to distill topology information from the graph. Formally, let $Z = GNN(A, X) \in \mathbb{R}^{n\times d}$ be the output of a GNN module, where $A \in \mathbb{R}^{n\times n}$ is a graph adjacency matrix. DiffPool defines a strategy to output a new coarsened graph with $m < n$ nodes and a new weighted adjacency matrix $A' \in \mathbb{R}^{m\times m}$ and node embeddings $Z' \in \mathbb{R}^{m\times d}$. DiffPool has the interactive property which means that the new coarsen graph can turn to be used as input to another DiffPool layer, and this whole process can be repeated L times, generating a model with L GNN layers that operate on a series of coarser and coarser versions of the input graph.

For specific, $S^{(l)} \in \mathbb{R}^{n_l\times n_{l+1}}$ denotes the cluster assigment matrix at layer l. $S^{(l)}$ gives a soft assignment of each node at layer l to a cluster at the next layer $l + 1$. In other words, each row of $S^{(l)}$ corresponds to one of the n_l nodes (or clusters) at layer l, and each column of $S^{(l)}$ corresponds to one of the n_{l+1} clusters at the next layer $l + 1$. In our work, we treat the number of cluster as a hyperparameter and $n_{l+1} = N$, where N is the number of components in the product space P. Intuitively, each row of $S^{(l)}$ shows a soft margin of each node to each each component space in P. At the end, we use a multihead attention pooling to achieve $\mathbf{w} \in \mathbb{R}^N$ which is presented for the average weight of N spaces in $\mathcal{P}$ over the n_l clusters. We refer to (Ying et al., 2018) for more information.

A.4 METRIC

We use two common metrics average distortion D_{avg}, and mean average precision mAP to measure the quality of embeddings.

Distortion measure. Distortion measure is used to quantify how far an embedding map is from an isometry. Assume we have a set data points $P = \{p_1, \ldots, p_N\} \subset \mathcal{X}$. An embedding map $f : \mathcal{X} \to Y$ is called D-embedding, where $D > 1$, and $d_{\mathcal{X}}(.,.)$ is the distance in the input space (Euclidean or graph distance) and $d_Y(.,.)$ is the distance in the embedding spaces. The total distortion is defined as:

$$D_{\text{avg}}(f) := \sum_{1\leq i\leq j\leq N} \left| \left(\frac{d_Y\left(f\left(p_i\right), f\left(p_j\right)\right)}{d_{\mathcal{X}}\left(p_i, p_j\right)} \right)^2 - 1 \right| \tag{3}$$

Mean average precision. Mean average precision (mAP) is used as a local measure of fidelity for embeddings of unweighted graphs. Let $G = (V, E)$ be a graph and node $a \in V$ have neighborhood $N_a = \{b_1, \ldots, b_{deg(a)}\}$ where $deg(a)$ is the degree of a. In the embedding f, define R_{a,b_i} to be smallest ball around $f(a)$ that contains b_i (that is, R_{a,b_i} is the smallest set of nearest points required to retrieve the i-th neighbor of a in f). The formula of mAP is:

$$\text{mAP}(f) = \frac{1}{|V|} \sum_{a\in V} \frac{1}{\deg(a)} \sum_{i=1}^{|N_a|} \frac{|N_a \cap R_{a,b_i}|}{|R_{a,b_i}|}. \tag{4}$$

Table 5: Distortion measure on synthetic datasets (lower is better). Best results from Product space method (Gu et al., 2018) are underlined, while our best results are boldfaced. w_i's are learnable weights depended on input data.

Dim	Method	Best model	**Cycle**	**Tree**	**Ring of trees**
d = 3	Product				
		$\mathbb{H}^2 \times \mathbb{S}^1$	0.11	0.055	<u>0.0632</u>
	Ours				
		$w_1\mathbb{H}^2 \times w_2\mathbb{S}^1$	$\mathbf{0.104}$	0.051	0.058
		$w_1\mathbb{H}^1 \times w_2\mathbb{S}^2$	0.116	$\mathbf{0.046}$	$\mathbf{0.0514}$
d = 5	Product				
		$\mathbb{H}^2 \times \mathbb{S}^3$	0.115	0.07	0.052
		$\mathbb{H}^2 \times \mathbb{S}^2 \times \mathbb{E}$	0.12	0.092	<u>0.047</u>
	Ours				
		$w_1\mathbb{H}^2 \times w_2\mathbb{S}^3$	0.11	$\mathbf{0.058}$	0.054
		$w_1\mathbb{H}^2 \times w_2\mathbb{S}^2 \times w_3\mathbb{E}$	$\mathbf{0.095}$	0.086	$\mathbf{0.0414}$
d = 10	Product				
		$(\mathbb{S}^5)^2$	0.194	0.254	0.0958
		$\mathbb{H}^5 \times \mathbb{S}^5$	0.2176	0.211	<u>0.0885</u>
	Ours				
		$w(\mathbb{S}^5)^2$	$\mathbf{0.1602}$	0.231	0.089
		$w_1\mathbb{H}^5 \times w_2\mathbb{S}^5$	0.1691	$\mathbf{0.1985}$	$\mathbf{0.0823}$

Table 6: Distortion measure of single space on synthetic datasets (lower is better).

Dim	Model	**Cycle**	**Tree**	**Ring of trees**
		$\|V\|$ = 40	$\|V\|$ = 40	$\|V\| = 40$
		$\|E\| = 40$	$\|E\| = 39$	$\|E\| = 40$
d = 3				
	$\mathbb{H}^3$	0.163	$\mathbf{0.0461}$	$\mathbf{0.0896}$
	$\mathbb{E}^3$	0.106	0.149	0.0989
	$\mathbb{S}^3$	$\mathbf{0.009}$	0.16	0.11
d = 5				
	$\mathbb{H}^5$	0.132	$\mathbf{0.042}$	$\mathbf{0.09}$
	$\mathbb{E}^5$	0.095	0.186	0.093
	$\mathbb{S}^5$	$\mathbf{0.011}$	0.14	0.13
d = 10				
	$\mathbb{H}^{10}$	0.275	$\mathbf{0.098}$	0.1104
	$\mathbb{E}^{10}$	0.236	0.1968	$\mathbf{0.103}$
	$\mathbb{S}^{10}$	$\mathbf{0.114}$	0.1769	0.1463

$mAP(f)$ is a ranking-based measure for local neighborhoods; it does not track explicit distances like a distortion measure.

A.5 Parameter settings

We choose the best parameters for each model. Several hyper-parameters having a major impact on the models are lr (learning rate); λ (to balance the influence of auxiliary loss). We run 500 epochs and 5000 epochs with the synthetic and benchmark datasets respectively. We turn hyper-parameters with lr in $\{0.001, 0.003, 0.005, 0.01\}$ for Cities and three synthetic graphs. lr is searched in the range $\{1, 3, 5, 10\}$ with the remaining dataset. λ is searched in range $\{0.5, 1, 1.5, 2, 3\}$. The tuning process gives us $lr = 0.005$ for Cities graph, $lr = 0.01$ for the three synthetic graphs, and $lr = 5$ for the two remaining benchmark datasets. We choose $\lambda = 1.5$ for the best distortion. In the weighted module, we use DiffPool with two GraphSAGE (Hamilton et al., 2017) pooling layers.

A.6 Additional experimental results

Table 7: Distortion measure on synthetic datasets (lower is better). w_i are learnable weights depended on input data.

Dim	Method	Models	Cycle	Tree	Ring of trees
d = 3	Product				
		$\mathbb{H}^2 \times \mathbb{S}^1$	0.11	0.055	$\mathbf{0.0632}$
	Ours				
		$w_1\mathbb{H}^2 \times w_2\mathbb{S}^1$	$\mathbf{0.104}$ (+5.4%)	0.051	0.058
		$w_1\mathbb{H}^1 \times w_2\mathbb{S}^2$	0.116	$\mathbf{0.046}$ (+16.3%)	$\mathbf{0.0514}$ (+18.6%)
d = 5	Product				
		$\mathbb{H}^3 \times \mathbb{S}^2$	0.124	$\mathbf{0.06}$	0.054
		$\mathbb{H}^2 \times \mathbb{S}^3$	$\mathbf{0.115}$	0.07	0.052
		$\mathbb{H}^2 \times \mathbb{S}^2 \times \mathbb{E}$	0.12	0.092	$\mathbf{0.047}$
	Ours				
		$w_1\mathbb{H}^3 \times w_2\mathbb{S}^2$	0.102	0.068	0.05
		$w_1\mathbb{H}^2 \times w_2\mathbb{S}^3$	0.11	$\mathbf{0.058}$ (+3.3%)	0.054
		$w_1\mathbb{H}^2 \times w_2\mathbb{S}^2 \times w_3\mathbb{E}$	$\mathbf{0.095}$ (+17.3%)	0.086	$\mathbf{0.0414}$ (+11.9%)
d = 10	Product				
		$(\mathbb{H}^5)^2$	0.221	0.223	0.0932
		$(\mathbb{S}^5)^2$	$\mathbf{0.194}$	0.254	0.0958
		$\mathbb{H}^5 \times \mathbb{S}^5$	0.2176	$\mathbf{0.211}$	$\mathbf{0.0885}$
	Ours				
		$(\mathbf{w}\mathbb{S}^5)^2$	$\mathbf{0.1602}$ (+17.4%)	0.231	0.089
		$w_1\mathbb{H}^5 \times w_2\mathbb{S}^5$	0.1691	$\mathbf{0.1985}$ (+5.9%)	$\mathbf{0.0823}$ (+7%)

Table 8: Statistics of all three benchmark datasets.

Dataset	Number of nodes ($\|V\|$)	Number of edges ($\|E\|$)
Cs PhDs	1025	1043
Power	4941	6594
Cities	312	48 516

Table 9: Distortion and mAP of models with single embedding space on benchmark datasets (D_{avg}: lower is better; mAP: higher is better).

Dim	Model	Cs PhDs		Power		Cities	
		D_{avg}	mAP	D_{avg}	mAP	D_{avg}	mAP
d = 10							
	$\mathbb{H}^{10}$	$\mathbf{0.0502}$	0.9310	$\mathbf{0.0388}$	0.8442	0.0938	0.814
	$\mathbb{E}^{10}$	0.0543	0.8691	0.0917	0.8860	0.0753	0.843
	$\mathbb{S}^{10}$	0.0569	0.8329	0.0500	0.7952	$\mathbf{0.0613}$	0.8513
d = 50							
	$\mathbb{H}^{50}$	$\mathbf{0.091}$	0.881	$\mathbf{0.0531}$	0.845	0.0732	0.832
	$\mathbb{E}^{50}$	0.097	0.842	0.0752	0.821	$\mathbf{0.062}$	0.853
	$\mathbb{S}^{50}$	0.112	0.812	0.0623	0.823	0.0623	0.879
d = 100							
	$\mathbb{H}^{100}$	0.16	0.82	$\mathbf{0.091}$	0.792	0.124	0.798
	$\mathbb{E}^{100}$	$\mathbf{0.057}$	0.889	0.112	0.784	0.112	0.802
	$\mathbb{S}^{100}$	0.125	0.853	0.095	0.812	$\mathbf{0.098}$	0.8103

Table 10: Distortion and mAP on benchmark datasets (D_{avg}: lower is better; mAP: higher is better).

Dim	Methods	Models	Cs PhDs		Power		Cities	
			D_{avg}	mAP	D_{avg}	mAP	D_{avg}	mAP
d = 10	Product							
		$(\mathbb{H}^2)^5$	$\mathbf{0.0357}$	0.9694	0.0396	0.8739	0.0687	0.842
		$(\mathbb{H}^5)^2$	0.0382	0.9628	0.0365	0.8605	0.0765	0.824
		$(\mathbb{S}^5)^2$	0.0579	0.7940	0.0471	0.8059	$\mathbf{0.0593}$	0.8387
		$(\mathbb{S}^2)^5$	0.0562	0.8314	0.0483	0.8818	0.0662	0.8301
		$\mathbb{H}^5 \times \mathbb{S}^5$	0.0529	0.9041	$\mathbf{0.0323}$	0.8850	0.0642	0.852
	Ours							
		$\mathbf{w}(\mathbb{H}^2)^5$	$\mathbf{0.0301}$ (+15.6%)	0.9699	0.0423	0.854	0.0693	0.8315
		$\mathbf{w}(\mathbb{H}^5)^2$	0.0489	0.8465	0.034	0.88	0.076	0.83
		$\mathbf{w}(\mathbb{S}^5)^2$	0.056	0.813	0.0431	0.812	0.06	0.832
		$\mathbf{w}(\mathbb{S}^2)^5$	0.055	0.8361	0.0402	0.894	0.0632	0.82
		$w_1\mathbb{H}^5 \times w_2\mathbb{S}^5$	0.0494	0.9231	$\mathbf{0.0231}$ (+28.4%)	0.8842	$\mathbf{0.0573}$ (+3%)	0.836
d = 50	Product							
		$(\mathbb{H}^{10})^5$	0.0657	0.913	0.0501	0.869	0.071	0.86
		$(\mathbb{H}^5)^{10}$	0.0786	0.86	0.0723	0.842	0.083	0.845
		$(\mathbb{S}^{10})^5$	0.0726	0.843	0.0682	0.84	0.0458	0.8932
		$(\mathbb{S}^5)^{10}$	0.07	0.89	0.0701	0.831	0.0523	0.879
		$(\mathbb{H}^{10})^3 \times (\mathbb{S}^{10})^2$	0.062	0.912	0.051	0.85	0.0512	0.8821
		$(\mathbb{H}^{10})^2 \times (\mathbb{S}^{10})^2 \times \mathbb{E}^{10}$	$\mathbf{0.057}$	0.907	$\mathbf{0.0495}$	0.856	$\mathbf{0.0449}$	0.912
	Ours							
		$\mathbf{w}(\mathbb{H}^{10})^5$	0.062	0.932	0.0479	0.871	0.082	0.8497
		$\mathbf{w}(\mathbb{H}^5)^{10}$	0.059	0.89	0.0503	0.857	0.08	0.851
		$\mathbf{w}(\mathbb{S}^{10})^5$	0.074	0.829	0.0613	0.86	0.0423	0.912
		$\mathbf{w}(\mathbb{S}^5)^{10}$	0.062	0.87	0.0648	0.871	0.051	0.881
		$\mathbf{w_1}(\mathbb{H}^{10})^3 \times \mathbf{w_2}(\mathbb{S}^{10})^2$	0.059	0.9312	$\mathbf{0.0426}$ (+13.9%)	0.862	0.0487	0.903
		$\mathbf{w_1}(\mathbb{H}^{10})^2 \times \mathbf{w_2}(\mathbb{S}^{10})^2 \times w_3\mathbb{E}^{10}$	$\mathbf{0.046}$ (+19.3%)	0.924	0.05	0.86	$\mathbf{0.0397}$ (+11.5%)	0.9258
d=100	Product							
		$(\mathbb{H}^{20})^5$	0.0485	0.86	0.078	0.83	0.097	0.821
		$(\mathbb{S}^{20})^5$	0.0713	0.921	0.08	0.812	0.094	0.8114
		$(\mathbb{H}^{50}) \times (\mathbb{S}^{50})$	0.0623	0.912	0.0712	0.796	0.098	0.81
		$(\mathbb{H}^{20})^2 \times (\mathbb{S}^{20})^2 \times \mathbb{E}^{20}$	$\mathbf{0.04774}$	0.934	$\mathbf{0.0632}$	0.8423	$\mathbf{0.0912}$	0.803
	Ours							
		$\mathbf{w}(\mathbb{H}^{20})^5$	0.044	0.89	0.0691	0.8432	0.103	0.802
		$\mathbf{w}(\mathbb{S}^{20})^5$	0.072	0.88	0.062	0.8	0.0793	0.84
		$w_1(\mathbb{H}^{50}) \times w_2(\mathbb{S}^{50})$	0.0532	0.9342	0.07	0.823	$\mathbf{0.0769}$ (+15.6%)	0.847
		$\mathbf{w_1}(\mathbb{H}^{20})^2 \times \mathbf{w_2}(\mathbb{S}^{20})^2 \times w_3\mathbb{E}^{20}$	$\mathbf{0.0321}$ (+32.5%)	0.943	$\mathbf{0.0487}$ (+14.3%)	0.862	0.092	0.798