L^{2}GC: Lorentzian Linear Graph Convolutional Networks For Node Classification

Qiuyu Liang[1], **Weihua Wang**[1,2,3 *], **Feilong Bao**[1,2,3], **Guanglai Gao**[1,2,3]
[1] College of Computer Science, Inner Mongolia University, Hohhot, China
[2] National and Local Joint Engineering Research Center of Intelligent Information Processing Technology for Mongolian, Hohhot, China
[3] Inner Mongolia Key Laboratory of Mongolian Information Processing Technology, Hohhot, China
liangqiuyu@mail.imu.edu.cn, wangwh@imu.edu.cn

Abstract

Linear Graph Convolutional Networks (GCNs) are used to classify the node in the graph data. However, we note that most existing linear GCN models perform neural network operations in Euclidean space, which do not explicitly capture the tree-like hierarchical structure exhibited in real-world datasets that modeled as graphs. In this paper, we attempt to introduce hyperbolic space into linear GCN and propose a novel framework for Lorentzian linear GCN. Specifically, we map the learned features of graph nodes into hyperbolic space, and then perform a Lorentzian linear feature transformation to capture the underlying tree-like structure of data. Experimental results on standard citation networks datasets with semi-supervised learning show that our approach yields new state-of-the-art results of accuracy 74.7% on Citeseer and 81.3% on PubMed datasets. Furthermore, we observe that our approach can be trained up to two orders of magnitude faster than other nonlinear GCN models on PubMed dataset. Our code is publicly available at https://github.com/llqy123/LLGC-master.

Keywords: Linear Graph Convolutional Networks, Hyperbolic Space, Graph Node Classification

1. Introduction

We consider the problem of classifying nodes (such as documents) on the graph-structured data (such as citation network), where models need to learn information from their neighbours. Graph Convolutional Network (GCN) is one of classification models that be able to aggregate and propagate the node information in graphs. Therefore, GCNs and their variants have achieved success in various tasks, such as text classification (Yao et al., 2019), relation extraction (Tian et al., 2021), recommendation systems (Ying et al., 2018; Shang et al., 2019; Wang et al., 2021a; Feng et al., 2022) and social networks analysis (Li and Goldwasser, 2019; Sankar et al., 2021).

In order to learn the features of each node, GCNs typically consist of two successive stages: node feature propagation and transformation. In the first stage, GCNs stacked multiple layers by a nonlinear activation function to aggregate and utilize nodes information from neighboring nodes. This multi-layer structure can obtain both local and global features naturally, yet the computation complexity increased rapidly and further hinder their applications. Therefore, some researchers attempted to improve the GCNs by eliminating the nonlinear active function between layers, such as Wu et al. (2019); Wang et al. (2021b); Li et al. (2022), which are referred to as linear GCN model. However, these traditional linear GCN models mainly focus on improving the nodes feature propagation scheme, while pay fewer attention to the feature transformation stage.

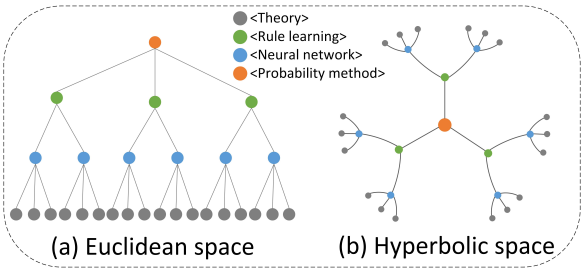

Figure 1: Visualization of a tree-like hierarchical structure citation network in Euclidean and hyperbolic space. In both graphs, nodes represent documents, and edges represent citations. Each color node represents a type of document.

In the second stage, GCNs will transform the learned features and structural information to further classify nodes. The feature transformation is usually performed in Euclidean space. In fact, real-world datasets typically exhibit tree-like hierarchical or scale-free structures, but feature transformation in Euclidean space may be distorted when faced with this kind of data. Unlike Euclidean spaces, hyperbolic spaces have a greater capacity to learn scale-free or hierarchical structures graph, which can be used to learn tree-like graphs (Clauset et al., 2008; Muscoloni et al., 2017). For example, we il-

*Corresponding author.

LREC-COLING 2024, pages 9988–9998

lustrated the same tree-like graph instance within Euclidean and hyperbolic spaces in Figure 1. From Figure 1, we obvious that the distance between two nodes will be larger when mapping to the hyperbolic space. Since the distance between two nodes is the Euclidean distance, while it will change to a geodesic distance in hyperbolic space. Furthermore, the local dependencies and hierarchical relationships between nodes can be preserved simultaneously when mapping into hyperbolic space.

To address the limitation of feature transformation in Euclidean space, we propose a novel **L**orentzian **L**inear **G**raph **C**onvolutional Networks (**L**^{2}**GC**) for classifying the tree-like graph node. In contrast to its Euclidean space counterparts, we modeled the node feature transformation with Lorentzian manifold. The main contributions of this study can be summarized as follows:

- To the best of our knowledge, our work is the first to present a framework for modeling the feature transformation of a linear GCN model in hyperbolic space with Lorentzian manifold.
- Our approach can be trained two orders of magnitude speedup over nonlinear models on the large-scale dataset (Pubmed) in our evaluation while be computationally more efficient and fitting significantly fewer parameters.
- Extensive experiments on semi-supervised node classification tasks show that our model achieves state-of-the-art results on tree-like datasets compared with other nonlinear or linear GCN models in different spaces.

2. Related Work

2.1. Linear Graph Convolutional Networks

Traditional GCNs are commonly used to process graph data since their abilities to aggregate and propagate information among nodes, but these models often burden excess complexity that inherit from their neural networks lineage. Recent efforts try to improve linear GCNs on the processing of node information propagation. For example, Wu et al. (2019) simplified GCNs by repeatedly removing the nonlinearities between GCN layers and folding weight matrices between layers to form the new objective function into a single linear transformation. Wang et al. (2021b) introduced Decoupled Graph Convolution which decouples the terminal time and the feature propagation steps. Li et al. (2022) introduced a novel propagation layer from a spectrum perspective. This new layer decoupled the three concentration properties: concentration center, maximum response and bandwidth, which implement a low-pass graph filter for a linear graph model. In addition, fully linear GCNs are proposed in Cai et al. (2023). This is a fully linear GCN that both straightforward and efficient for semi-supervised and unsupervised learning tasks.

While above linear GCNs have achieved competitive performance, they only focus on the graph node feature propagation stage and overlook the feature transformation stage. They perform feature transformation operations in Euclidean space, which leads to greater distortion when faced with real-world graphs with scale-free or hierarchical structures. Different from the above work, our work takes the foundation of linear GCNs and transforms the feature within hyperbolic spaces.

2.2. Hyperbolic Graph Convolutional Networks

Recently, a growing number of researchers have investigated hyperbolic geometry since their greater capacity to learn graphs that characterized by scale-free or hierarchical structures. For example, Chami et al. (2019) lifted graph convolutional network to hyperbolic geometry and then proposed hyperbolic Graph Convolutional Network (HGCN) model. Zhang et al. (2021a) studied the Graph Neural Network with attention mechanism in hyperbolic spaces at the first attempt. To address the issue of the HGCN model's operation on the tangent space that dissatisfied definition of hyperbolic geometry, Zhang et al. (2021b) introduced the Lorentzian version of traditional GCN and reconstructed the graph operations of hyperbolic GCN. Chen et al. (2022) devised a completely hyperbolic framework without tangent spaces to address the limitation of hyperbolic GCN models.

Although existing hyperbolic GCN models could address the challenge of scale-free or hierarchical structures when handling graph, these models often suffer from complex structures and large amount of parameters. In order to reduce the time cost and number of parameters, we try to simplify the calculation process in hyperbolic space.

3. Preliminaries

In this section, we briefly review the key concepts of hyperbolic geometry. A thorough and detailed explanation of hyperbolic geometry can be found in (Willmore, 2013). Hyperbolic geometry is a non-Euclidean geometry with constant negative curvature k $(k > 0)$. Several hyperbolic geometric models have been utilized in previous studies, including the Poincaré ball model (Ganea et al., 2018), the Poincaré half-plane model (Tifrea et al., 2018), the Klein model (Gulcehre et al., 2018), and the Lorentz model (Nickel and Kiela, 2018). These models are

mathematically equivalent.

We choose the Lorentz model as the eigenspace in our framework due to the numerical stability and computational simplicity provided by its exponential/logarithmic maps.

3.1. The Lorentz Model

An n-dimensional Lorentz model is the Riemannian manifold $\mathbb{L}_k^n = (\mathcal{L}^n, \mathfrak{g}_X^k)$ with negative curvature k $(k > 0)$. $\mathfrak{g}_X^k = diag(-1, 1, \cdots, 1)$ is the Riemannian metric tensor. $\mathcal{L}^n = \left\{\mathrm{x} \in \mathbb{R}^{n+1} | \langle \mathrm{x}, \mathrm{x} \rangle_{\mathcal{L}} = \frac{1}{k}, x_0 > 0\right\}$ is a point set, which corresponds to the upper sheet of a hyperboloid in an $(n+1)$ dimensional Minkowski space with the origin. $\langle \mathrm{x}, \mathrm{y} \rangle_{\mathcal{L}}$ is the Lorentzian inner product of two point x and y in Riemannian manifold, which is defined as follows:

$$\langle \mathrm{x}, \mathrm{y} \rangle_{\mathcal{L}} = -x_0 y_0 + \sum_{i=1}^{n} x_i y_i, \tag{1}$$

Tangent Space. Each point in Riemannian manifold $\mathbb{L}_k^n$ has the form of $\mathrm{x} = \begin{bmatrix} x_0 \\ x_s \end{bmatrix}$, where $\mathrm{x} \in \mathbb{R}^{n+1}, x_0 \in \mathbb{R}, \mathrm{x}_s \in \mathbb{R}^n$. Let x denote as a point in $\mathbb{L}_k^n$. Considering the Lorentzian inner product (Eq. 1), the orthogonal space of $\mathbb{L}_k^n$ at x can be represented as:

$$\mathcal{T}_{\mathrm{x}} \mathbb{L}_k^n = \left\{\mathrm{y} \in \mathbb{R}^{n+1} | \langle \mathrm{y}, \mathrm{x} \rangle_{\mathcal{L}^n} = 0\right\}, \tag{2}$$

where $\mathcal{T}_{\mathrm{x}} \mathbb{L}_k^n$ is a Euclidean subspace of $\mathbb{R}^{n+1}$. Specifically, the tangent space at the origin is defined as $\mathcal{T}_0 \mathbb{L}_k^n$.

Exponential and Logarithmic Maps. Given a point $\mathrm{x} \in \mathcal{T}_{\mathrm{x}} \mathbb{L}_k^n$ in tangent space. The tangent space is a Euclidean subspace. The exponential map is a mapping from the tangent space $\mathcal{T}_{\mathrm{x}} \mathbb{L}_k^n$ to the hyperbolic space $\mathbb{L}_k^n$, which move along the geodesic γ, satisfying $\gamma(0) = \mathrm{x}$ and $\gamma'(0) = \mathrm{z}$. The exponential map can be defined as:

$$\begin{aligned} &\exp_{\mathrm{x}}^k(\mathrm{z}) = \cosh(\alpha)\,\mathrm{x} + \sinh(\alpha)\,\frac{\mathrm{z}}{\alpha}, \\ &\alpha = \sqrt{-K}\,\|\mathrm{z}\|_{\mathcal{L}}, \|\mathrm{z}\|_{\mathcal{L}} = \sqrt{\langle \mathrm{z}, \mathrm{z} \rangle_{\mathcal{L}}}, \end{aligned} \tag{3}$$

The logarithmic map is the reversed mapping that maps back to the tangent space $\mathcal{T}_{\mathrm{x}} \mathbb{L}_k^n$. Given a point $\mathrm{y} \in \mathbb{L}_k^n$ in hyperbolic space, the logarithmic map can be defined as:

$$\begin{aligned} &\log_{\mathrm{x}}^k(\mathrm{y}) = \frac{\cosh^{-1}(\beta)}{\sqrt{\beta^2 - 1}}(\mathrm{y} - \beta \mathrm{x}), \\ &\beta = k \langle \mathrm{x}, \mathrm{y} \rangle_{\mathcal{L}} \end{aligned} \tag{4}$$

3.2. Lorentzian version of Function

The Lorentzian version of a function $f^{\otimes}$ to map from $\mathbb{L}_k^n$ to $\mathbb{L}_k^m$ for two points $\mathrm{x} = (x_0, \cdots, x_n) \in \mathbb{L}_k^n$ and $\mathrm{v} = (v_0, \cdots, v_n) \in \mathcal{T}_{\mathbf{0}} \mathbb{L}_k^n$ is defined as:

$$\begin{aligned} f^{\otimes^k}(\mathrm{x}) &= \exp_0^k\left(\hat{f}\left(\log_0^k(\mathrm{x})\right)\right), \\ \hat{f}(\mathrm{v}) &= (0, f(v_1, \cdots, v_n)), \end{aligned} \tag{5}$$

where $f : \mathbb{R}^n \to \mathbb{R}^m (n, m > 2)$, $\exp_0^k : \mathcal{T}_0 \mathbb{L}_k^n \to \mathbb{L}_k^m$, $\log_0^k : \mathbb{L}_k^n \to \mathcal{T}_0 \mathbb{L}_k^m$. k is the constant curvature.

Given a point $\mathrm{v} = (v_0, \cdots, v_n) \in \mathcal{T}_0 \mathbb{L}_k^n$, existing methods directly apply Euclidean transformations to all coordinates $(v_0, \cdots, v_n)$ within tangent spaces, such as (Chami et al., 2019; Liu et al., 2019). Different from previous methods, canonical Lorentzian version transformation only leverages the Euclidean transformations on the last n coordinates$(v_1, \cdots, v_n)$ in tangent spaces, and the first coordinate (v_0) is set to 0, which satisfying the constraint in Eq. (2).

4. Methodology

Here, we describe our proposed framework approach, which is depicted in Figure 2. Our framework mainly consists of three steps: parameter-free neighborhood feature propagation in Euclidean space, Lorentzian linear feature transformation in hyperbolic space and graph node labels prediction in Euclidean space. According to the paper (Wu et al., 2019), it has been observed that the inclusion of a nonlinear activation layer can impact the performance of linear GCNs. Therefore, in our framework, we remove the nonlinear activation layer after the Lorentzain feature transformation layer to simplify the model architecture and further improve the performance of our model.

4.1. Parameter-Free Propagation

To mitigate the over-smoothing problem, we adopt a personalized propagation scheme to effectively balance the contributions of graph structure and node features. As shown in Figure 2, given a graph $G = (\mathcal{V}, E)$ with n nodes and m edges, where $\mathcal{V}$ represents the set of nodes and E represents the set of edges.

In addition, $\tilde{G} = \left(\mathcal{V}, \tilde{E}\right)$ is defined as the self-looped graph, we attach a self-loop to each node in the original graph G. Let denote the adjacency matrix of G as $\mathtt{A}$ and the diagonal degree matrix as $\mathtt{D}$. Therefore, we also define the adjacency matrix and diagonal degree matrix of the self-looped graph $\tilde{G}$ as $\tilde{\mathtt{A}} = \mathtt{A} + \mathtt{I}$ and $\tilde{\mathtt{D}} = \mathtt{D} + \mathtt{I}$, where $\mathtt{I}$ is the identity matrix, $\mathtt{X} \in \mathbb{R}^{n \times d}$ is the node feature matrix, and each node v is associated with a d dimensional feature vector represented by $\mathtt{X}_v$.

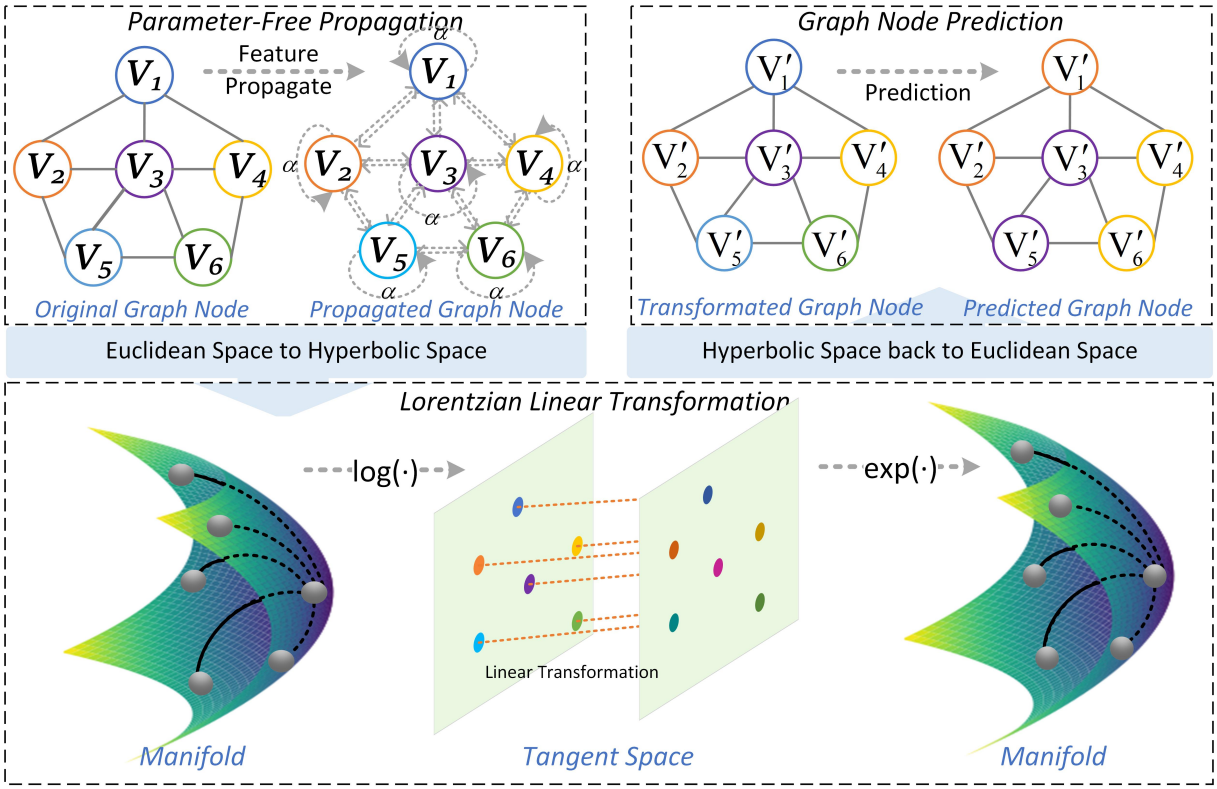

Figure 2: The framework of Lorentzian Linear Graph Convolutional Networks for node Classification.

The linear propagation matrix $\mathtt{P}$ can be written as:

$$\mathtt{P} = (\mathtt{D}+\mathtt{I})^{-1/2}(\mathtt{A}+\mathtt{I})(\mathtt{D}+\mathtt{I})^{-1/2}. \quad (6)$$

We precompute the information transfer between a node and its n-power neighbors as follows:

$$\mathtt{H}^{(l+1)} = (1-\alpha)\,\mathtt{P}\mathtt{H}^{(l)} + \alpha\mathtt{X}, \quad (7)$$

where $\alpha \in (0,1]$ is the teleport probability of the topic-sensitive PageRank. The initial node features $\mathtt{X}$ as topics to be ranked in the topic-sensitive PageRank algorithm. The nodes feature propagation with Eq. (7) guarantees that $\mathtt{H}^{(l)}$ is consistently influenced by both the graph structure and the initial node features $\mathtt{X}$ with a fixed proportion α, which enables us to acquire the graph node features $\mathtt{H}^{(n)}$ obtained through n-power propagations.

4.2. Lorentzian Linear Transformation

Theoretically, the linear transformation operations in Euclidean space cannot be directly applied in the hyperbolic space. Consequently, we define the linear transformation under Lorentz model to ensure the features transformation processing adhere to the hyperbolic geometry. In order to apply the linear transformation in hyperbolic space, following Lorentz version, we derive the Lorentzian matrix-vector multiplication as:

Definition: Given two points $\mathtt{x} = (x_0, \cdots, x_n) \in \mathbb{L}_k^n$, $\mathtt{v} = (v_0, \cdots, v_n) \in \mathcal{T}_0\mathbb{L}_k^n$, and $\mathtt{M}$ is a linear map from $\mathbb{R}^n \to \mathbb{R}^m$ $(n, m > 2)$ with matrix representation, we have:

$$\begin{aligned} \mathtt{M}^{\otimes^k}(\mathtt{x}) &= \exp_0^k\left(\hat{\mathtt{M}}\left(\log_0^k(\mathtt{x})\right)\right), \\ \hat{\mathtt{M}}(\mathtt{v}) &= (0, \mathtt{M}(v_1, \cdots, v_n)). \end{aligned} \quad (8)$$

$\mathtt{M}$ is a $m \times n$ matrix, $\mathtt{M}'$ is a $l \times n$ matrix, $\mathtt{x} \in \mathbb{L}_k^n$, $\mathtt{M}^{\otimes^k}\mathtt{x} = \mathtt{M}^{\otimes^k}(\mathtt{x})$, we have matrix associativity as: $(\mathtt{M}'\mathtt{M}) \otimes^k \mathtt{x} = \mathtt{M}' \otimes^k (\mathtt{M} \otimes^k \mathtt{x})$.

A crucial distinction between Lorentzian matrix-vector multiplication and other matrix-vector multiplications on the hyperboloid model lies in the size of the matrix $\mathtt{M}$. Assuming that a n-dimensional feature needs to be transformed into a m-dimensional feature, it is natural for the size of the matrix $\mathtt{M}$ to be $m \times n$. This requirement is fulfilled by Lorentzian matrix-vector multiplication. However, in other methods (Chami et al., 2019; Liu et al., 2019), the size of the matrix $\mathtt{M}$ is $(m+1) \times (n+1)$, which leads to the constraint of tangent spaces cannot be satisfied. Therefore, the Lorentzian transformation strictly adheres to hyperbolic geometry and effectively preserves the graph structure and properties within the hyperbolic space.

As shown in Figure 2, after acquiring $\mathtt{H}^{(n)}$, we apply the exponential mapping (Eq. 3) to map the learned node features $\mathtt{H}^{(n)}$ into the hyperbolic

space and then perform Lorentzian linear transformation within the hyperbolic space using Eq. 8. This specific process is given by the following:

$$\mathrm{M}^{\otimes^k}\left(\mathrm{H}^{(n)}\right) = \exp_0^k\left(\hat{\mathrm{M}}\left(\log_0^k\left(\exp_0^k\left(\mathrm{H}^{(n)}\right)\right)\right)\right), \quad (9)$$

where $\exp_0^k : \mathcal{T}_0\mathbb{L}_k^n \to \mathbb{L}_k^m$, $\log_0^k : \mathbb{L}_k^n \to \mathcal{T}_0\mathbb{L}_k^m$. k is the constant curvature. Since the tangent space $\mathcal{T}_0\mathbb{L}^n$ is a subspace of Euclidean space $\mathbb{R}^n$, Euclidean linear transformations can be applied to tangent spaces.

4.3. Graph Node Prediction

Finally, we use logarithm mapping (Eq. 4) to map the transformed node features back into Euclidean space for node prediction. The process is as follows:

$$\begin{aligned} \mathrm{H}^{(n)'} &= \log_0^k\left(\mathrm{M}^{\otimes^k}\left(\mathrm{H}^{(n)}\right)\right), \\ \hat{\mathrm{Y}} &= \operatorname{argmax}\left(\mathrm{H}^{(n)'}\right), \end{aligned} \quad (10)$$

where $\hat{\mathrm{Y}}$ is the index of the maximum probability in the category.

5. Experiments

5.1. Datasets

To validate the effectiveness and robustness of our framework, we conduct experiments on five publicly node classification datasets. The task can be performed under both semi-supervise and fully supervised learning. We used Cora, Citeseer, and PubMed (Sen et al., 2008) with semi-supervised learning. In these datasets, papers and their citation links are modeled as graphs, where nodes are papers and edges are citation links. For a fair comparison, we adopted the data splitting and evaluation method commonly used in previous studies.

With fully supervised learning, we use another two real world networks datasets, including Disease and Airport datasets (Zhang et al., 2021b). The Disease dataset is a graph with tree structure where node features indicate the susceptibility to the disease. The Airport dataset is a dataset where nodes represent airports and edges are the airline routes. In the Airport dataset, we utilize one-hot encoding to represent the nodes as node features. We split nodes in Disease dataset into $30/10/60\%$ and Airport dataset into $70/15/15\%$ for training, validation, and test sets, respectively. These split ratio is in line with (Chami et al., 2019; Zhang et al., 2021b; Chen et al., 2022).

These five datasets statistics are described in Table 1, where δ refers to Gromovs δ-hyperbolicity. The lower δ, the more tree-like of the graph. From Table 1, we see that the Cora dataset has the highest δ, which indicates that the hierarchical structure of Cora is not obvious.

Dataset	Nodes	Edges	Label	Features	δ
Cora	2,708	5,429	7	1,433	11
Citeseer	3,327	4,732	6	3,703	4.5
PubMed	19,717	44,338	3	500	3.4
Disease	1,044	1,043	2	1,000	0
Airport	3,188	18,631	4	3,188	1

Table 1: Dataset statistic on five datasets.

5.2. Baselines and Setup

Baselines. In the evaluation on citation network datasets, we compared our model L^{2}GC with 16 baselines. These baselines are grouped into four categories by their space used and type of GCN.

- Nonlinear models with Euclidean space. These models are traditional GCN and their variants, they are GCN (Kipf and Welling, 2017), GAT (Veličković et al., 2018), APPNP (Gasteiger et al., 2018), GraphHeat (Xu et al., 2019), ElasticGNN (Liu et al., 2021), and SCGNN (Liu et al., 2023).
- Linear models with Euclidean space. These models used liner transformation in GCN, they are SGC (Wu et al., 2019), SIGN-linear (Frasca et al., 2020), DGC (Wang et al., 2021b), G^{2}CN (Li et al., 2022) and FLGC (Cai et al., 2023).
- Nonlinear models with hyperbolic space. These model extend GCN to hyperbolic space, they are HGCN (Chami et al., 2019), HAT (Zhang et al., 2021a), LGCN (Zhang et al., 2021b), HGCL (Liu et al., 2022) and HYBONET (Chen et al., 2022).
- Linear model with hyperbolic space. Our model (L^{2}GC) extend linear GCN to hyperbolic space. It is worth noting that there are no other linear models on hyperbolic spaces. Our proposed L^{2}GC is the first simplification of nonlinear GCN models in hyperbolic spaces.

For a fair comparison, we report the best results of these models from the corresponding literature. Due to space limitations, a detailed description of these baselines is presented in Appendix A.

Setup. To train our model, we employ the Adam optimizer (Kingma and Ba, 2014) and take cross-entropy function as our loss function. The hyperparameters are determined by Bayesian optimization. The parameter optimization scopes are: learning rate (λ): $[0, 2]$; weight decay (w): $\left[1^{-10}, 1^{-1}\right]$; retention probability (α): $[0, 1]$; propagation step (n): $[0, 30]$. For the bias b, 1 represents the utilization of bias and 0 denotes the opposite. The dropout probability d is set to 0 for all experiments. The curvature K of the hyperbolic space is 1.

Finally, the hyperparameters used in our experiments are given in Table 3. All experiments were conducted on a machine with an NVIDIA GeForce

Space	Type	Method	Cora $\delta = 11$	Citeseer $\delta = 4.5$	PubMed $\delta = 3.4$
Euclidean	Nonlinear	GCN(Kipf and Welling, 2017)	81.5	70.3	79
		GAT(Veličković et al., 2018)	83.0	72.5 ± 0.7	79.0 ± 0.3
		APPNP(Gasteiger et al., 2018)	83.3	71.8	80.1
		GraphHeat(Xu et al., 2019)	83.7	72.5 ± 0.7	80.5
		ElasticGNN(Liu et al., 2021)	83.7 ± 0.2	72.2 ± 0.6	80.5 ± 0.1
		SCGNN(Liu et al., 2023)	**84.5 ± 0.3**	73.5 ± 0.5	80.8 ± 0.5
	Linear	SGC(Wu et al., 2019)	81.0 ± 0.0	71.9 ± 0.1	78.9 ± 0.0
		SIGN-linear(Frasca et al., 2020)	81.7	72.4	78.6
		DGC(Wang et al., 2021b)	83.3 ± 0.0	73.3 ± 0.1	80.3 ± 0.1
		G^{2}CN(Li et al., 2022)	82.7	73.8	80.4
		FLGC(Cai et al., 2023)	84.0 ± 0.0	73.2 ± 0.0	81.1 ± 0.0
Hyperbolic	Nonlinear	HGCN(Chami et al., 2019)	81.3 ± 0.6	70.9 ± 0.6	78.4 ± 0.4
		HAT(Zhang et al., 2021a)	83.1 ± 0.6	71.9 ± 0.6	78.6 ± 0.5
		LGCN(Zhang et al., 2021b)	83.3 ± 0.7	71.9 ± 0.7	78.6 ± 0.7
		HYBONET(Chen et al., 2022)	80.2 ± 1.3	-	78.0 ± 1.0
		HGCL(Liu et al., 2022)	82.3 ± 0.5	72.1 ± 0.6	79.14 ± 0.7
	Linear	**L^{2}GC**(ours)	82.4 ± 0.0	**74.7 ± 0.0**	**81.3 ± 0.0**

Table 2: Test accuracy (%) of semi-supervised node classification on citation networks. "-" indicates that there is no result reported for this dataset in the literature.

GTX 1660 GPU with 6GB memory. We evaluate our model using the accuracy rate (ACC). The results are obtained by averaging the results of ten random runs.

Dataset	λ	w	α	n	b
Cora	0.6	3.0^{-5}	0.1	20	1
Citeseer	1.2	9.8^{-5}	0.1	20	1
PubMed	0.94	8.8^{-6}	0.05	23	0
Disease	1.6	1.9^{-8}	0.1	4	1
Airport	1.0	1.1^{-10}	0.1	9	0

Table 3: Hyper-parameter settings of our model.

5.3. Performance on Semi-supervised Node Classification

We present a summary of experimental results in Table 2. We analyse the results as follow:

Comparing with models in Euclidean spaces. Our L^{2}GC outperforms all previous SOTA nonlinear and linear models on both Citeseer and PubMed datasets. These datasets exhibit a distinct tree-like structure characterized by smaller values of δ. We attribute these successes to the ability of hyperbolic spaces to effectively capture the hierarchical structure implicitly in the data. Specifically, compared to SGC, our L^{2}GC achieves improvements by 1.2%, 3.8% and 3.0% absolutely on Cora, Citeseer, and PubMed, respectively. Compared with the current SOTA linear model (G^{2}CN), our approach achieves improvement by 1.2% and 0.2% absolutely on the Citeseer and PubMed. However, we observed that our approach does not exhibit a significant advantage when compared to most models on Cora dataset, which has the biggest δ. We think that the absence of a tree-like structure in the Cora dataset might be the reason why our approach does not perform as well on this task.

Comparing with models in hyperbolic spaces. Compared to nonlinear models with hyperbolic space, our approach also demonstrate superior performance on the Citeseer and PubMed datasets, since our model performs Lorentzian linear transformation operations solely in hyperbolic space. Specifically, compared with the recent SOTA model (HGCL), our approach has improved by 3.6% and 2.7% on Citeseer and PubMed datasets, respectively. However, our model fail behind the models in Euclidean space. This observation in line with other previous hyperbolic models, like (Chami et al., 2019; Zhang et al., 2021a,b; Liu et al., 2022; Chen et al., 2022). It is worth noting that there is no GCN model of linear type in hyperbolic space. Our proposed approach represents a notable advance, as it is the first attempt to simplify nonlinear GCN models specifically designed for hyperbolic spaces. This highlights the unique contribution and advancement of our proposed approach.

Space	Method	Disease $\delta = 0$	Airport $\delta = 1$
$\mathbb{E}$	GCN(2017)	69.7 ± 0.4	81.4 ± 0.6
	GAT(2018)	70.4 ± 0.4	81.5 ± 0.3
	SGC(2019)	69.5 ± 0.2	80.6 ± 0.1
	SCGNN(2023)	85.3 ± 0.4	-
$\mathbb{H}$	HGCN(2019)	82.8 ± 0.8	90.6 ± 0.2
	HAT(2021a)	83.6 ± 0.9	-
	LGCN(2021b)	84.4 ± 0.8	90.9 ± 1.7
	HGCL(2022)	93.4 ± 0.8	92.3 ± 1.0
	HYBONET(2022)	$\mathbf{96.0 \pm 1.0}$	90.9 ± 1.4
	L^{2}GC(ours)	94.4 ± 0.1	$\mathbf{94.0 \pm 0.1}$

Table 4: Test accuracy (%) of fully supervised node classification on Disease and Airport datasets. $\mathbb{E}$ and $\mathbb{H}$ represent Euclidean space and hyperbolic space, respectively. "-" indicates that there is no report for this dataset in the literature.

5.4. Performance on Fully supervised Node Classification

We further validate the effectiveness and robustness of our approach in different domains under fully supervised node classification. We utilize Disease and Airport two datasets, which have lower δ values. Other settings in these experiments remain consistent with semi-supervised tasks. Here, we compare with GCN, GAT, HGCN, SGC, HAT, LGCN, HGCL, HYBONET, and SCGNN models. For a fair comparison, we provide the results reported in the corresponding literature. The results are shown in Table 4.

In both datasets, our model improved large margin over Euclidean space model since the datasets have lower δ value. In hyperbolic space, our model achieves the best accuracy with the Airport dataset. Our model improves 3.4% and 1.8% over HYBONET and HGCL on accuracy, respectively. On the Disease dataset, our approach demonstrates an improvement over most previous models. Although L^{2}GC does not show a significant improvement over HYBONET on the Disease dataset, our model exhibits advantages in terms of reducing training time and increasing stability. Ultimately, this result highlights the ability of our model to effectively capture the hierarchical structure present in the data. Our model further enhances the performance of linear GCNs and demonstrates the advantage of leveraging hyperbolic geometry.

5.5. Analysis and Visualization

To analyse the effective and efficiency of our model, we further do extensive experiments and visualize some classification results.

Ablations. To analyse the impact of the components in L^{2}GC, we explore the performance of two model variants.

	Cora $\delta = 11$	Citeseer $\delta = 4.5$	PubMed $\delta = 3.4$
$Variant\ I$	**83.1**	73.0	79.4
$Variant\ II$	80.2	73.6	79.7
L^{2}GC	82.4	**74.7**	**81.3**

Table 5: The ablation experiments of L^{2}GC.

- $Variant\ I$: L^{2}GC with Personalized Propagation scheme and without Lorentz model, which indicates the features transformation stage in Euclidean space.
- $Variant\ II$: L^{2}GC without Personalized Propagation scheme and with Lorentz model, we use the SGC propagation scheme in the feature propagation stage.

We compare the test accuracy of all variants against L^{2}GC in Table 5. It is clear that the L^{2}GC outperforms the other two variants on Citeseer and PubMed datasets. However, on the Cora dataset, we observe that the performance of $Variant\ I$ improves without the hyperbolic component. This result could be attributed to the largest δ of the dataset. This observation demonstrates that hyperbolic space is suitable for tree-like structure.

Efficiency. To further validate the efficiency of our model, we compared its performance with other models, such as GCN, GAT, HGCN, HAT, HYBONET, SGC, DGC, G^{2}CN and FLGC. We plot the performance of these models relative to the training time of L^{2}GC on the PubMed dataset in Figure 3. The training time of L^{2}GC includes the feature propagation time for a fair comparison. Overall, Figure 3 shows that L^{2}GC is trained faster one or two orders of magnitude than other nonlinear models. We measure the training time on the same NVIDIA GeForce GTX 1660 GPU with 6GB memory.

Moreover, our model also simplifies the hyperbolic model by leveraging the structure of the linear model. We compared the amount of parameters with HYBONET, as shown in Table 6. It obviously shows that our model can maintain superior performance with significantly fewer parameters.

Model	HYBONET	L^{2}GC(ours)
Cora	23356	8604 ($\downarrow$ 63.16%)
Citeseer	59659	18520 ($\downarrow$ 68.95%)
PubMed	8360	1002 ($\downarrow$ 88.01%)

Table 6: Comparison on parameters number.

Visualizations. We visualized the test set of PubMed dataset and its predicted label distribution

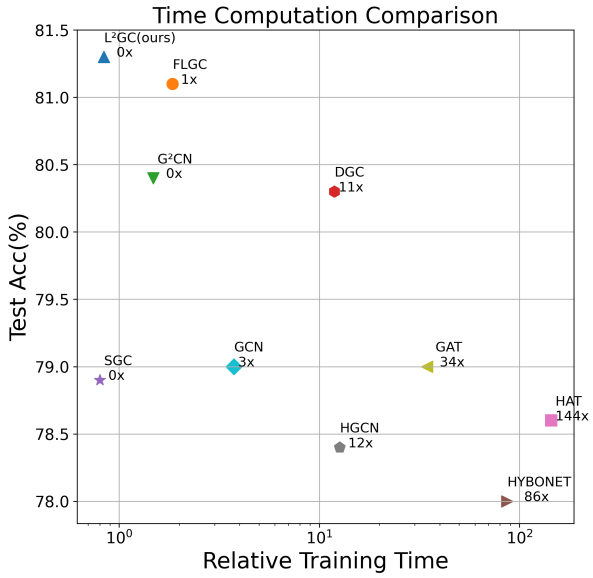

Figure 3: Comparison over training time on PubMed dataset.

in two dimensions with Principal Component Analysis (PCA). We compared the classification results with G^{2}CN and HYBONET on PubMed dataset. The results are shown in Figure 4 and Figure 5, respectively. From these figures, we see that our model can achieve better node classifcation results.

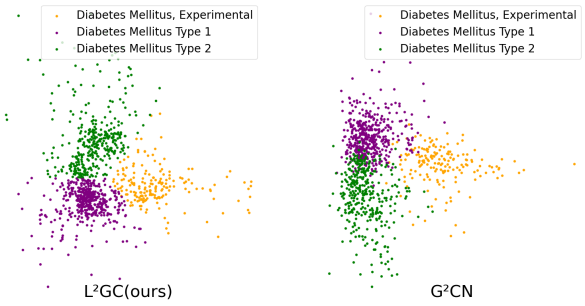

Figure 4: Comparsion node classifcation result with linear model (G^{2}CN). The color indicates the label category of nodes.

6. Conclusion and Future Work

In this paper, we propose a novel Lorentzian Linear Graph Convolutional Networks framework for node classification based on hyperbolic space. Our work is the first generalization of the linear GCN model to hyperbolic space, which capturing of the hierarchical structure in the data. Our approach not only leverages the strengths of the linear model, but also integrates the properties of hyperbolic spaces to achieve new SOTA results on the Citeseer and

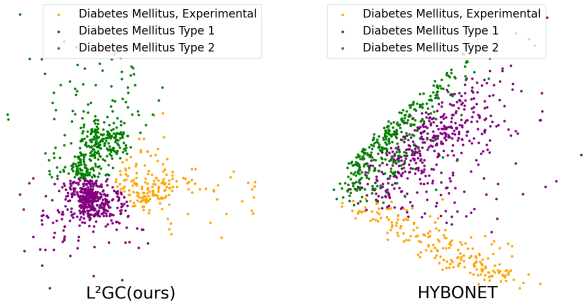

Figure 5: Comparsion node classifcation result with hyperboilc model (HYBONET). The color indicates the label category of nodes.

PubMed datasets. However, there are limitations to our approach, when compared to other Euclidean space models with large δ value. In the future, we plan to explore modeling our approach in a mixed spaces to enhance the ability of our model.

Acknowledgements

This work is supported by National Natural Science Foundation of China (Nos.62066033, 61966025); Inner Mongolia Natural Science Foundation (Nos.2020BS06001, 2022JQ05); Inner Mongolia Autonomous Region Science and Technology Programme Project (Nos.2023YFSW0001, 2022YFDZ0059); Collaborative Innovation Project of Universities and Institutes in Hohhot City. We also thank Qing Zhang for his helpful discussions.

References

Yaoming Cai, Zijia Zhang, Pedram Ghamisi, Zhihua Cai, Xiaobo Liu, and Yao Ding. 2023. Fully linear graph convolutional networks for semi-supervised and unsupervised classification. *ACM Transactions on Intelligent Systems and Technology*, 14(3):1–23.

Ines Chami, Rex Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems*, 32:4869–4880.

Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. Fully hyperbolic neural networks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5672–5686.

Aaron Clauset, Cristopher Moore, and Mark EJ Newman. 2008. Hierarchical structure and the

prediction of missing links in networks. *Nature*, 453(7191):98–101.

Lixia Feng, Yongqi Cai, Erling Wei, and Jianwu Li. 2022. Graph neural networks with global noise filtering for session-based recommendation. *Neurocomputing*, 472:113–123.

Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. 2020. Sign: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198*.

Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. 2018. Hyperbolic neural networks. In *Advances in neural information processing systems*, pages 5345–5355.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations (ICLR)*.

Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, et al. 2018. Hyperbolic attention networks. *arXiv preprint arXiv:1805.09786*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Chang Li and Dan Goldwasser. 2019. Encoding social information with graph convolutional networks forpolitical perspective detection in news media. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2594–2604.

Mingjie Li, Xiaojun Guo, Yifei Wang, Yisen Wang, and Zhouchen Lin. 2022. G^2CN: Graph gaussian convolution networks with concentrated graph filters. In *International Conference on Machine Learning*, pages 12782–12796. PMLR.

Jiahong Liu, Menglin Yang, Min Zhou, Shanshan Feng, and Philippe Fournier-Viger. 2022. Enhancing hyperbolic graph embeddings via contrastive learning. *arXiv preprint arXiv:2201.08554*.

Qi Liu, Maximilian Nickel, and Douwe Kiela. 2019. Hyperbolic graph neural networks. *Advances in neural information processing systems*, 32:12–22.

Xiaorui Liu, Wei Jin, Yao Ma, Yaxin Li, Hua Liu, Yiqi Wang, Ming Yan, and Jiliang Tang. 2021. Elastic graph neural networks. In *International Conference on Machine Learning*, pages 6837–6849. PMLR.

Yanbei Liu, Shichuan Zhao, Xiao Wang, Lei Geng, Zhitao Xiao, and Jerry Chun-Wei Lin. 2023. Self-consistent graph neural networks for semi-supervised node classification. *IEEE Transactions on Big Data*, 9(4):1186–1197.

Alessandro Muscoloni, Josephine Maria Thomas, Sara Ciucci, Ginestra Bianconi, and Carlo Vittorio Cannistraci. 2017. Machine learning meets complex networks via coalescent embedding in the hyperbolic space. *Nature communications*, 8(1):1615.

Maximillian Nickel and Douwe Kiela. 2018. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International conference on machine learning*, pages 3779–3788. PMLR.

Aravind Sankar, Yozen Liu, Jun Yu, and Neil Shah. 2021. Graph neural networks for friend ranking in large-scale social platforms. In *Proceedings of the Web Conference 2021*, pages 2535–2546.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine*, 29(3):93–93.

Junyuan Shang, Cao Xiao, Tengfei Ma, Hongyan Li, and Jimeng Sun. 2019. Gamenet: Graph augmented memory networks for recommending medication combination. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1126–1133.

Yuanhe Tian, Guimin Chen, Yan Song, and Xiang Wan. 2021. Dependency-driven relation extraction with attentive graph convolutional networks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4458–4471.

Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. 2018. Poincaré glove: Hyperbolic word embeddings. *ArXiv*, abs/1810.06546.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations (ICLR)*.

Yanda Wang, Weitong Chen, Dechang Pi, Lin Yue, Sen Wang, and Miao Xu. 2021a. Self-supervised

adversarial distribution regularization for medication recommendation. In *IJCAI*, pages 3134–3140.

Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. 2021b. Dissecting the diffusion process in linear graph convolutional networks. *Advances in neural information processing systems*, 34:5758–5769.

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

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR.

Bingbing Xu, Huawei Shen, Qi Cao, Keting Cen, and Xueqi Cheng. 2019. Graph convolutional networks using heat kernel for semi-supervised learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 1928–1934.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7370–7377.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983.

Yiding Zhang, Xiao Wang, Chuan Shi, Xunqiang Jiang, and Yanfang Ye. 2021a. Hyperbolic graph attention network. *IEEE Transactions on Big Data*, 8(6):1690–1701.

Yiding Zhang, Xiao Wang, Chuan Shi, Nian Liu, and Guojie Song. 2021b. Lorentzian graph convolutional networks. In *Proceedings of the Web Conference 2021*, pages 1249–1261.

A. Appendix

Details of the baselines: Nonlinear models based on Euclidean space:

- **GCN**(Kipf and Welling, 2017) is the extension of convolutional neural networks to handle graph-structured data, which can obtain a better data representation.
- **GAT**(Veličković et al., 2018) solves the problems of previous models based on graph convolution using masked-self attention layers.
- **APPNP**(Gasteiger et al., 2018) uses the relationship between GCN and PageRank algorithm to derive an improved propagation scheme based on personalized PageRank.
- **GraphHeat**(Xu et al., 2019) utilizes thermal kernels to enhance low-frequency filters and enhance the smoothness of signal changes on the graph.
- **ElasticGNN**(Liu et al., 2021) designs an Elastic GNN with a new message passing mechanism by adding L_1 regularization on top of traditional GNN's L_2 regularization.
- **SCGNN**(Liu et al., 2023) performs graph data augmentation and leverages a self-consistent constraint to maximize the mutual information of the unlabeled nodes across different augmented graph views.

Linear models based on Euclidean space:

- **SGC**(Wu et al., 2019) simplifies the GCN by removing the nonlinear activation function and reducing the entire GCN architecture to a direct multi-class logistic regression on preprocessed features.
- **SIGN-linear**(Frasca et al., 2020) proposes a universal probability graph sampler that constructs training batches by sampling the original graph. The graph sampler can be designed according to different schemes.
- **DGC**(Wang et al., 2021b) decouples the terminal time and the feature propagation steps, making it more flexible and capable of exploiting a very large number of feature propagation steps.
- **G^{2}CN**(Li et al., 2022) has developed a new spectral analysis framework from a detailed spectral analysis called concentration analysis.
- **FLGC**(Cai et al., 2023) linearizes GCN and then decouple it into neighborhood propagation and prediction stages, resulting in a flexible framework.

Nonlinear models based on hyperbolic space:

- **HGCN**(Chami et al., 2019) aggregates the expressive power of GCN and hyperbolic geometry to learn node representations in scale-free graphs or hierarchical graphs
- **HAT**(Zhang et al., 2021a) introduces hyperbolic attention networks to endow neural networks with enough capacity to match the complexity of data with hierarchical and power-law structure.
- **LGCN**(Zhang et al., 2021b) reconstructs the Lorentz version of GCN, making the operation of the hyperbolic neural network strictly comply with the definition of hyperbolic geometry.
- **HGCL**(Liu et al., 2022) takes advantage of contrastive learning to enhance the representation ability of the hyperbolic graph model.
- **HYBONET**(Chen et al., 2022) builds hyperbolic networks based on the Lorentz model by adapting the Lorentz transformations (including boost and rotation) to formalize essential operations of neural networks.