Beyond Euclidean Realms: Hyperbolic Node Embeddings

Divyanshu Bhatt*, Hitesh Donepudi*, Mirza Affan Ahmad Baig*, Tejadhith S*
Indian Institute of Technology Hyderabad
{ee21btech11016, ee21btech11017, ee21btech11034, ee21btech11055}@iith.ac.in

Abstract

Graphs are ubiquitous as a form of data and machine learning on graphs has applications ranging from drug design to recommendation systems. The knowledge extraction and representation from graphs using machine learning is a challenge as graphs are not structured data. Feature extraction using traditional machine learning algorithms has been the go-to method to encode structural information from graphs until the advent of neural network based graph representation models. But traditional Graph Neural Network methods are limited by their ability to represent Euclidean geometries and struggle to represent datasets with a non-Euclidean latent anatomy defined by a tree-like structure. Hyperbolic embeddings offset this problem due to their ability to represent these hierarchical distributions accurately. We explore Hyberbolic Graph Neural Networks through different Hyperbolic models like the Poincaré model and the Lorentz model.

1. Introduction

The need for graph representation learning is justified through its various applications in recommendation systems [1] [2], human protein identification [3], and molecular design [4]. A popular approach to analyze a network is to embed its vertices onto a low-dimensional Euclidean space and apply machine learning algorithms in this space.

Different algorithms [5] [6] [7] [8] have been proposed for embedding the nodes of the graph onto a low dimensional Euclidean space like node2vec and DeepWalk, which use random walks to preserve the community structure of nodes. Another popular deep learning method uses Graph Neural Networks (GNNs).

Previous works in this domain have assumed the embedded vector space to be Euclidean. However, recent research [9] [10] [11] [12] in deep learning models has shown that using non-Euclidean Riemannian manifolds like hyperbolic geometries can capture the structures of networks better due to the hierarchical structure of real-life graphs.

In this paper, we take a bird's eye view on node embedding techniques before delving into the intricacies of Hyberbolic Graph Embeddings. We analyze traditional node feature extraction techniques like Node Centrality Measures, Clustering Coefficients and Graphlet Degree Vectors. We take a mathematical approach in analyzing Graphlet Degree Vector by finding closed form solutions for it. We also study popular Random Walk based node embedding algorithms like node2vec and DeepWalk. We then examine Graph Neural Networks, where we study different architectures like Graph Convolutional Networks (GCNs), GraphSAGE and Graph Attention Networks.

The aforementioned literature gives us a foundation to study Hyperbolic Node Embeddings in a meticulous manner, where we break down the mathematical representations of two closed form expressions in the Hyperbolic Space - the Poincaré model and the Lorentz model.

2. Traditional Node Features Extraction

Traditional node feature extraction involves capturing relevant characteristics associated with individual nodes within a graph. These features help understand the roles and properties of nodes in the graph and can be used for various tasks, including classification, prediction, and visualization through their embeddings.

Node Centrality: It provides insights into which node is most influential or central within a network and captures the importance of neighboring nodes. Various ways of measuring Node Centrality are

a) Betweenness Centrality: It helps us identify whether a node is acting as a bridge or bottleneck in a network.

$$c_v = \sum_{s \neq v \neq t} \frac{\#(\text{shortest path between } s \text{ and } t \text{ that contains } v)}{\#(\text{shortest path between } s \text{ and } t)}$$

b) Closeness Centrality: It indicates the closeness of a node to all other nodes. It is a measure of shortest path length from current node to all the other nodes present

*Equal contribution.
†Implementation for the results could be found in the following GitHub link

within the graph.

$$c_v = \frac{1}{\sum_{u \neq v} (\text{shortest path length between } u \text{ and } v)}$$

c) Eigenvector Centrality: It focuses on the number of connections a node has while considering the centrality of the node, i.e. it focuses on both the quality and quantity of a node's connection. A node v is considered to be important if it is surrounded by important neighbor nodes $u \in N(v)$.

$$\begin{aligned} c_v &= \frac{1}{\lambda} \sum_{u \in N(v)} c_u \\ &= \frac{1}{\lambda} \sum_j A_{vj} c_j \end{aligned}$$

Taking $\mathbf{c}$ to be the vector with elements as the centrality of each node, the equation is reduced to

$$A\mathbf{c} = \lambda \mathbf{c}$$

with $\lambda_{\max}$ as the largest eigenvalue, and the eigenvector corresponding to $\lambda_{\max}$, i.e. $\mathbf{c}_{\max}$, contains the centralities of each node.

Clustering Coefficient: The clustering coefficient is a metric used to measure the degree of connectivity of nodes within a network. It quantifies how well connected v's neighboring nodes are

$$e_v = \frac{\#(\text{edges among neighboring nodes})}{\binom{k_v}{2}}$$

Graphlet Degree Vector: Similar to how the clustering coefficient counts the number of triangles, the Graphlet Degree Vector of a graph gives information on the number of different sub-graphs present in the network.

A graphlet is a sub-graph that describes the structure of a node's neighborhood. The Graphlet Degree Vector is a vector of dimension n, where n is the total number of possible graphlets that tells us the frequency of each type of graphlet. Figure 1 displays all of the graphlets up to four nodes, or 15 possible configurations. We analyse how to compute the Graphlet vectors in the Appendix.

3. Random Walk Based Node Embeddings

Performing prediction and classification tasks over nodes in a graph requires us to embed the node features into a space. This representation is learnt by learning an encoder function f, which maps the nodes in the original graph to the embedded space while preserving the notion of similarity between the two nodes

$$\text{similarity}(u, v) \approx \mathbf{z}_v^T \mathbf{z}_u$$

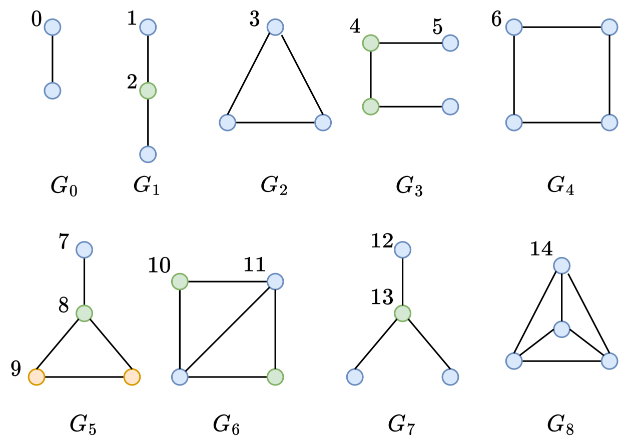


Figure 1. Constellation of graphlets containing all topologically unique node oribts. Nodes with the same color are isomorphic.

Given a graph $G(V, E)$, we find a mapping $f : u \to \mathbb{R}^N$ where $f(u) = \mathbf{z}_u$. Thus, the objective function is:

$$\max_f \sum_{u \in V} \log P(N_R(u) \mid \mathbf{z}_u)$$

where $N_R(u)$ is the neighborhood of u using a strategy R to traverse the nodes about u. $N_R(u)$ is a multiset, i.e. it can contain the same element more than once. The strategy R used depends on the embedding algorithm. In the upcoming subsection, we will explore how similarity can be defined using random walk-based embedding algorithms like DeepWalk and node2vec.

3.1. DeepWalk

DeepWalk [7] is an algorithm that learns latent representations of vertices in a network. The algorithm uses local information obtained from truncated random walks to learn latent representations by treating walks as the equivalent of sentences, using optimization techniques originally designed for language modeling.

Algorithm 1 performs r random walks of length l over all the vertices, and the $N_R(u)$ obtained for each vertex is passed through the SkipGram algorithm to update the latent representations.

3.2. node2vec

Node2vec [6] is a computationally efficient and widely used algorithm for network embedding. The algorithm balances the tradeoff between the homophily (i.e., network communities) and structural equivalence (i.e., structural roles of nodes) of the network through a flexible neighborhood sampling strategy using biased random walks.

The biasing of the random walks is done using a search bias parameter α, which samples the next node by using

Algorithm 1 DeepWalk(G, w, d, r, l)

```
Input: graph $G(V, E)$
  window size $w$
  embedding size $d$
  walks per vertex $r$
  walk length $l$
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V|\times d}$
  Initialization: Sample $\Phi$ from $\mathcal{U}^{|V|\times d}$
  Build a binary Tree $T$ from $V$
  for $i = 0$ to $r$ do
    $\mathcal{O} = \text{Shuffle}(V)$
    for each $u_i \in \mathcal{O}$ do
      $\mathcal{N}_R^{u_i} = RandomWalk(G, u_i, l)$
      SkipGram($\Phi$, $\mathcal{N}_R^{u_i}$, $w$)
    end for
  end for
```

parameters p and q, which are the return parameters and in-out parameters, respectively.

Algorithm 2 The *node2vec* algorithm.

```
node2vecWalk (Graph $G' = (V, E, \pi)$, Start node $u$,
  Length $l$)
  Inititalize $walk$ to $[u]$
  for $walk\_iter = 1$ to $l$ do
    $curr = walk[-1]$
    $V_{curr}$ = GetNeighbors($curr$, $G'$)
    $s$ = AliasSample($V_{curr}, \pi$)
    Append $s$ to $walk$
  end for
  return $walk$
```

The pseudocode for node2vec is given in Algorithm 2. In any random walk, there is an implicit bias due to the choice of the start node u. Since we learn representations for all nodes, we offset this bias by simulating r random walks of fixed length l starting from every node. At every step of the walk, sampling is done based on the transition probabilities π_{vx}. Thus, the node2vecWalk function computes and returns $N_R(u)$ as the $walk$ variable.

4. Graph Neural Networks

Consider a graph G, with a vertex set V, adjacency matrix A and a feature matrix $X \in \mathbb{R}^{m\times|V|}$. If there are no external features provided for the nodes, then X is an Identity matrix.

Graph Neural Networks (GNNs) can be interpreted as a message-passing algorithm between the nodes of a graph. The message function at the l^{th} layer, i.e. $\texttt{MSG}^{(l)}$ transforms the embeddings into a message given by that node. The aggregation function at the l^{th} layer i.e. $\texttt{AGG}^{(l)}$ aggregates these messages. Mathematically, the message-passing algorithm can be shown as

$$\mathbf{m}_v^{(l)} = \texttt{MSG}_{\text{nbhd}}^{(l)}\left(\mathbf{h}_v^{(l-1)}\right) \qquad \forall v \in N(u)$$

$$\mathbf{m}_u^{(l)} = \texttt{MSG}_{\text{self}}^{(l)}\left(\mathbf{h}_u^{(l-1)}\right)$$

$$\mathbf{h}_v^{(l)} = \texttt{AGG}_{\text{self}}^{(l)}\left(\texttt{AGG}_{\text{nbhd}}^{(l)}\left(\left\{\mathbf{m}_v^{(l)} \mid v \in N(u)\right\}, \mathbf{m}_u^{(l)}\right)\right)$$

where $\mathbf{h}_i^{(l)}$ is the hidden state of the node i in layer l. $\texttt{MSG}_{\text{nbhd}}$ transforms the message from the neighbors of the node whereas $\texttt{MSG}_{\text{self}}$ transforms the message from the node itself.

At the 0^{th} layer the embeddings are the node features themselves, i.e. $\mathbf{h}_v^{(0)} = \mathbf{x}_v$ where $\mathbf{x}_v$ is the row corresponding to the node v in the feature matrix X. The message function $\texttt{MSG}$ is usually implemented as a neural network. The aggregation function $\texttt{AGG}$ is a permutation invariant function like summation, mean, maximum, etc.

There exist various Graph Neural Network architectures like Graph Convolutional Networks [13], GraphSAGE [14] and Graph Attention Networks [15] which we describe below.

4.1. Graph Convolutional Networks

GCNs operate by aggregating information from neighboring nodes, essentially taking the mean of messages from adjacent nodes. This approach treats all neighboring nodes equally and assigns uniform importance to all the neighbors.

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\frac{1}{|N(u)|}\sum_{u\in N(v)} \mathbf{W}^{(l)}\mathbf{h}_u^{(l)} + \mathbf{B}^{(l)}\mathbf{h}_v^{(l)}\right)$$

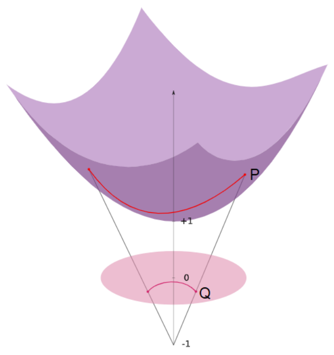
4.2. GraphSAGE

GraphSAGE modifies the GCN model by generalizing the linear function used over the messages from the neighbouring nodes.

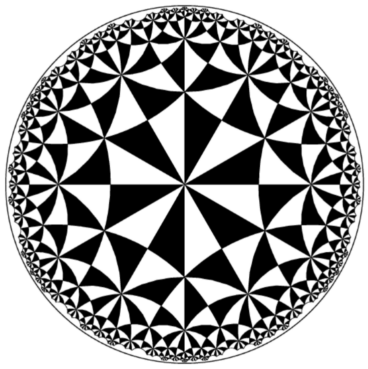
$$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{W}^{(l)}\texttt{CONCAT}\left(\mathbf{h}_v^{(l)}, \texttt{AGG}\left(\left\{\mathbf{h}_u^{(l)}, \forall u \in N(v)\right\}\right)\right)\right)$$

4.3. Graph Attention Networks

In contrast to GCNs, Graph Attention Networks learn an ”importance score” for each neighboring node concerning a particular node in the graph. This score reflects the relevance of the information from a given neighbor to the target node. Using the normalized importance score, a convex combination is carried out across the messages from the



(a) Lorentz Model



(b) Poincaré Ball

Figure 2. (a) Tangents are drawn such that they pass through $(-1, 0, 0)$. The point at which they intersect the Lorentz disc is considered as their embeddings in the Lorentz space. (b) The white and black triangles all possess the same area in the hyperbolic space but in the Poincaré Ball representation, the shapes farther away from the centre have a smaller area.

neighboring nodes.

$$e_{vu} = \mathtt{NN}\left(W^{(l)}\mathbf{h}_u^{(l-1)}, \mathbf{W}^{(l)}\mathbf{h}_v^{(l-1)}\right)$$
$$\alpha_{vu} = \frac{\exp\left(e_{vu}\right)}{\sum_{k\in N(v)}\exp(e_{vk})}$$
$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u\in N(v)} \alpha_{vu}\mathbf{W}^{(l)}\mathbf{h}_u^{(l-1)}\right)$$

5. Hyperbolic Spaces

Hyperbolic geometry is a Riemannian manifold with a constant negative curvature. It can be visualized using various mathematically equivalent models like the Poincaré model, Lorentz, and Klein models. We will mainly focus on the Poincaré model and Lorentz model in this analysis. Figure 2 depicts the visualizations for the Poincaré and Lorentz models.

5.1. Poincaré Model

The Poincaré model [16] is a Riemannian manifold defined by $\left(\mathcal{B}_c^N, g_{\mathbf{x}}^{\mathcal{B}}\right)$ i.e., for some $c > 0$

$$\mathcal{B}_c^N := \left\{\mathbf{x} \in \mathbb{R}^N : \|\mathbf{x}\|_2^2 < 1/c\right\}$$
$$g_{\mathbf{x}}^{\mathcal{B}} := \left(\lambda_{\mathbf{x}}^c\right)^2 g^E, \text{ where } \lambda_{\mathbf{x}}^c = \frac{2}{1 - c\|\mathbf{x}\|_2^2}$$

where g^E is the Euclidean metric i.e. the identity matrix $\mathcal{I}_N$. The manifold is an open unit ball, where the volume of the ball increases exponentially with respect to the radius. The origin in the Poincaré model is the same as the origin in the Euclidean space.

The distance between two points $\mathbf{x}, \mathbf{y} \in \mathcal{B}_c^N$ is given by

$$d_{\mathcal{B}}^c(\mathbf{x}, \mathbf{y}) := \frac{1}{\sqrt{c}}\cosh^{-1}\left(1 + \frac{2c\|\mathbf{x} - \mathbf{y}\|_2^2}{\left(1 - c\|\mathbf{x}\|_2^2\right)\left(1 - c\|\mathbf{y}\|_2^2\right)}\right)$$

The division term in the Poincaré distance could lead to numerical instability. We define an exponential map from the tangent space of a point $\mathbf{x} \in \mathcal{B}_c^N$ to the manifold, i.e. $\exp_{\mathbf{x}}^c : \mathcal{T}_{\mathbf{x}}\mathcal{B}_c^N \mapsto \mathcal{B}_c^N$ as

$$\exp_{\mathbf{x}}^c(\mathbf{v}) := \mathbf{x} \oplus_c \left(\tanh\left(\sqrt{c}\frac{\lambda_{\mathbf{x}}^c\|\mathbf{v}\|_2}{2}\right)\frac{\mathbf{v}}{\sqrt{c}\|\mathbf{v}\|_2}\right)$$

where $\oplus_c$ is the Möbius addition for any $\mathbf{x}, \mathbf{y} \in \mathcal{B}_c^N$ defined as

$$\mathbf{x} \oplus_c \mathbf{y} := \frac{\left(1 + 2c\langle\mathbf{x}, \mathbf{y}\rangle + c\|\mathbf{y}\|_2^2\right)\mathbf{x} + \left(1 - c\|\mathbf{x}\|_2^2\right)\mathbf{y}}{1 + 2c\langle\mathbf{x}, \mathbf{y}\rangle + c^2\|\mathbf{x}\|_2^2\|\mathbf{y}\|_2^2}$$

We also define a logarithmic map, i.e. the inverse of the exponential map, $\log_{\mathbf{x}}^c : \mathcal{B}_c^N \mapsto \mathcal{T}_{\mathbf{x}}\mathcal{B}_c^N$ as

$$\log_{\mathbf{x}}^c(\mathbf{v}) := \frac{2}{\sqrt{c}\lambda_{\mathbf{x}}^c}\tanh^{-1}\left(\sqrt{c}\|-\mathbf{x} \oplus_c \mathbf{v}\|_2\right)\frac{-\mathbf{x} \oplus_c \mathbf{v}}{\|-\mathbf{x} \oplus_c \mathbf{v}\|}$$

Parallel Transport is a differential geometric notion that helps in capturing the parallel movement of a vector in space. In the context of tangent spaces, it ensures that when a vector is parallel transported along the manifold, it maintains its direction relative to the local geometry of the surface at each point. It is defined as

$$P_{\mathbf{0}\to\mathbf{x}}^c(\mathbf{v}) = \log_{\mathbf{x}}^c\left(\mathbf{x} \oplus_c \exp_{\mathbf{0}}^c(\mathbf{v})\right)$$

5.2. Lorentz Model

The Lorentz model [12] [10] is a Riemannian manifold defined by $\left(\mathcal{L}_c^N, g_{\mathbf{x}}^{\mathcal{L}}\right)$ i.e., for some $c > 0$

$$\mathcal{L}_c^N := \left\{\mathbf{x} \in \mathbb{R}^{N+1} : \langle\mathbf{x}, \mathbf{x}\rangle_{\mathcal{L}} = -1/c\right\}$$
$$g_{\mathbf{x}}^{\mathcal{L}} := \operatorname{diag}\left(\begin{bmatrix}-1 & 1 & 1 & \cdots & 1\end{bmatrix}\right)$$

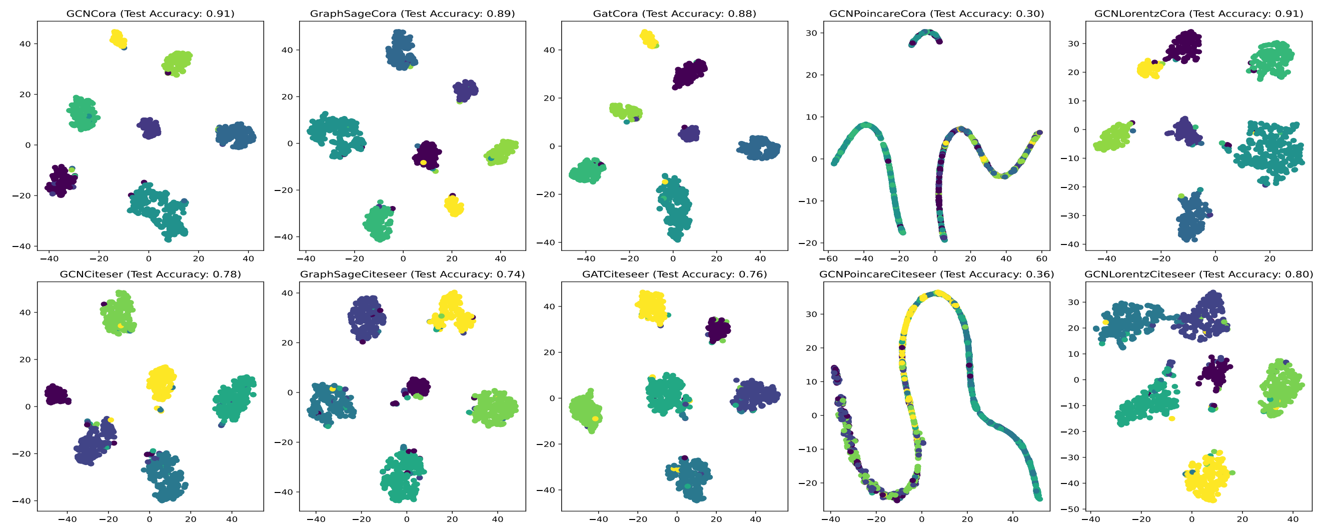


Figure 3. t-SNE plots of GNNs and HGNNs based embeddings on the Cora and CiteSeer dataset

Models	DeepWalk	node2vec	GCN	GraphSage	GAT	HGCN Poincaré	HGCN Lorentz
Cora	39.80%	34.15%	73.54%	87.38%	87.12%	29.52%	88.82%
CiteSeer	25.80%	22.15%	76.88%	72.13%	74.62%	30.54%	78.11%

Table 1. Test Accuracies on Cora and CiteSeer datasets averaged over 10 different initializations.

where $\langle \cdot, \cdot \rangle_{\mathcal{L}}$ is the Lorentz inner product defined on $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{N+1}$ as

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} = -x_0 y_0 + \sum_{k=1}^{N} x_k y_k$$

The distance function induced by the above Riemannian metric is

$$d_{\mathcal{L}}^c(\mathbf{x}, \mathbf{y}) := \frac{1}{\sqrt{c}} \cosh^{-1}\left(-c\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}\right)$$

The origin in the Lorentz model is defined as $\mathbf{0} : (1, 0, 0 \cdots, 0)$.The exponential map $\exp_{\mathbf{x}}^c : \mathcal{T}_{\mathbf{x}}\mathcal{L}_c^N \mapsto \mathcal{L}_c^N$ and the logarithmic map $\log_{\mathbf{x}}^c : \mathcal{L}_c^N \mapsto \mathcal{T}_{\mathbf{x}}\mathcal{L}_c^N$ for Lorentz model is defined as

$$\exp_{\mathbf{x}}^c(\mathbf{v}) := \cosh\left(\sqrt{c}\|\mathbf{v}\|_{\mathcal{L}}\right)\mathbf{x} + \left(\frac{\sinh\left(\sqrt{c}\|\mathbf{v}\|_{\mathcal{L}}\right)}{\sqrt{c}\|\mathbf{v}\|_{\mathcal{L}}}\right)\mathbf{v}$$

$$\log_{\mathbf{x}}^c(\mathbf{v}) := \left(\frac{\cosh^{-1}\left(-c\langle \mathbf{x}, \mathbf{v} \rangle_{\mathcal{L}}\right)}{\sinh\left(\cosh^{-1}\left(-c\langle \mathbf{x}, \mathbf{v} \rangle_{\mathcal{L}}\right)\right)}\right)\left(\mathbf{v} + c\langle \mathbf{x}, \mathbf{v} \rangle_{\mathcal{L}}\mathbf{x}\right)$$

Parallel Transport for this model is defined as

$$P_{\mathbf{0}\to\mathbf{x}}^c(\mathbf{v}) = \mathbf{v} + \frac{c\langle \mathbf{x}, \mathbf{v} \rangle_{\mathcal{L}}}{1 - c\langle \mathbf{0}, \mathbf{x} \rangle_{\mathcal{L}}}(\mathbf{x} + \mathbf{0})$$

6. Hyperbolic GNNs

6.1. Initialization of Feature Vectors

Initially, the feature vectors $X = \left[x_1, \cdots, x_{|V|}\right]$ lie in the Euclidean space. For working with the hyperbolic space, we map these feature vectors using an exponential map. In the Poincaré model, we consider the initial Euclidean space to itself be the tangent space of origin, i.e. $\mathcal{T}_0\mathcal{B}_c^N$, and we map it to $\mathcal{B}_c^N$ using the exponential map i.e.

$$\mathbf{x}^{\mathcal{B}} = \exp_{\mathbf{0}}^c\left(\mathbf{x}^E\right)$$

In the Lorentz model, we map the feature vectors to the tangent space of origin by concatenating a 0 in the front, and then map it to the hyperbolic space using the exponential map, i.e.

$$\mathbf{x}^{\mathcal{T}_0} = \texttt{CONCAT}\left(0, \mathbf{x}^E\right)$$
$$\mathbf{x}^{\mathcal{L}} = \exp_{\mathbf{0}}^c\left(\mathbf{x}^{\mathcal{T}_0}\right)$$

6.2. Feature Transformation

The notion of matrix multiplication in both models is defined as follows

$$\mathbf{W} \otimes_c^{\mathcal{B}} \mathbf{x}^{\mathcal{B}} = \exp_0^c\left(\mathbf{W}\log_0^c\left(\mathbf{x}^{\mathcal{B}}\right)\right)$$
$$\mathbf{W} \otimes_c^{\mathcal{L}} \mathbf{x}^{\mathcal{L}} = \exp_0^c\left(\texttt{CONCAT}\left(0, \left(\mathbf{W}\log_0^c\left(\mathbf{x}^{\mathcal{L}}\right)_{[1:n]}\right)\right)\right)$$

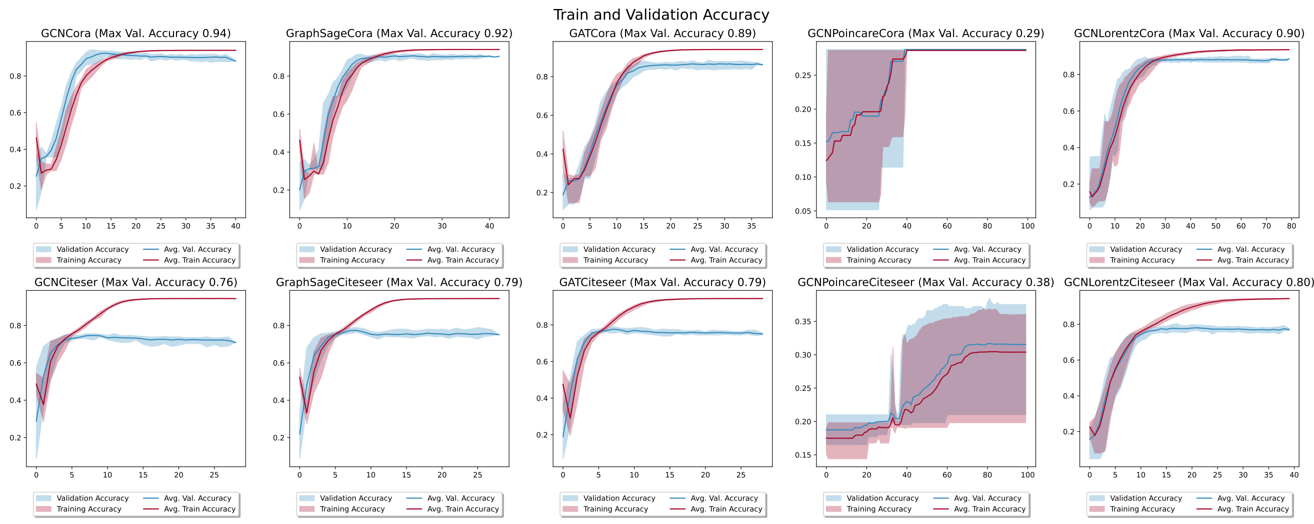


Figure 4. Accuracies of GNNs and HGNNs based embeddings on the Cora and CiteSeer dataset

We define the notion of vector addition in both models as

$$\mathbf{x}^{\mathcal{H}} \oplus \mathbf{b} = \exp^{c}_{\mathbf{x}^{\mathcal{H}}}\left(P^{c}_{\mathbf{0}\to\mathbf{x}^{\mathcal{H}}}(\mathbf{b})\right)$$

Here, both $\mathbf{W}, \mathbf{b}$ belong to the Euclidean space. Similarly, we define non-linear activation functions for hyperbolic space using projection of the points in the tangent space to the origin and then apply the Euclidean non-linearity layer to map them back to the manifold.

$$\sigma^{c_1,c_2}\left(\mathbf{x}^{\mathcal{B}}\right) = \exp^{c_2}_{\mathbf{0}}\left(\sigma\left(\log^{c_1}_{\mathbf{0}}\left(\mathbf{x}^{\mathcal{B}}\right)\right)\right)$$
$$\sigma^{c_1,c_2}\left(\mathbf{x}^{\mathcal{L}}\right) = \exp^{c_2}_{\mathbf{0}}\left(\texttt{CONCAT}\left(0, \sigma\left(\log^{c_1}_{\mathbf{0}}\left(\mathbf{x}^{\mathcal{L}}\right)\right)_{[1:n]}\right)\right)$$

6.3. Graph Neural Networks in Hyperbolic Space

The equations we defined in Section 4 are applicable to only Euclidean Graph Neural Networks. We thus have to re-define the Euclidean matrix multiplication and bias addition notations for the hyperbolic space. The updated equations for Hyperbolic Graph Convolutional Networks are given by

$$\mathbf{m}^{(l+1)}_{v} = \left(\mathbf{W}^{(l)} \otimes^{\mathcal{H}}_{c} \mathbf{h}^{(l)}_{v}\right) \oplus^{\mathcal{H}}_{c} \mathbf{b}^{(l)} \quad \forall v \in N(u) \cup \{u\}$$

$$\mathbf{h}^{(l+1)}_{u} = \sigma^{c_{l-1},c_l}\left(\exp^{c_l}_{\mathbf{0}}\left(\sum_{k\in N(u)} \frac{\log^{c_l}_{\mathbf{0}}\left(\mathbf{m}^{(l+1)}_{k}\right)}{|N(u)|} + \log^{c_l}_{\mathbf{0}}\left(m^{(l+1)}_{u}\right)\right)\right)$$

6.4. Multinomial Logistic Regression

After deriving the embeddings z_v for each node $v \in V$, we can map them to the tangent space of the origin using logarithmic maps and then perform Euclidean Logistic Regression to classify the nodes.

Another idea proposed [8] is to extend radial basis function networks to Riemannian manifolds. They learn a centroid for each class in the hyperbolic space and calculate the distance between z_v with each of these centroids i.e. $\phi_{vk} = d^{\mathcal{H}}_{c}\left(\mathbf{z}_v, \mathbf{C}_k\right)$ which represents the distance between the node and the centroid of the k^{th} class. We concatenate these distances and train a logistic regression classifier over it.

$$p(\mathbf{y}_v) = \text{softmax}\left(\mathbf{W}\texttt{CONCAT}\left(\phi_{v1}, \cdots, \phi_{v|\mathcal{C}|}\right)\right)$$

Hyperbolic Multinomial Logistic Regression [9] can also be used directly on the embeddings z_v to predict the class of node v.

7. Experiments

The Cora [17] and CiteSeer [18] datasets have been used to evaluate all the models used in this study. The Cora dataset consists of 2708 scientific publications classified into one of seven classes and the CiteSeer dataset consists of 3312 scientific publications classified into one of six classes.

We have evaluated and compared the following models - node2vec, DeepWalk, GCN, GraphSAGE, Graph Attention Network and the Poincaré and Lorentz Hyperbolic GCNs. The hyperparameters for the models have not been fine-tuned separately. The models have been evaluated 10 times, with different initializations each time. The final accuracy is computed by taking the average of all 10 runs. The curvature for the hyperbolic models has been taken as -1.

The resulting plots in Figure 3 and 4 showcase the accuracies of the models when run on the Cora and CiteSeer dataset which have been visualized using t-SNE.

The parameterized models output better accuracies than the Random Walk based counterparts. This could be attributed to the ability of graph neural networks to simultaneously capture the local and global information in a graph.

The Hyperbolic Lorentz model for Graph Neural Networks showcases a better empirical performance and has a higher accuracy across both datasets as compared to the other models. The same is not observed for the Poincaré model due to gradient instability. The division instability of the exponential and logarithmic maps of the Ponicaré model could be one of the potential causes for the highly fluctuating training loss.

8. Conclusions

Through this study, we have explored node embeddings both theoretically and practically all the way from the traditional machine learning based methods, to Random Walk based techniques and Euclidean Graph Neural Networks and finally Hyperbolic GNNs. The Poincaré and Lorentz models both take different mathematical approaches to embed graphs, using different notions of exponential and logarithmic maps. Our analysis shows the efficiency of these hyperbolic models in capturing the hierarchical structures of graphs with the Lorentz model outperforming the Poincaré model on the chosen datasets.

References

[1] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: A survey, 2022. 1

[2] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):346–353, jul 2019. 1

[3] Damian Szklarczyk, John Morris, Helen Cook, Michael Kuhn, Stefan Wyder, Milan Simonovic, Alberto Santos, Nadezhda Doncheva, Alexander Roth, Peer Bork, Lars Jensen, and Christian von Mering. The string database in 2017: quality-controlled protein-protein association networks, made broadly accessible. *Nucleic acids research*, 45, 10 2016. 1

[4] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L. Gaunt. Constrained graph variational autoencoders for molecule design, 2019. 1

[5] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications, 2018. 1

[6] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks, 2016. 1, 2

[7] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, aug 2014. 1, 2

[8] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, feb 2018. 1, 6

[9] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks, 2018. 1, 6

[10] Menglin Yang, Min Zhou, Zhihao Li, Jiahong Liu, Lujia Pan, Hui Xiong, and Irwin King. Hyperbolic graph neural networks: A review of methods and applications, 2022. 1, 4

[11] Ines Chami, Rex Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks, 2019. 1

[12] Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks, 2019. 1, 4

[13] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017. 3

[14] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018. 3

[15] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018. 3

[16] Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations, 2017. 4

[17] Andrew McCallum. Cora Dataset, 2017. 6

[18] Cornelia Caragea, Jian Wu, Alina Ciobanu, Kyle Williams, Juan Fernández-Ramírez, Hung-Hsuan Chen, Zhaohui Wu, and Lee Giles. Citeseerx: A scholarly big dataset. In Maarten de Rijke, Tom Kenter, Arjen P. de Vries, ChengXiang Zhai, Franciska de Jong, Kira Radinsky, and Katja Hofmann, editors, *Advances in Information Retrieval*, pages 311–322, Cham, 2014. Springer International Publishing. 6

A. Graphlet Degree Vector

We have proposed a closed form solution for finding the graphlet count as a function of adjacency matrix. Let $G = (V, E)$ be a graph with node set V and edge set E. Let $\mathbf{A}$ be the adjacency matrix corresponding to G. We have taken a simple, unweighted and undirected graph, therefore $\mathbf{A}_{ii} = 0$, $A_{ij} \in \{0, 1\}$ and $\mathbf{A}_{ij} = \mathbf{A}_{ji}$. We represent the number of graphlets as $G_{a,b}$ where a represent the graphlet in reference to 1 and b represent the node corresponding to that graphlet.

$$G_{0,0} = \sum_{j=1}^{N} \mathbf{A}_{ij}$$

$$G_{1,1} = \sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{jk}\mathbb{I}\{k \neq i\}$$
$$= \sum_{j=1}^{N} \mathbf{A^2}_{ij} - \mathbf{A^2}_{ii}$$

$$G_{1,2} = \sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{ik}\mathbb{I}\{j \neq k\}$$
$$= \sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{ik} - \mathbf{A^2}_{ii}$$

$$G_{2,3} = \sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{ik}\mathbf{A}_{jk}$$
$$= \mathbf{A^3}_{ii}$$

$$G_{3,4} = \sum_{l=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{jk}\mathbf{A}_{il}\mathbb{I}\{l \neq j\}\mathbb{I}\{l \neq k\}\mathbb{I}\{i \neq k\}$$
$$= \sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A^2}_{ij}\mathbf{A}_{ik} - \mathbf{A^3}_{ii} - \sum_{k=1}^{N}\sum_{j=1}^{N} \left(\mathbf{A}_{ij}\right)^2 \mathbf{A}_{jk}$$
$$- \mathbf{A^2}_{ii}\sum_{j=1}^{N} \mathbf{A}_{ij} + \sum_{j=1}^{N} \left(\mathbf{A}_{ij}\right)^3$$

$$G_{3,5} = \sum_{l=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{jk}\mathbf{A}_{kl}\mathbb{I}\{i \neq k\}\mathbb{I}\{i \neq l\}\mathbb{I}\{j \neq l\}$$
$$= \sum_{j=1}^{N} \mathbf{A^3}_{ij} - \mathbf{A^2}_{ii}\sum_{j=1}^{N} \mathbf{A}_{ij} - \mathbf{A^3}_{ii}$$
$$- \sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A^2}_{jj} + \sum_{j=1}^{N} \left(\mathbf{A}_{ij}\right)^3$$

$$G_{4,6} = \sum_{l=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{jk}\mathbf{A}_{kl}\mathbf{A}_{li}\mathbb{I}\{i \neq k\}\mathbb{I}\{j \neq l\}$$
$$= \mathbf{A^4}_{ii} - \left(\mathbf{A^2}_{ii}\right)^2 - \sum_{j=1}^{N} \left(\mathbf{A}_{ij}\right)^2 \mathbf{A^2}_{jj} + \sum_{j=1}^{N} \left(\mathbf{A}_{ij}\right)^4$$

$$G_{5,7} = \sum_{l=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{jk}\mathbf{A}_{jl}\mathbf{A}_{kl}\mathbb{I}\{i \neq k\}\mathbb{I}\{i \neq l\}$$
$$= \sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{jk}\mathbf{A^2}_{jk} - 2\sum_{j=1}^{N} \left(\mathbf{A}_{ij}\right)^2 \mathbf{A^2}_{ij}$$

$$G_{5,8} = \sum_{l=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{ik}\mathbf{A}_{il}\mathbf{A}_{kl}\mathbb{I}\{i \neq k\}\mathbb{I}\{i \neq l\}$$
$$= \sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{ik}\mathbf{A^2}_{ik} - 2\sum_{j=1}^{N} \left(\mathbf{A}_{ij}\right)^2 \mathbf{A^2}_{ij}$$

$$G_{5,9} = \sum_{l=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ik}\mathbf{A}_{il}\mathbf{A}_{jl}\mathbf{A}_{kl}\mathbb{I}\{j \neq k\}\mathbb{I}\{i \neq j\}$$
$$= \sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{il}\mathbf{A}_{jl}\mathbf{A^2}_{il} - \sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{ik}\mathbf{A}_{jk}\left(\mathbf{A}_{ij} + \mathbf{A}_{jk}\right)$$

$$G_{6,10} = \sum_{l=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{il}\mathbf{A}_{jk}\mathbf{A}_{jl}\mathbf{A}_{kl}\mathbb{I}\{i \neq k\}$$
$$= \sum_{l=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{il}\mathbf{A}_{jk}\mathbf{A}_{jl}\mathbf{A}_{kl} - \sum_{k=1}^{N}\sum_{j=1}^{N} \left(\mathbf{A}_{ij}\right)^2 \left(\mathbf{A}_{ij}\right)^2 \mathbf{A^2}_{jl}$$

$$G_{6,11} = \sum_{l=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{il}\mathbf{A}_{jk}\mathbf{A}_{ik}\mathbf{A}_{kl}\mathbb{I}\{j \neq l\}$$
$$= \sum_{l=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{il}\mathbf{A}_{jk}\mathbf{A}_{ik}\mathbf{A}_{kl} - \sum_{k=1}^{N}\sum_{j=1}^{N} \left(\mathbf{A}_{ij}\right)^2 \left(\mathbf{A}_{jk}\right)^2 \mathbf{A^2}_{ik}$$

$$G_{7,12} = \sum_{l=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{jk}\mathbf{A}_{jl}\mathbb{I}\{i \neq k\}\mathbb{I}\{i \neq l\}\mathbb{I}\{k \neq l\}$$
$$= \sum_{l=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{jk}\mathbf{A}_{jl} - \sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{jk}\left(2\mathbf{A}_{ij} + \mathbf{A}_{jk}\right)$$
$$+ 2\sum_{j=1}^{N} \left(\mathbf{A}_{ij}\right)^3$$

$$G_{7,13} = \sum_{l=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{ik}\mathbf{A}_{il}\mathbb{I}\{j \neq k\}\mathbb{I}\{j \neq l\}\mathbb{I}\{k \neq l\}$$
$$= \sum_{l=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{ik}\mathbf{A}_{il} - 3\mathbf{A^2}_{ii}\sum_{j=1}^{N} \mathbf{A}_{ij} + 2\sum_{j=1}^{N} \left(\mathbf{A}_{ij}\right)^3$$

$$G_{8,14} = \sum_{l=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} \mathbf{A}_{ij}\mathbf{A}_{ik}\mathbf{A}_{il}\mathbf{A}_{jk}\mathbf{A}_{jl}\mathbf{A}_{kl}$$