CS 229 Project: Final Report
Hyperbolic Graph Convolutional Network for Link Prediction on Synthetic Graphs

Damir Vrabac and Evan Laksono (dvrabac & elaksono)

Abstract

This work explores the embedding of synthetic graphs using Hyperbolic Graph Convolutional Network (HGCN) and evaluates its performance in link prediction. The synthetic graphs are generated with an underlying exponential geometry, giving rise to properties which mimic those of real networks. We showed a particular class of graphs in which embedding in a hyperbolic space provides a decisive advantage over embedding in an Euclidean space. We consistently attained high accuracy in link prediction for our synthetic graphs embedded using HGCN, despite the low-dimensionality of its embedding space. Our results also pointed to the importance of the underlying hyperbolic metric space of graphs for HGCN to perform better than its Euclidean counterpart.

1 Introduction

Graphs are known to be ubiquitous in nature, and their mathematical formulation proves to be a powerful theoretical tool to understand various phenomena ranging from social networks, knowledge graphs, protein networks, drug developments to e-commerce activities [1, 2]. The task of effective and accurate graph representations with minimal feature engineering is thus essential for tasks such as graph classifications, link prediction (LP) and node classification (NC) [3, 4].

Graphs present some challenges for representation learning, i.e. the embedding of graphs in a low dimensional space. Firstly, their representation should take into account the node edges, which vary for each individual node. Within the deep learning paradigm, the problem has been addressed by a Graph Convolutional Network (GCN), the state-of-the-art method which operates and embeds graphs in the Euclidean space [1, 2, 5]. The convolution which takes place on each individual layer encodes the information of the neighboring nodes. However, embedding in the Euclidean space introduces significant distortions when tree-like structures are represented, while such structures are prevalent in real networks, such as social [6] and biological networks [7]. This problem can be intuitively understood as tree-like structures exhibit exponential growth in the number of nodes deeper in their hierarchies, while the volume in Euclidean space only grows polynomially [8]. On the other hand, the tree-like graphs fit more naturally into the hyperbolic space and some works have demonstrated the possibility to embed such graphs with arbitrarily small distortions [8, 9]. Some recent work developed deep learning schemes which operate in the hyperbolic space [3, 4], and it has been proven that hyperbolic neural networks significantly improve the performance of prediction tasks on tree-like graphs.

We designed synthetic graphs which capture some of the features of real world data as our inputs, so that our theoretical investigation can be expected to be relevant to address real problems. We will elaborate the schemes to generate our graphs in Section 2. In this work, we used Hyperbolic Graph Convolutional Network (HGCN) to perform LP tasks on synthetic graphs. Since the incorporation of hyperbolic space into deep learning is a very recent development, our understanding of the techniques would benefit from experiments on synthetic graphs. The issue will be addressed in Section 3 of our report. Some details of our experiments, results and discussions are presented in section 4, with more emphasis on LP tasks. We carefully studied the performance of HGCN, and demonstrated that it exhibits a decisive advantage over GCN in performing LP for synthetic graphs generated with a hidden hyperbolic metric space [10, 11]. We will also discuss possible future works and applications in section 5, and conclude our findings in section 6.

2 Dataset: Synthetic Graphs

Our experiment used synthetic graphs as inputs. Our report would largely focus on LP task on graphs with a hidden hyperbolic metric space.

Hyperbolic Random Graphs (HRG). This type of graphs is the most important type of graphs that we used as inputs. We use this scheme because of two reasons. Firstly, the scheme works in the hidden hyperbolic metric space [10], which is a facet of graph hyperbolicity in conjunction with Gromov-δ metrics [4]. Interestingly, hyperbolicity in graphs has been shown to give rise to scale-free topology and high degree of clustering, both are important characteristics of real networks which exhibit hierarchies [10, 12]. The parameters in the scheme can thus be tuned in order to approximate some properties of a real data set, such as the distribution of the node degrees and the average size of the clusters. Secondly, this construction can also be understood as the outcome of reverse-engineering of graphs which can be intuitively expected to exploit the advantages of HGCN.

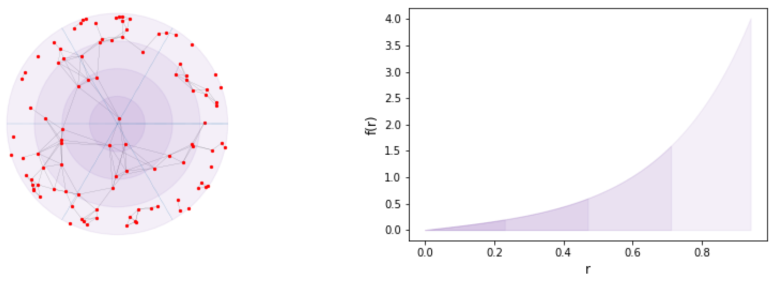


Figure 1: (Left) A random graph with $N = 100$, $\alpha = 4$, $R_0 = 0.95$, $n_r = 4$, and $n_\theta = 6$. (Right) The corresponding probability distribution function in r, divided into four equal bins.

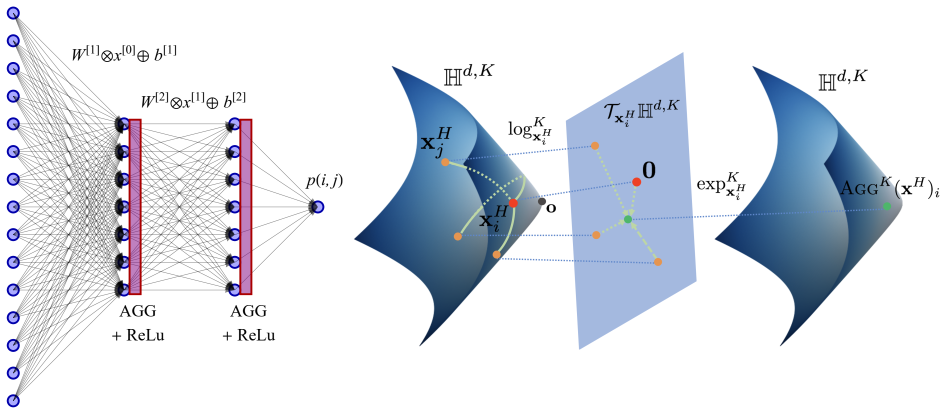

$W^{[1]} \otimes x^{[0]} \oplus b^{[1]}$
$W^{[2]} \otimes x^{[1]} \oplus b^{[2]}$
$p(i,j)$
AGG + ReLu
AGG + ReLu
$\mathbb{H}^{d,K}$
$\mathbf{x}_j^H$
$\log^K_{\mathbf{x}_i^H}$
$\mathbf{x}_i^H$
o
$\mathcal{T}_{\mathbf{x}_i^H}\mathbb{H}^{d,K}$
$\mathbf{0}$
$\exp^K_{\mathbf{x}_i^H}$
$\mathbb{H}^{d,K}$
$\text{AGG}^K(\mathbf{x}^H)_i$

Figure 2: (Left) HGCN architecture. The number of node features define the number of input nodes while the embedding dimension determines the number of nodes in the hidden layers. (Right) The aggregation (Eq. 4) is performed upon projection to the tangent space.

In the radial direction, N nodes are distributed according to the following hyperbolic function, which characterizes the hidden metric space:

$$f(r) = \frac{\alpha \sinh \alpha r}{\cosh \alpha R - 1}, \qquad 0 \le r \le R, \qquad \alpha > 0, \tag{1}$$

Therefore, an increasing α would lead to a high node concentration near the rim at $r = R$. On the other hand, their angular distribution is uniform for $\theta \in (-\pi, \pi]$. The existence of edges are also determined probabilistically through the following function: $p((i,j) \in \varepsilon) = \left[\exp\left(\eta\left(d(i,j) - r_0\right)\right)\right]^{-1}$. For $\eta \gg 1$, this effectively means only nodes separated by $d(i,j) \le r_0$ are connected to each other. Distance function can be defined in various ways, but here we chose to use Euclidean and Poincare distance functions (see Eq. 2).

We also used the coordinate of each node as our input features, which are encoded in n_r, n_θ uniform bins in radial and angular direction respectively. We therefore lose some resolutions regarding the nodes' position, and test if the structures in our data for LP task can be recovered during training.

Grid Graphs (GG). We also trained our network to perform LP task in grid graphs in 2D, in which each node is located on a square lattice. The position of each node is also used as our input features.

3 Methods: Graph Embedding in the Hyperbolic Space

Poincare Ball

The Poincare ball model is a hyperbolic manifold which inhabits a sphere of radius $r = |K|$, where $-1/|K|$ denotes the curvature of the hyperbolic space. The distance between two points in the Poincare Ball, is given by the following expression:

$$d_{\mathcal{L}}^{K_\ell}(\mathbf{x}, \mathbf{y}) = \operatorname{arcosh}\left(1 + \frac{2\|\mathbf{x} - \mathbf{y}\|^2 K}{(K - \|\mathbf{x}\|^2)(K - \|\mathbf{y}\|^2)}\right), \tag{2}$$

where $\|\cdot\|$ is the Euclidean norm, while $\mathbf{x}$ and $\mathbf{y}$ are vectors on D-dimensional space where Poincare ball manifold is defined.

Hyperbolic Graph Convolution Network

Our HGCN implementation was based on the work by Chami, et al [4]. The main operations which define the forward propagation for each layer in this architecture are as follows:

$$\mathbf{h}_i^{\ell,H} = \left(W^\ell \otimes^{K_{\ell-1}} \mathbf{x}_i^{\ell-1,H}\right) \oplus^{K_{\ell-1}} \mathbf{b}^\ell \tag{3}$$

$$\mathbf{y}_i^{\ell,H} = \mathrm{AGG}^{K_{\ell-1}}\left(\mathbf{h}^{\ell,H}\right)_i \tag{4}$$

$$\mathbf{x}_i^{\ell,H} = \sigma^{\otimes^{K_{\ell-1},K_\ell}}\left(\mathbf{y}_i^{\ell,H}\right). \tag{5}$$

Here, $\otimes$ and $\oplus$ denote the matrix multiplication and vector addition in Poincare ball respectively, while $K_{\ell,H}$ quantifies the ℓ-th layer's curvature of the Poincare ball which is trainable. The concept of convolution comes in through the attention based neighborhood aggregation function AGG, which operates on the tangent (Euclidean) plane of each node. The AGG function incorporates the information of the nodes' nearest neighbors. It follows that for L-layer architecture, our embedding of a particular node is influenced by the information of nodes which are separated L edges away. We also used ReLU as our activation function in the hidden layers.

We applied Fermi-Dirac decoder on our outputs to compute $p(i,j)$, which quantifies the probability for an edge to connect the node pair (i,j). The expression for the decoder is as follows:

$$p\left((i,j)\in\mathcal{E}|\mathbf{x}_i^{L,H},\mathbf{x}_j^{L,H}\right) = \left[\exp\left(\left(d_{\mathcal{L}}^{K_L}\left(\mathbf{x}_i^{L,H},\mathbf{x}_j^{L,H}\right)^2 - r\right)/t\right)+1\right]^{-1}$$

where $d_{\mathcal{L}}^{K_L}$ is the distance function in the Poincare manifold. Therefore, the hyperparameter r determines the threshold distance for the network to predict the existence of edge between node pairs. The network is trained using the binary cross-entropy loss function when LP is concerned. For NC, we carry out a multinomial regression in the tangent space on the output from the neural network.

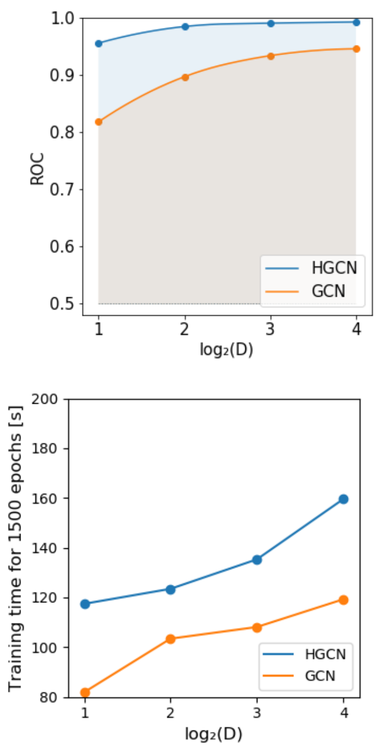
4 Experiments, Results and Discussions

Our experiment was performed using a HGCN with 2 layers, which effectively lead to a convolution up to the second-nearest neighbor. A similar GCN architecture was also used as our baseline model, and we compared their performances on LP tasks. The number of input nodes in our architecture is defined by the node features. Our method discretize the features values into 20 bins in both radial and angular directions ($n_r = n_\theta = 20$), so that we have $D^{[0]} = 40$ input nodes for experiment on hyperbolic random graphs. This is subsequently projected into a low dimensional manifold of dimension $D < D^{[0]}$. Note that D also determines the number of hidden nodes.

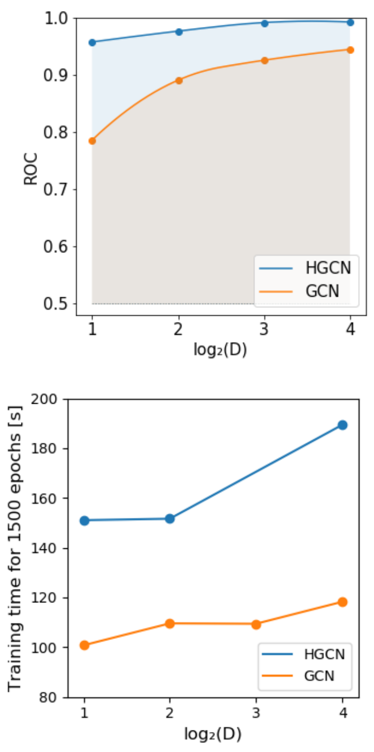
For LP tasks, our training/dev/test data consisted of the pairs of nodes which are connected by the edges (”1”’s). We also included the same number of node pairs which are not connected (”0”’s) by edges for each data set. Therefore, indiscriminate predictions of all ”0”’s and ”1”’s would yield a ROC value of 0.5. Our training/dev/test data split was given by 85/5/10. We tried to compare the performance for $D = 2$, 4, 8, 16 and particularly explored the variation in performance with respect to the learning and dropout rates which highly affects the resulting LP performance. The learning rates are varied between 10^{-3} to 10^{-1}, while the dropout rates are changed from 0 to 1. The set of hyperparameters which yield the highest ROC on our validation data were subsequently chosen to make prediction on our test dataset. Furthermore, in a typical training of both GCN/HGCN, we limited our training epochs to be 1500.

Link Prediction Tasks

We decided to focus on the LP task as intuitively we can expect the performance of HGCN and GCN to be clearly distinguished on graphs which exhibit tree-likeness. The underlying reasons are as follows: (1) There is a qualitative difference between the capacity of Poincare ball and that of Euclidean space, i.e. the former (latter) grows exponentially (polynomially), and (2) link prediction performance depends critically on the amount of distortion the embedding task introduces on the graphs. Hyperbolic space is known to have a greater expressive power to faithfully encode the tree-like structures [9, 8], thus HGCN potentially shows a decisive advantage over GCN on LP.



(a) HRG, distance function in Poincare ball



(b) HRG, distance function in Euclidean space.

Figure 3: HGCN and GCN performance on validation data measured with ROC value for different dimensions in the hidden layers, D (top figure). The solid line shows the optimal performance for HGCN (blue) and GCN (orange), while the shading shows the resulting performance for a range of hyperparameters that we explored. Training times for HGCN and GCN for different dimensions.

		HRG with Poincaré distance	HRG with Euclidean distance	GG
	D	ROC		
HGCN	2	0.956	0.959	0.952
	4	0.987	0.980	0.957
	8	0.991	0.992	0.955
	16	0.992	0.993	0.960
GCN	2	0.818	0.785	0.949
	4	0.881	0.911	0.952
	8	0.921	0.926	0.954
	16	0.946	0.945	0.927

Table 1: Results on test dataset using the hyper parameters that corresponds to the highest ROC value on dev set. For HRG with Poincaré distance, i.e. distance defined in (3), we used $n = 1000$, $\alpha = 10$, $R = 0.95$, $r_0 = 1$, and $\eta = 100$. To generate HRG with Euclidean distance we used $n = 1000$, $\alpha = 0.1$, $R = 0.95$, $r_0 = 0.2$, and $\eta = 100$. For the GG we used 50x50 nodes.

Our experiments confirmed our intuition that the HGCN is indeed superior to GCN on LP on HRG, which was constructed with a hidden hyperbolic metric space as reflected in its node distributions $f(r)$. Our results are also consistent with the expected better performance for increasing D. Our result complemented and corroborated the findings of previous works, demonstrating that HGCN is better on some real networks with low Gromov-δ, i.e. more tree-like [4]. The importance of the hidden hyperbolic metric space should be highlighted here by the robustness of the HGCN/GCN performance to the distance function used in defining the edges in our synthetic graphs (see Fig. 3a and 3b). This is an exciting result as the hidden hyperbolic metric space also underlies many real world data. Moreover, it is possible to map the synthetic constructions to such data through analysis of some of the properties, such as the scaling power, the degrees of clustering, or the average numbers of edges [11].

One possible scenario in which the HGCN potentially offers an advantage is as follows. Imagine several regions of a graph have high concentration of nodes which are highly interconnected within the cluster, yet a significant portion of node pairs is not connected to each other. This might pose a problem for embedding in low-dimensional Euclidean space as there is a lack of space to locate these nodes while preserving their respective distance. This would result in poor LP performance as these nodes might be packed in so small a space, so that there is a high rate of false predictions of edges. Such a clustering scenario potentially leads to a lower GCN performance, and it might be relevant in our synthetic graphs.

We also observed a significant variation in the performance of both HGCN and GCN with respect to the choice of learning and dropout rates. The test performances after hyperparameter tuning are reported in Table 1. There are few extreme cases in which either HGCN or GCN failed to learn the graph structures. HGCN performance was also more sensitive compared to that of GCN with respect to the choice of hyper parameters. There are also some drawbacks for the use of current HGCN technique. Firstly, its training step is slower than GCN, which is not surprising due to the computations in the hyperbolic space, or some mapping of vectors into spaces of different curvatures. However, this disadvantage might be offset since GCN needs a larger embedding dimension D to attain a similar level of accuracy for which the training time is comparable. A more careful study of accuracy and training time is needed to understand how both quantities scale according to D. Another drawback has to do with the instability of the Poincare disk, which is recurrent in our training as distance goes to infinity near the rim of the the disk.

Node Classification Tasks

Our earlier experiment performed NC tasks on connected Erdős-Renyí and tree-like random graphs whose nodes can be classified into C-classes. However, it could be demonstrated that HGCN did not have any advantage over GCN in such scenarios, and the performance of both networks were only limited by the noise we introduced in the node clustering. The results stem from the fact that optimal NC performance on such graphs did not depend on their faithful representation, and it was possible to cramp all the nodes which belong to a certain class to a localized region in the embedding space without compromising on the NC performance. We do not present any further of our NC results due to space limitation for our report.

5 Future Directions and Potential Applications

Our future theoretical work aims to further explore relevant metrics which can be used to approximate some properties of real networks, and classify the cases on which HGCN or GCN can be expected to work better. In this report, we showed some evidence that hidden metric space of graphs significantly influences the performance of HGCN and GCN. A more careful characterization of such metric space, and its relation to Gromov-δ remains to be understood. We also expect to extend our scheme to spherical spaces or product spaces, and identify some cases on which it might be advantageous for prediction tasks. In terms of applications, our findings potentially impact some areas which are classically linked to LP tasks, such as drug development, recommendation systems and analysis of disease spreading networks [1, 2].

6 Conclusions

To sum up, we have demonstrated a class of synthetic graphs on which the Hyperbolic Graph Convolutional Network (HGCN) is superior to its Euclidean counterpart in performing link prediction tasks. In particular, we showed some evidence for the importance of the hidden hyperbolic metric space of our synthetic graphs to achieve the decisive advantage of HGCN. Our synthetic graph construction opens up an avenue for characterization of real world data and how their properties influence the performance of HGCN.

Acknowledgements

We would like to thank our mentors Fred Sala, Ines Chami and Adva Wolf for their invaluable guidance for the completion of the project. A special thanks to Ines Chami for providing us the codes to develop this project.

Contributions

Both authors worked closely together and contributed equally to this work. We have been working on the code development, exploring the whole pipelines of HGCN toolboxes, analyzing results and write this report. Each of these tasks has been done by both authors. Our codes are available HERE.

References

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *arXiv preprint arXiv:1901.00596*, 2019.

[2] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun, "Graph neural networks: A review of methods and applications," *arXiv preprint arXiv:1812.08434*, 2018.

[3] Q. Liu, M. Nickel, and D. Kiela, "Hyperbolic graph neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 8228–8239.

[4] I. Chami, Z. Ying, C. Ré, and J. Leskovec, "Hyperbolic graph convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 4869–4880.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[6] A. B. Adcock, B. D. Sullivan, and M. W. Mahoney, "Tree-like structure in large social and information networks," in *2013 IEEE 13th International Conference on Data Mining.* IEEE, 2013, pp. 1–10.

[7] A. Clauset, C. Moore, and M. E. Newman, "Hierarchical structure and the prediction of missing links in networks," *Nature*, vol. 453, no. 7191, p. 98, 2008.

[8] C. De Sa, A. Gu, C. Ré, and F. Sala, "Representation tradeoffs for hyperbolic embeddings," *Proceedings of machine learning research*, vol. 80, p. 4460, 2018.

[9] R. Sarkar, "Low distortion delaunay embedding of trees in hyperbolic plane," in *International Symposium on Graph Drawing.* Springer, 2011, pp. 355–366.

[10] F. Papadopoulos, D. Krioukov, M. Boguñá, and A. Vahdat, "Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces," in *2010 Proceedings IEEE INFOCOM.* IEEE, 2010, pp. 1–9.

[11] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguná, "Hyperbolic geometry of complex networks," *Physical Review E*, vol. 82, no. 3, p. 036106, 2010.

[12] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.