Embedding Heterogeneous Networks into Hyperbolic Space Without Meta-path

Lili Wang,[1] **Chongyang Gao,** [1] **Chenghan Huang,** [2] **Ruibo Liu,** [1] **Weicheng Ma,** [1]
Soroush Vosoughi [1]

[1] Department of Computer Science, Dartmouth College
[2] Millennium Management LLC
lili.wang.gr@dartmouth.edu, soroush.vosoughi@dartmouth.edu

Abstract

Networks found in the real-world are numerous and varied. A common type of network is the heterogeneous network, where the nodes (and edges) can be of different types. Accordingly, there have been efforts at learning representations of these heterogeneous networks in low-dimensional space. However, most of the existing heterogeneous network embedding methods suffer from the following two drawbacks: (1) The target space is usually Euclidean. Conversely, many recent works have shown that complex networks may have hyperbolic latent anatomy, which is non-Euclidean. (2) These methods usually rely on meta-paths, which require domain-specific prior knowledge for meta-path selection. Additionally, different down-streaming tasks on the same network might require different meta-paths in order to generate task-specific embeddings. In this paper, we propose a novel self-guided random walk method that does not require meta-path for embedding heterogeneous networks into hyperbolic space. We conduct thorough experiments for the tasks of network reconstruction and link prediction on two public datasets, showing that our model outperforms a variety of well-known baselines across all tasks.

Introduction

There has been an ever-increasing amount of network data available to researchers in recent decades. Through encoding network structure into low-dimensional embeddings, researchers can mine and model networks without resorting to feature engineering.

In the field of natural language processing, there has been a long history of generating low dimensional embeddings for words based on co-occurrence in the same contexts, which is similar to the local proximity in a graph. Using language models to embed networks was first proposed by Perozzi et al. in their work DeepWalk (Perozzi, Al-Rfou, and Skiena 2014). The random walks are adopted to generate the context of nodes from the network, which are then fed into a *Skip-Gram* model (Mikolov et al. 2013) as sentences. This way, similar nodes in a network will have similar contexts and thus have related embeddings that are near one another. Grover et al. proposed an extension of DeepWalk called node2vec (Grover and Leskovec 2016) by smoothly interpolating breadth-first and depth-first sampling, thus combining different views of node equivalence.

After these two pioneering papers, there has been an explosion of work focused on representation learning for different types of networks. One way to divide the prior work is to separate embedding methods for homogeneous and heterogeneous networks. Homogeneous networks contain only one type of node. Heterogeneous networks, on the other hand, have two or more types of nodes. For example, citation networks where all nodes are papers and all edges correspond to citations are homogeneous. However, if we extend the network to include authors and venues as nodes, then it becomes a heterogeneous network. In this network, the edges represent different kinds of relationships. For instance, author-to-author edges would represent collaboration, while author-to-paper edges would represent authorship and so on.

Various methods for embedding heterogeneous networks have been proposed (see for instance, Dong, Chawla, and Swami 2017; Fu, Lee, and Lei 2017; Hussein, Yang, and Cudré-Mauroux 2018; He et al. 2019; Shi et al. 2018a; Tang, Qu, and Mei 2015; Shi et al. 2018b; Wang, Zhang, and Shi 2019). However, to the best of our knowledge, all the proposed methods, with the exception of HHNE (Wang, Zhang, and Shi 2019), embed heterogeneous networks into Euclidean space. In this paper, we introduce a novel embedding method for heterogeneous networks in hyperbolic space.

Hyperbolic and Euclidean spaces can both preserve certain properties, such as angles between two vectors when the conformal transformation is used. However, there are two main advantages of hyperbolic space compared with Euclidean space. First, hyperbolic space has a stronger representation ability for hierarchical structures; for instance, Sarkar (2011) show that trees can be embedded into the two-dimensional hyperbolic space (Poincaré disk) with arbitrarily low distortion. On the other hand, Linial et al. (1995) show that Euclidean space cannot represent a tree with low distortion. A sizeable number of networks have a hierarchical structure, which makes them suitable for embedding into hyperbolic space. Another advantage of hyperbolic space is that it represents high-degree vertices better than Euclidean space, which is important when dealing with complex networks where the degree of the vertices follow a power-law

distribution (i.e., where there are hubs with many large degrees).

Having established the advantages of hyperbolic space for network embedding, we propose a novel method for embedding heterogeneous networks into hyperbolic space. As mentioned earlier, as far as we are aware the only other method proposed for embedding heterogeneous networks into hyperbolic space is HHNE (Wang, Zhang, and Shi 2019). However, HHNE suffers from the following disadvantages:

- It requires meta-paths to guide random walk. Existing literature shows that the selection of meta-paths highly affects the quality of the learnt embeddings (Hussein, Yang, and Cudré-Mauroux 2018), and the selection of meta-paths needs domain-specific prior knowledge. This makes HHNE not easily generalizable.
- Even for the same network, HHNE requires different types of meta-path for training models for down-stream tasks of different types of edges (e.g. link prediction for paper-author and paper-venue edges). We believe it is preferable to have one general and consistent embedding for all down-stream tasks, reducing the need for training models multiple times.
- HHNE embeds networks to a *Poincaré ball*. But to the best of our knowledge, the gradient on the *Poincaré ball* cannot be calculated precisely and needs a first-order approximation to the exponential map, called a *retraction* (Bonnabel 2013).

In this paper, we propose a novel self-guided random walk method that solves the issues listed above. Our method does not require meta-paths and uses very few non-sensitive parameters across all the domains and down-streaming tasks. We use the *hyperboloid* model to embed networks in order to avoid the approximation of *retraction*. We conduct thorough experiments for the tasks of network reconstruction and link prediction on two public datasets, showing that our model can achieve superior performance across all tasks.

Related Work

Homogeneous Network Embedding

Besides random walk based methods DeepWalk and node2vec, LINE (Tang et al. 2015) was proposed for large-scale network embedding, and can preserve first and second-order proximities. Cao et al. (2015), extended LINE by creating GraRep, which preserves k-step proximities when embedding networks. Their model captures the different k-step relational information with different values of k amongst vertices from the graph directly by manipulating different global transition matrices defined over the graph, without involving slow and complex sampling processes. SDNE (Wang, Cui, and Zhu 2016) is also an extension of LINE which uses a semi-supervised deep autoencoder model to capture both first and second-order proximities in networks. HOPE (Ou et al. 2016) learns vertex representations that capture the asymmetric high-order proximity in directed networks by solving a matrix factorization problem, while APP (Zhou et al. 2017) is another network embedding method designed to capture asymmetric proximity by using a *Monte Carlo* approach to approximate the asymmetric *rooted PageRank* proximity (Song et al. 2009). In the area of hyperbolic embedding, Nickel et al. (2017) proposed a method (PoincaréEmb) of learning hierarchical representations of symbolic data; McDonald et al. (2019) proposed a "teleport" walk to embed attributed graphs and Ganea et al. (2018) use hyperbolic cones as a heuristic for embedding directed acyclic graphs. Our prior work (Wang et al. 2020) also presents a method for embedding structural role similarity of nodes of homogeneous networks into hyperbolic space.

Heterogeneous Network Embedding

To deal with heterogeneous networks, metapath2vec (Dong, Chawla, and Swami 2017) extends DeepWalk and node2vec by a meta-path guided random walk. HIN2Vec proposes a multi-task learning method based on different relation types and network structures. JUST (Hussein, Yang, and Cudré-Mauroux 2018) provides a Jump/Stay strategy of random walk to balance the heterogeneous edges and homogeneous edges. HeteSpaceyWalk (He et al. 2019) first formalizes the meta-path guided random walk to a high-order Markov chain process, and then utilizes an efficient random walk which they call heterogeneous personalized spacey random walk. AspEm (Shi et al. 2018a) proposes the concept of aspects in networks, and preserves semantic information in heterogeneous information networks based on multiple aspects. PTE (Tang, Qu, and Mei 2015) decomposes a network to multiple bipartite networks by the type of edges to learn representations from the one-hop neighbourhood, and HEER (Shi et al. 2018b) extends it by considering typed closeness. In the area of hyperbolic embedding, there is only one heterogeneous embedding method, HHNE (Wang, Zhang, and Shi 2019), which extends metapath2vec by using meta-path guided random walk to embed nodes into a *Poincaré ball*. With the development of Graph Neural Networks (GNN), many method like R-GCN (Schlichtkrull et al. 2018), HGT (Hu et al. 2020), HAN (Wang et al. 2019) and HetGNN (Zhang et al. 2019) utilize GNNs for heterogeneous networks. Finally, knowledge bases are a special kind of heterogeneous network. Many relation-learning based methods for knowledge bases like TransE (Bordes et al. 2013), TransH (Wang et al. 2014), RotatE (Sun et al. 2019), DistMult (Yang et al. 2014), NKGE (Wang et al. 2018), and SACN (Shang et al. 2019) have been proposed.

Framework

Preliminaries

A heterogeneous network is defined as a graph $G = (V, E, T)$, in which each node v and each edge e are associated with their mapping functions $\phi(v) : V \rightarrow T_V$ and $\varphi(e) : E \rightarrow T_E$ respectively. T_V and T_E denote the sets of object and relation types, where $|T_V| + |T_E| > 2$. The task is to learn a d-dimensional hyperbolic embedding $\mathrm{X} \in \mathbb{H}^{|V| \times d}, d \ll |V|$ in a unsupervised way.

Self-guided Random Walk

Before we propose the self-guided random walk, we first revisit the meta-path guided random walk. Meta-paths are used in heterogeneous graph embeddings since random walks on such graphs are biased to highly visible types of nodes. This means that sequences generated by random walk have a skewed distribution of nodes, with highly visible node types being over-represented. Node embeddings learned from these sequences will also be biased towards the highly visible domains. Thus, meta-paths are used to guide random walks to overcome this problem (Hussein, Yang, and Cudré-Mauroux 2018).

In this section, we propose a very simple but useful self-guided random walk method, which can adaptively change the probability of each node type in the next walk step and automatically balance the distribution of domains in the context.

A self-guided walk on G is a sequence of vertices $\langle v_1, v_2, \cdots, v_k \rangle$ such that $\langle v_i, v_{i+1} \rangle \in E$ and node type $\phi(v_i)$ appears $\mathcal{N}_{\phi(v_i)}$ times in the current sequence, for $1 \leq i < k$. Instead of walking to all the neighbor nodes with equal probability like standard random walks, we define a modified probability for type ϕ nodes:

$$\frac{e^{-\mathcal{N}_{\phi}}}{\sum\limits_{\phi' \in T_V} e^{-\mathcal{N}_{\phi'}}}, \tag{1}$$

Assume we have $\langle v_1, v_2, \cdots, v_k \rangle$, after fraction reduction, the transition probability at step k+1 can be normalized to:

$$p\left(v_{k+1} | \langle v_1, v_2, \cdots, v_k \rangle\right) = \frac{e^{\frac{-\mathcal{N}_{\phi_{v_{k+1}}}}{|\{\phi_{v_{k+1}} = \phi_{v_i} | (v_k, v_i) \in E\}|}}}{\sum\limits_{(v_k, v_j) \in E} e^{\frac{-\mathcal{N}_{\phi_{v_j}}}{|\{\phi_{v_j} = \phi_{v_i} | (v_k, v_i) \in E\}|}}}, \tag{2}$$

In this way, all the types of nodes will be balanced in the context. The prior work closest to this idea is JUST (Hussein, Yang, and Cudré-Mauroux 2018). They proposed a complex Jump & Stay to balance the heterogeneous edges (connecting different types of nodes) and homogeneous edges (connecting the same type of nodes). However, our method focuses on achieving a balanced distribution of each type of node, which means that homo/heterogeneous edges will be automatically balanced.

In the next section, we first briefly introduce the *hyperboloid* model and the gradient on it, then we explain how to train embedding from the results of the self-guided random walk.

Hyperbolic Embedding Learning

Hyperbolic space has a negative curvature. There are multiple models that can be used to represent hyperbolic space, each having different advantages. The *Poincaré ball* model is the best model for low dimensional visualizations of the embeddings. The *Klein* model is often used for the calculation of Einstein midpoints because of its computational efficiency. The *hyperboloid* model gives a closed-form of the gradient descent formula. Therefore, in this paper, we use the *hyperboloid* model for the calculation of gradient. Below we go over several definitions needed to define the *hyperboloid* model and the gradient on it (Wilson and Leimeister 2018).

Definition 1. The Minkowski inner product is defined as

$$\langle x, y \rangle_{n:1} = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n - x_{n+1} y_{n+1}. \tag{3}$$

The Minkowski inner product is not a traditional inner product since it's not positive definite.

Definition 2. The *Hyperboloid* model is given by

$$\mathbb{H}^n = \left\{ x \in \mathbb{R}^{n:1} \mid \langle x, x \rangle_{n:1} = -1,\, x_{n+1} > 0 \right\}. \tag{4}$$

$\mathbb{R}^{n:1}$ stands for the ambient Minkowski space. A simple *hyperboloid* model is the hyperboloid of revolution: $\mathbb{H}^2 = \left\{ x \in \mathbb{R}^3 \,|\, x_1^2 + x_2^2 - x_3^2 = -1,\, x_3 > 0 \right\}$.

Definition 3. The hyperbolic distance between points x and y is defined as

$$d_{\mathrm{H}^n}(x, y) = \cosh^{-1}\left(- \langle x, y \rangle_{n:1} \right). \tag{5}$$

We can calculate the gradient of distance function in $\mathbb{R}^{n:1}$:

$$\begin{aligned} \nabla_x^{\mathbb{R}^{n:1}} d(x, y) &= \nabla_x^{\mathbb{R}^{n:1}} \cosh^{-1}\left(- \langle x, y \rangle_{n:1} \right) \\ &= -\frac{1}{\sqrt{\langle x, y \rangle_{n:1}^2 - 1}} \nabla_x^{\mathbb{R}^{n:1}} \langle x, y \rangle_{n:1} \\ &= -\frac{y}{\sqrt{\langle x, y \rangle_{n:1}^2 - 1}}. \end{aligned}$$

Definition 4. The tangent space $T_x \mathbb{H}^n$ to point $x \in \mathbb{H}^n$ is the set of points satisfying

$$T_x \mathbb{H}^n = \left\{ u \in \mathbb{R}^{n:1} \mid \langle u, x \rangle_{n:1} = 0 \right\}. \tag{6}$$

The projection from ambient space $\mathbb{R}^{n:1}$ to tangent space $T_x \mathbb{H}^n$ is

$$\Pi_x(u) = u + \langle u, x \rangle_{n:1} x. \tag{7}$$

Definition 5. Exponential map is a common way to map a vector from tangent space to the *hyperboloid* manifold. Considering $u \in T_x \mathbb{H}^n$, it is exponential mapped as

$$\cosh(\sqrt{\langle u, u \rangle_{n:1}})x + \sinh(\sqrt{\langle u, u \rangle_{n:1}}) \frac{u}{\sqrt{\langle u, u \rangle_{n:1}}}. \tag{8}$$

In this way, when we calculate the gradient of a given function f, first we can calculate its gradient in the ambient space $\nabla_x^{\mathbb{R}^{n:1}} f$. In the following steps, we need to project it to the tangent space using formula 7 and apply exponential map on the vector we get using formula 8.

Once we are able to perform hyperbolic gradient descent, the next step is to design the loss function, which is the evaluation of an embedding. Following HEAT (McDonald and He 2019), after generating the self-guided random walk sequences, a sliding window is used to examine all the sequences and add pairs of nodes that appear within the window to a multi-set $\mathcal{P}$ to serve as all the positive sample pairs for training. Intuitively, the further the distance between two nodes, the smaller the probability that there is a connection

Dataset	DBLP											
Edge	P-A						P-V					
Dimension	2	5	10	15	20	25	2	5	10	15	20	25
Deepwalk	0.6933	0.8034	0.9324	0.9666	0.9722	0.9794	0.7324	0.7906	0.8813	0.9353	0.9505	0.9558
LINE(1st)	0.5286	0.5397	0.6740	0.7220	0.7457	0.7668	0.5182	0.5500	0.7070	0.7295	0.7369	0.7436
LINE(2nd)	0.6740	0.7379	0.7541	0.7868	0.7600	0.7621	0.6242	0.6349	0.6333	0.6343	0.6444	0.6440
node2vec	0.7107	0.8162	0.9418	0.9719	0.9809	0.9881	0.7595	0.8019	0.8922	0.9382	0.9524	0.9596
metapath2vec	0.6686	0.8261	0.9202	0.9500	0.9623	0.9690	0.7286	0.9072	0.9691	0.9840	0.9879	0.9899
PoincaréEmb	0.8251	0.8769	0.8921	0.8989	0.9024	0.9034	0.5718	0.5529	0.6271	0.6446	0.6600	0.6760
HHNE	**0.9835**	0.9838	0.9887	0.9898	0.9913	0.9930	0.8449	0.9984	0.9985	0.9985	0.9985	0.9985
JUST	0.7373	0.8682	0.9416	0.9664	0.9793	0.9889	0.6553	0.8007	0.8828	0.9265	0.9545	0.9567
PTE	0.5857	0.6702	0.7266	0.7483	0.7605	0.7590	0.6487	0.6758	0.6817	0.6905	0.7024	0.7135
Hin2Vec	0.7095	0.8465	0.9303	0.9561	0.9758	0.9748	0.6983	0.8435	0.9574	0.9760	0.9715	0.9802
HeteSpaceyWalk	0.6487	0.8128	0.9113	0.9459	0.9569	0.9543	0.7090	0.8826	0.9404	0.9725	0.9807	0.9835
Our method	0.9602	**0.9893**	**0.9972**	**0.9982**	**0.9985**	**0.9986**	**0.9728**	**0.9995**	**0.9999**	**0.9999**	**0.9999**	**0.9999**

Table 1: AUC scores for the network reconstruction task on the DBLP network.

between them. Here we adopt that the probability should be proportional to $e^{-d^2_{\mathbb{H}^n}(\mathbf{e}_u,\mathbf{e}_v)}$, where $\mathbf{e}_u$ and $\mathbf{e}_v$ are the embeddings of two nodes u and v. We can also easily deal with large networks using negative sampling. In this way, our loss function $\mathcal{L}$ is given by:

$$\mathcal{L} = -\frac{1}{|\mathcal{P}|}\sum_{(u,v)\in\mathcal{P}} \log \frac{e^{-d^2_{\mathbb{H}^n}(\mathbf{e}_u,\mathbf{e}_v)}}{\sum\limits_{v'\in\mathcal{M}(u)} e^{-d^2_{\mathbb{H}^n}(\mathbf{e}_u,\mathbf{e}_{v'})}}. \quad (9)$$

In formula 9, for a given v, $\mathcal{M}(u) := \{v\}\cup\{w|(u,w)\notin\mathcal{P}\}$ contains two parts. One is v, the other is the negative sampling set. The probability for a node w to be chosen into the negative sample set $\mathcal{M}(u)$ is proportional to the frequency of that node w in the full sample set $\mathcal{P}$. Since we already have the gradient of distance function in $\mathbb{R}^{n:1}$ and the methodology to project it to the *hyperboloid*, the gradient of the loss function can be easily calculated by chain rule.

Experiments

In this section, we evaluate the performance of our proposed method both qualitatively through visualization, and quantitatively on the tasks of network reconstruction and link prediction. We compare our method with several baselines and explore the sensitivity of our method to the choice of parameters. For a fair comparison, we use the exact same datasets, experiments, settings, and metrics as HHNE.

Datasets

The following datasets were given to us by the authors' of HHNE:

- **DBLP**: This is a subset network of DBLP, which contains three node types: 14,475 authors (A), 14,376 papers (P), and 20 venues (V). The network also contains the following edge types: 41,794 *paper authorship* relations (P-A) and 14,376 *publish* relations (P-V).
- **MovieLens**: This is a subset network of MovieLens, which contains three node types: 11,718 actors (A), 9,160 movies (M), and 3,510 directors (D). The network also contains the following edge types: 64,051 *act in* relations (M-A) and 9,160 *direct* relations (M-D).

Experiment Settings

We compared our method with several homogeneous and heterogeneous network embedding methods. The homogeneous methods include DeepWalk, LINE, node2vec, and the hyperbolic embedding method, PoincaréEmb. The heterogeneous methods include metapath2vec, JUST, PTE, Hin2Vec, HeteSpaceyWalk, and the hyperbolic embedding method HHNE. Note that many recent works on Graph Neural Networks focusing on designing a learning mechanism on graphs, in most cases they are end-to-end, supervised or semi-supervised. Thus we do not include them in baselines as we are interested in comparing the performance of our method against other network embedding methods that are unsupervised and focusing on transforming networks into low-dimensional space.

For methods based on meta-path guided random walks, we use "APA" for the relation "P-A", and "APVPA" for the relation "P-V" in network reconstruction and link prediction experiments on the DBLP dataset. For the experiments on the MovieLens dataset, we use "AMDMA" for both relations "M-A" and "M-D". (Same as Wang et al. (2019)). For PTE, we use the unsupervised setting and construct two bipartite graphs for each dataset: DBLP (A-P, P-V), MovieLens (A-M, M-D).

The performance of DeepWalk, LINE, node2vec, metapath2vec, PoincaréEmb, and HHNE reported here is taken from Wang et al. (2019) (since we used the same datasets and settings). For JUST, PTE, Hin2Vec, and HeteSpaceyWalk, we tried our best to fine-tune the parameters and report the best results. For our method, we use the following parameters: We do 10 random walks from each node in the training set with the length of 80, and use a sliding window of size 5 to generate positive samples. For the *hyperboloid* embedding learning, we generate 20 negative samples for each positive one, and use the learning rate of 0.3 and a batch size of 512 to train 5 epochs. All the experiments are run on an Amazon AWS "p2.8xlarge" instance running a Linux OS with 488GB of RAM, and the random seeds are set to 0 at

Dataset	MoiveLens											
Edge	M-A						M-D					
Dimension	2	5	10	15	20	25	2	5	10	15	20	25
Deepwalk	0.6320	0.6763	0.7610	0.8244	0.8666	0.8963	0.6626	0.7263	0.8246	0.8784	0.9117	0.9345
LINE(1st)	0.5424	0.5675	0.6202	0.6593	0.6925	0.7251	0.5386	0.5839	0.6114	0.6421	0.6748	0.7012
LINE(2nd)	0.6378	0.7047	0.7739	0.7955	0.8065	0.8123	0.6016	0.6521	0.6969	0.7112	0.7503	0.7642
node2vec	0.6402	0.6774	0.7653	0.8304	0.8742	0.9035	0.6707	0.7283	0.8308	0.8867	0.9186	0.9402
metapath2vec	0.6404	0.6578	0.7231	0.7793	0.8189	0.8483	0.6589	0.7230	0.8063	0.8455	0.8656	0.8800
PoincaréEmb	0.5231	0.5317	0.5404	0.5479	0.5522	0.5545	0.6213	0.7266	0.7397	0.7378	0.7423	0.7437
HHNE	0.8832	0.9168	0.9211	0.9221	0.9239	0.9233	**0.9952**	0.9968	0.9975	0.9972	0.9982	0.9992
JUST	0.7108	0.8451	0.9213	0.9514	0.9790	0.9832	0.7267	0.8538	0.9106	0.9510	0.9735	0.9874
PTE	0.5601	0.6078	0.6496	0.6768	0.7023	0.7127	0.6035	0.6540	0.6830	0.7149	0.7388	0.7404
Hin2Vec	0.7345	0.8628	0.9204	0.9571	0.9677	0.9702	0.7440	0.8658	0.9127	0.9438	0.9615	0.9735
HeteSpaceyWalk	0.6556	0.7121	0.7888	0.8331	0.8688	0.8997	0.6830	0.7568	0.8399	0.8817	0.9005	0.9138
Our method	**0.9251**	**0.9690**	**0.9873**	**0.9934**	**0.9959**	**0.9970**	0.9904	**0.9996**	**0.9999**	**0.9999**	**0.9999**	**0.9999**

Table 2: AUC scores for the network reconstruction task on the MovieLens network.

Dataset	DBLP											
Edge	P-A						P-V					
Dimension	2	5	10	15	20	25	2	5	10	15	20	25
Deepwalk	0.5813	0.7370	0.8250	0.8664	0.8807	0.8878	0.7075	0.7197	0.7292	0.7325	0.7522	0.7640
LINE(1st)	0.5090	0.5168	0.5427	0.5631	0.5742	0.5857	0.5160	0.5663	0.5873	0.5896	0.5891	0.5846
LINE(2nd)	0.5909	0.6351	0.6510	0.6582	0.6644	0.6782	0.5121	0.5216	0.5332	0.5425	0.5492	0.5512
node2vec	0.6709	0.7527	0.8469	0.8881	0.9037	0.9102	0.7369	0.7286	0.7481	0.7583	0.7674	0.7758
metapath2vec	0.6536	0.7294	0.8279	0.8606	0.8740	0.8803	0.7059	0.8516	0.9248	0.9414	0.9504	0.9536
PoincaréEmb	0.6742	0.7381	0.7699	0.7743	0.7806	0.7830	0.8257	0.8878	0.9113	0.9142	0.9185	0.9192
HHNE	0.8777	0.9041	0.9111	0.9111	0.9106	0.9117	0.9331	0.9409	0.9619	0.9625	0.9620	0.9612
JUST	0.6577	0.7538	0.7926	0.8041	0.8056	0.8026	0.5847	0.7060	0.7468	0.7543	0.7610	0.7628
PTE	0.5070	0.5369	0.5509	0.5684	0.5869	0.5816	0.5425	0.6209	0.6407	0.6584	0.6693	0.6761
Hin2Vec	0.6412	0.7690	0.8187	0.8269	0.8315	0.8341	0.6544	0.6800	0.7399	0.7230	0.7405	0.7558
HeteSpaceyWalk	0.6337	0.7573	0.8302	0.8792	0.8916	0.9027	0.6931	0.8474	0.9042	0.9201	0.9344	0.9450
Our method	**0.8803**	**0.9129**	**0.9264**	**0.9290**	**0.9294**	**0.9303**	**0.9547**	**0.9624**	**0.9681**	**0.9701**	**0.9742**	**0.9751**

Table 3: AUC scores for the link prediction task on the DBLP network.

Dataset	MoiveLens											
Edge	M-A						M-D					
Dimension	2	5	10	15	20	25	2	5	10	15	20	25
Deepwalk	0.6278	0.6353	0.6680	0.6791	0.6868	0.6890	0.6258	0.6482	0.6976	0.7163	0.7324	0.7446
LINE(1st)	0.5053	0.5636	0.5914	0.6184	0.6202	0.6256	0.5139	0.5496	0.5885	0.6647	0.6742	0.6957
LINE(2nd)	0.5712	0.5874	0.6361	0.6442	0.6596	0.6700	0.6501	0.6607	0.7499	0.7756	0.7982	0.8051
node2vec	0.6349	0.6402	0.6700	0.6814	0.6910	0.6977	0.6299	0.6589	0.7034	0.7241	0.7412	0.7523
metapath2vec	0.6168	0.6212	0.6332	0.6382	0.6453	0.6508	0.6191	0.6332	0.6687	0.6702	0.6746	0.6712
PoincaréEmb	0.5535	0.5779	0.5984	0.5916	0.5988	0.5995	0.5856	0.6290	0.6518	0.6715	0.6821	0.6864
HHNE	0.7715	0.8255	0.8312	0.8319	0.8318	0.8309	0.8520	0.8967	0.8984	0.9007	0.9000	0.9018
JUST	0.6297	0.7199	0.8004	0.7948	0.8057	0.8089	0.6217	0.7562	0.8320	0.8361	0.8394	0.8403
PTE	0.5035	0.5100	0.5428	0.5462	0.5436	0.5542	0.5529	0.6450	0.6426	0.6480	0.6477	0.6529
Hin2Vec	0.6534	0.7486	0.7666	0.7693	0.7614	0.7784	0.6792	0.7645	0.8030	0.8081	0.8002	0.8179
HeteSpaceyWalk	0.6135	0.6247	0.6221	0.6374	0.6429	0.6433	0.6188	0.6296	0.6536	0.6614	0.6709	0.6754
Our method	**0.8498**	**0.8707**	**0.8720**	**0.8723**	**0.8747**	**0.8765**	**0.8781**	**0.9023**	**0.9058**	**0.9083**	**0.9096**	**0.9124**

Table 4: AUC scores for the link prediction task on the MovieLens network.

the beginning.

Network Reconstruction

The network reconstruction task tests the ability of embedding methods to preserve the original network structure. That is, to recover the structure of a network from embeddings learned on the whole network. To test our method, we use all the edges between two node types as the positive set and all the non-edges as the negative set. We use the distances between node embeddings (on the *hyperboloid*) to get an AUC score by thresholding on different distance values for predicting whether a link exists between two nodes.

The results for the DBLP and MoiveLens networks are shown in Table 1 and Table 2, respectively. Our models outperform all the baselines for all embedding dimensions from 2 to 25, with two exceptions (out of 12) where HHNE generates better results when embedding in 2 dimensions. Note that even though HHNE requires task-specific meta-paths, our model outperforms it in most cases without any external knowledge. Our better results may be attributed to the accurate gradient calculation in the *hyperboloid* and the more diverse context generated by the self-guided random walk. HHNE also outperforms other baselines in most cases, suggesting the superiority of hyperbolic heterogeneous network embedding methods in low dimensions. Another hyperbolic embedding method, PoincaréEmb, performs much worse than HHNE and our method. The reason for this may be that this method is designed for homogeneous networks and only preserves the first-order proximity, suggesting that to embed heterogeneous networks into hyperbolic space, either meta-paths or our proposed self-guided random walk should be utilized. Also, other models tend to have better performance than LINE, PTE, and PoincaréEmb, because these three models only capture first or second-order proximities, while others capture longer context information between nodes.

Link Prediction

The link prediction task tests the ability of embedding methods to predict unseen network structure. For each type of relation, we randomly remove 20% of the edges from the network without increasing the number of connected components, the network is then used for learning embeddings. For testing, all the removed edges are used as the positive set and a negative set is created by randomly sampling an equal number of non-edges. The AUC score is calculated in the same manner as for the network reconstruction task.

The results for the DBLP and MoiveLens networks for this task are shown in Table 3 and Table 4, respectively. Overall, the results of link prediction experiments are consistent with the results of network reconstruction experiments. Our models outperform all the baselines, with the greatest difference seen in lower dimensions. As with the prior task, HHNE ranked second across all the methods. LINE, PTE, and PoincaréEmb, which only capture 1-hop or 2-hop neighbourhood of nodes perform worse than other baselines. Link prediction is in general a harder task than network reconstruction, since it involves the prediction of unseen edges. Thus, we see that for each method under the same edge type and dimensions, the AUC score for link prediction is almost always lower than the AUC of network reconstruction.

Visualization

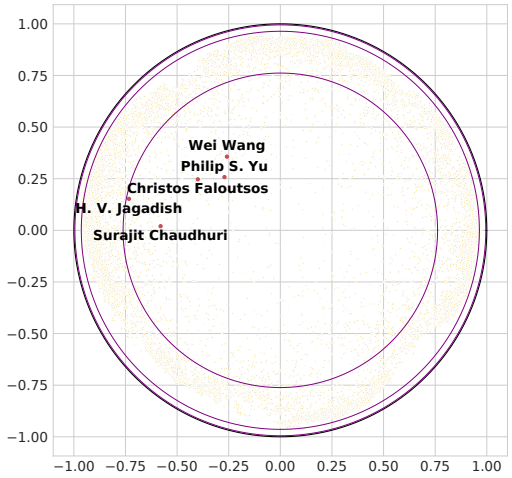

Figure 1: Embedding of each author in DBLP after projected to a 2-dimensional *Poincaré disk*, each dot represent an author

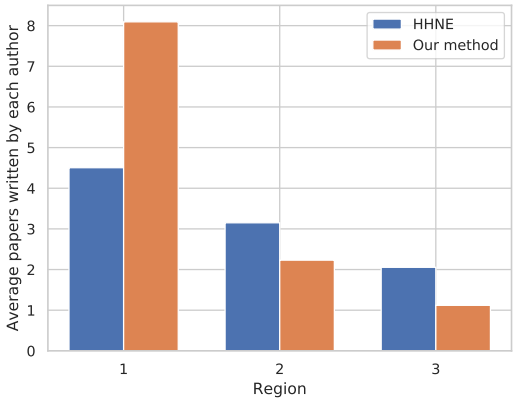

Figure 2: The results of HHNE and our methods for average number of papers written by each author

We repeat the visualization experiment from HHNE to evaluate the latent hierarchy learned by our model. We show our model's embedding of each author in the DBLP network projected to a 2-dimensional *Poincaré disk* in Figure 1. We divide the disk into three equal-radius regions using the distance function on a *Poincaré disk*, shown in Equation 10 below:

$$d_{\mathbb{H}}(u,v) = \cosh^{-1}\left(1 + 2\frac{\|u-v\|^2}{\left(1-\|u\|^2\right)\left(1-\|v\|^2\right)}\right) \tag{10}$$

Region 1: $d_{\mathbb{H}}(u,0) \leq 2$. **Region 2**: $2 \leq d_{\mathbb{H}}(u,0) \leq 4$. **Region 3**: $4 \leq d_{\mathbb{H}}(u,0) \leq 6$

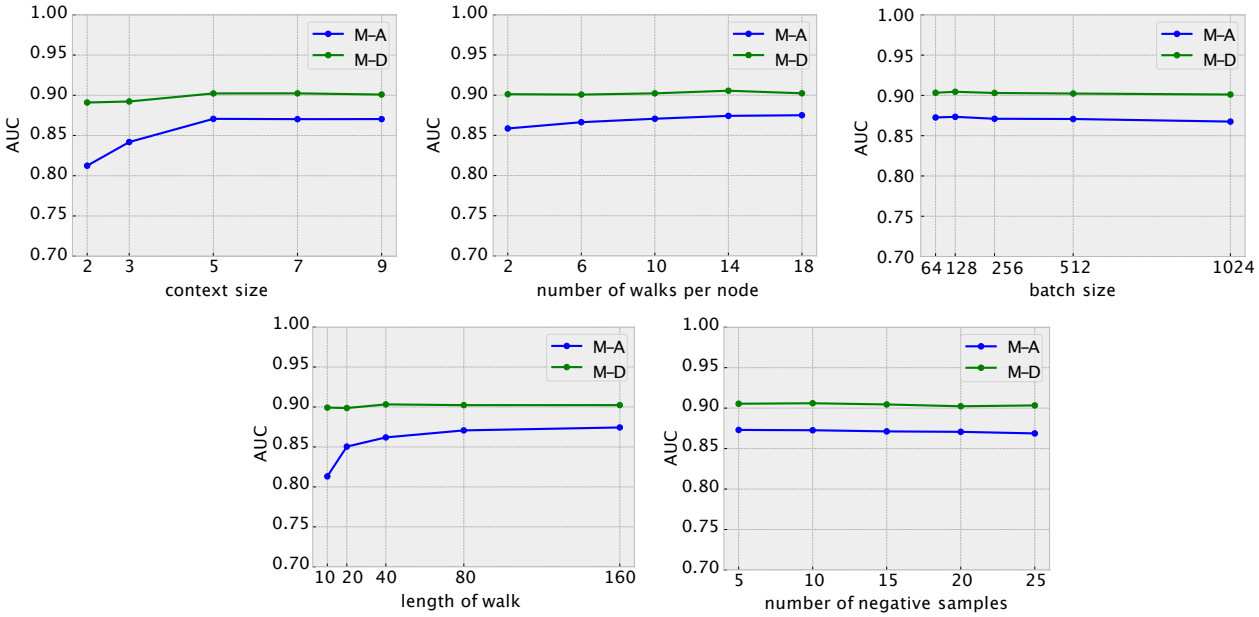

Figure 3: Parameter sensitivity for the link prediction task on the MovieLens dataset.

We mark the three boundaries in Figure 1 using purple circles[1]. The latent hierarchy of authors is based on their degree (i.e., number of publications). The distance between the authors and the origin of the disk reflects the latent hierarchy of the authors captured by our model. The closer an author to the origin, the higher its latent hierarchy. This means that authors in Region 1 have higher latent hierarchy than authors in Region 2 and so on. In Region 1, we can find all the five high-impact authors mentioned by HHNE (labelled in Figure 1). Furthermore, we calculate the average number of papers written by the authors in each of the three regions (1,804, 11,704, and 967 authors respectively) and show that in Figure 2. We also show the same measurement for HHNE from Wang et al. (2019). The average number of papers written by the authors in Region 1 is greater for our model compared to HHNE. Also, as seen in Figure 2, our model better captures the power-law degree distribution inherent in authorship networks. This means that the latent hierarchy learned by our model is better than the one learned by HHNE.

Parameter Sensitivity Analysis

Our model has several hyper-parameters:
batch size - The batch size used for training.
context size - The window size used for collecting the positive samples.
number of walks per node - The number of random walks started at each node.
length of walk - The length of each random walk.
negative samples - The number of negative samples per positive sample.

The parameter sensitivity analysis for our model for the link prediction task on the MovieLens dataset (for M-A and M-D edges) is shown in Fig. 3. We tune each parameter separately while fixing the other parameters to their default values and fix the dimension of embeddings to 5. Overall, our model is not parameter-sensitive. The most sensitive parameters are the context size and the length of the random walk. These two parameters need to be set to at least 5 and 10 (respectively) to get stable results. Generally, a larger context enables a better representation of neighbourhood information, and a longer length of walk enables the self-guided random walk to generate more balanced node sequences for each type of node.

Conclusion

In this paper, we propose a novel hyperbolic embedding method for heterogeneous networks. Our model uses a variant of random walk called self-guided random walk. This novel random walk method removes the need for meta-paths, enabling us to get consistent embeddings that can be used for different down-stream tasks without the need for retraining our model. Lack of reliance on meta-paths also removes the need for external knowledge. Through thorough experimentation using publicly available datasets, we show that our model outperforms a variety of well-known baselines, including the only other hyperbolic heterogeneous network embedding method, HHNE. Future avenues for research could involve the extension of our framework to embed time-evolving heterogeneous networks into hyperbolic space. The code and data for this paper will be made available upon request.

[1]Note that regions look disproportionate because they are visualized on a 2D Euclidean grid. Also note that the hyperbolic space spans exponentially, so even though the embeddings in Region 3 look overlapped, they are not.

Acknowledgements

We thank the authors of HHNE (Wang, Zhang, and Shi 2019) for sharing their implementation/datasets and answering our questions about their paper. This research was supported in part by the Dartmouth Burke Research Initiation Award.

References

Bonnabel, S. 2013. Stochastic gradient descent on Riemannian manifolds. *IEEE Transactions on Automatic Control* 58(9): 2217–2229.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, 2787–2795.

Cao, S.; Lu, W.; and Xu, Q. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, 891–900. ACM.

Dong, Y.; Chawla, N. V.; and Swami, A. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 135–144.

Fu, T.-y.; Lee, W.-C.; and Lei, Z. 2017. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 1797–1806.

Ganea, O.-E.; Bécigneul, G.; and Hofmann, T. 2018. Hyperbolic entailment cones for learning hierarchical embeddings. *arXiv preprint arXiv:1804.01882* .

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 855–864. ACM.

He, Y.; Song, Y.; Li, J.; Ji, C.; Peng, J.; and Peng, H. 2019. HeteSpaceyWalk: a heterogeneous spacey random walk for heterogeneous information network embedding. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 639–648.

Hu, Z.; Dong, Y.; Wang, K.; and Sun, Y. 2020. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, 2704–2710.

Hussein, R.; Yang, D.; and Cudré-Mauroux, P. 2018. Are Meta-Paths Necessary? Revisiting Heterogeneous Graph Embeddings. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 437–446.

Linial, N.; London, E.; and Rabinovich, Y. 1995. The geometry of graphs and some of its algorithmic applications. *Combinatorica* 15(2): 215–245.

McDonald, D.; and He, S. 2019. HEAT: Hyperbolic Embedding of Attributed Networks. *arXiv preprint arXiv:1903.03036* .

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Nickel, M.; and Kiela, D. 2017. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, 6338–6347.

Ou, M.; Cui, P.; Pei, J.; Zhang, Z.; and Zhu, W. 2016. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 1105–1114.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710. ACM.

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

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, 593–607. Springer.

Shang, C.; Tang, Y.; Huang, J.; Bi, J.; He, X.; and Zhou, B. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 3060–3067.

Shi, Y.; Gui, H.; Zhu, Q.; Kaplan, L.; and Han, J. 2018a. Aspem: Embedding learning by aspects in heterogeneous information networks. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, 144–152. SIAM.

Shi, Y.; Zhu, Q.; Guo, F.; Zhang, C.; and Han, J. 2018b. Easing embedding learning by comprehensive transcription of heterogeneous information networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2190–2199.

Song, H. H.; Cho, T. W.; Dave, V.; Zhang, Y.; and Qiu, L. 2009. Scalable proximity estimation and link prediction in online social networks. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, 322–335.

Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197* .

Tang, J.; Qu, M.; and Mei, Q. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1165–1174.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, 1067–1077. International World Wide Web Conferences Steering Committee.

Wang, D.; Cui, P.; and Zhu, W. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD*

international conference on Knowledge discovery and data mining, 1225–1234. ACM.

Wang, K.; Liu, Y.; Xu, X.; and Lin, D. 2018. Knowledge graph embedding with entity neighbors and deep memory network. *arXiv preprint arXiv:1808.03752* .

Wang, L.; Lu, Y.; Huang, C.; and Vosoughi, S. 2020. Embedding Node Structural Role Identity into Hyperbolic Space. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2253–2256.

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*, 2022–2032.

Wang, X.; Zhang, Y.; and Shi, C. 2019. Hyperbolic Heterogeneous Information Network Embedding. AAAI.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge graph embedding by translating on hyperplanes. In *Aaai*, volume 14, 1112–1119. Citeseer.

Wilson, B.; and Leimeister, M. 2018. Gradient descent in hyperbolic space. *arXiv preprint arXiv:1805.08207* .

Yang, B.; Yih, W.-t.; He, X.; Gao, J.; and Deng, L. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* .

Zhang, C.; Song, D.; Huang, C.; Swami, A.; and Chawla, N. V. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 793–803.

Zhou, C.; Liu, Y.; Liu, X.; Liu, Z.; and Gao, J. 2017. Scalable graph embedding for asymmetric proximity. In *Thirty-First AAAI Conference on Artificial Intelligence*.