2021 IEEE 37th International Conference on Data Engineering (ICDE) | 978-1-7281-9184-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICDE51399.2021.00193

Node2LV: Squared Lorentzian Representations for Node Proximity

Shanshan Feng
Inception Institute of Artificial Intelligence
Abu Dhabi, UAE
victor_fengss@foxmail.com

Lisi Chen
KAUST
Saudi Arabia
chenlisi.cs@gmail.com

Kaiqi Zhao
University of Auckland
Auckland, New Zealand
kaiqi.zhao@auckland.ac.nz

Wei Wei
Huazhong University of Science and Technology
Wuhan, China
weiw@hust.edu.cn

Fan Li
Fraunhofer Singapore
Singapore
lifan3572@gmail.com

Shuo Shang*
KAUST
Saudi Arabia
jedi.shang@gmail.com

***Abstract*—Recently, network embedding has attracted extensive research interest. Most existing network embedding models are based on Euclidean spaces. However, Euclidean embedding models cannot effectively capture complex patterns, especially latent hierarchical structures underlying in real-world graphs. Consequently, hyperbolic representation models have been developed to preserve the hierarchical information. Nevertheless, existing hyperbolic models only capture the first-order proximity between nodes. To this end, we propose a new embedding model, named Node2LV, that learns the hyperbolic representations of nodes using squared Lorentzian distances. This yields three advantages. First, our model can effectively capture hierarchical structures that come from the network topology. Second, compared with the conventional hyperbolic embedding methods that use computationally expensive Riemannian gradients, it can be optimized in a more efficient way. Lastly, different from existing hyperbolic embedding models, Node2LV captures higher-order proximities. Specifically, we represent each node with two hyperbolic embeddings, and make the embeddings of related nodes close to each other. To preserve higher-order node proximity, we use a random walk strategy to generate local neighborhood context. We conduct extensive experiments on four different types of real-world networks. Empirical results demonstrate that Node2LV significantly outperforms various graph embedding baselines.**

***Index Terms*—Squared Lorentzian Distance, Graph Embedding, Node Proximity**

I. Introduction

Recently, network embedding has attracted extensive research interest, aiming to learn low-dimensional representations of nodes to preserve the original network structures. Various network embedding models have been proposed [1], [2]. For instance, some embedding methods [3]–[5] exploit the word2vec technique [6] to learn node representations. Recently, deep learning models [7], [8] and graph neural networks [9], [10] have also been utilized to learn node representations. However, most existing models represent the nodes in a Euclidean space, and therefore may fail to capture the complex patterns of graphs, especially latent hierarchical structures [11], which have been demonstrated to exist in many real-world information graphs [12].

Hyperbolic embedding models [11], [13]–[17] have been proposed to capture hierarchical structures in hyperbolic spaces. Compared with the conventional Euclidean spaces, hyperbolic spaces are much more effective for learning tree-like data, since they can be naturally viewed as continuous versions of tree structures [18]. In addition, hyperbolic spaces have higher representative strength, especially in low-dimensional spaces, because they expand much faster than Euclidean spaces [11], [18].

However, existing hyperbolic embedding models have several limitations. First, they mainly focus on tree datasets where hierarchies are clearly defined. Although many real-world graphs exhibit tree-like properties to some degree [12], there are no explicit hierarchies defined in these graphs. Hyperbolic embedding methods have not been widely explored on general graphs. Second, existing hyperbolic embedding models only preserve the first-order neighborhood relationships. Higher-order neighborhoods have proven important for capturing the node proximity and improving the quality of network embeddings [19]. Third, existing hyperbolic embedding models commonly exploit the Riemannian gradient method [20] to learn node representations, which is computationally expensive.

To learn the representations of nodes, we propose a novel hyperbolic network embedding model, Node2LV, which exploits the squared Lorentzian distance [15]. Specifically, we associate each node with two vectors in the hyperbolic space: a source vector that indicates its capability of influencing other nodes, and a target vector that reflects the likeliness of being influenced by other nodes. To preserve higher-order proximity, Node2LV explores wider neighborhood context and thus improves the embedding quality. Lastly, instead of using the expensive Riemannian gradient method [13], [20], Node2LV can be trained by an efficient Euclidean stochastic gradient descent (SGD) approach to reduce the training time.

The main contributions of this work are summarized as follows.

* Corresponding author.

978-1-7281-9184-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDE51399.2021.00193

- We investigate a hyperbolic network embedding problem, in which we aim to preserve the hierarchical structures of general networks.
- We develop a new model, named Node2LV, to learn hyperbolic embeddings of graphs based on the squared Lorentzian distance. Node2LV is capable of capturing hierarchical structures and higher-order node proximity.
- We conduct extensive experiments on four real-world networks with two evaluation tasks. The results demonstrate that Node2LV significantly outperforms various state-of-the-art network embedding models.

II. Related Work

A. Network Embedding

A variety of network embedding models have been proposed, such as word2vec techniques [3]–[6]. Tang *et al.* [5] designed a large-scale information network embedding model, which combines first-order and second-order proximity. Perozzi *et al.* [4] proposed the DeepWalk model, which learns node representations from neighborhood context generated by random walks. To explore a flexible neighborhood context, Grover *et al.* [3] developed the Node2vec model, which uses a biased random walk strategy to combine breadth-first and depth-first sampling. To capture both asymmetric and high-order similarities between nodes, Zhou *et al.* [21] proposed a rooted pagerank path sampling approach to generate directed sequences. Recently, several deep neural methods [7], [8] have also been proposed for network embedding. Wang *et al.* [8] developed a structural deep network embedding method (SDNE), which utilizes a deep auto-encoder framework to capture both first-order and second-order node proximity. Various graph neural networks (GNNs) have been proposed [10], which learn network representations by collectively aggregating node features from graph structures. However, most existing network embedding methods are based on Euclidean spaces, which limits their ability to model hierarchical structures [11], [16].

B. Hyperbolic Embedding

Recently, hyperbolic embedding approaches have been proposed to capture hierarchical structures [11], [13], [14]. Nickel *et al.* developed a Poincaré model [11] and a Lorentz model [13] for learning hyperbolic embeddings. Sala *et al.* [14] proposed a combinatorial construction process for encoding hierarchies into a hyperbolic space. Ganea *et al.* [22] developed a hyperbolic entailment cone embedding model and Suzuki *et al.* [23] proposed a hyperbolic disk embedding model to capture asymmetric relations. Law *et al.* [15] proposed to use the squared Lorentzian distance to encode hierarchical structures. Our work is different from [15] in two aspects. First, [15] mainly targets at learning tree datasets with well-defined hierarchies. In contrast, we exploit the squared Lorentzian distance for general graphs without explicit tree structures. Second, [15] only considers the first-order node proximity. We explore a wider local neighborhood where higher-order relations are captured. Recently, based on the Poincaré distance, Wang *et al.* [24] developed the HHNE method to embed heterogeneous networks into hyperbolic spaces. Different from HHNE, we utilize the squared Lorentzian distance, which has proven to be more effective in capturing hierarchical structures than the Poincaré model [15]. In addition, HHNE uses the Riemannian stochastic gradient descent method [20], while Node2LV employs a more efficient optimization procedure.

Recently, hyperbolic geometry has been integrated into various deep learning frameworks, including recurrent neural networks [25], attention networks [26], and graph neural networks [16], [17]. The hyperbolic embedding problem has been studied in multiple application domains, e.g., word embedding [27] and recommender systems [28], [29]. It would be interesting to combine our Node2LV model with deep neural network frameworks and apply it to different research tasks. We leave this as future work.

III. Problem Statement

To study the characteristics of network structures, we examine the degree distributions of four graph datasets, which will be described in detail in Section V. We observe that the node degrees follow power-law distributions, which is consistent with the fact that power-law distributions widely exist in many complex networks [30]. The power-law distributions of node degrees indicate underlying hierarchical structures [11], [31], which are commonly observed in various information networks [12]. However, this important network characteristic has not been widely investigated in existing network embedding studies. Most of the existing network embedding models are based on Euclidean spaces, and therefore cannot effectively capture the underlying hierarchical structures in the network topology [11]. To address this, we aim at developing novel network embedding models that are able to preserve node proximities and capture hierarchical structures simultaneously.

A network is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the edge set. Each directed edge $(a \to b)$ indicates that node a is connected to node b. In undirected graphs, one edge (a, b) can be regarded as two directed edges $(a \to b)$ and $(b \to a)$. A node has two roles: acting as a source node to influence other nodes, and acting as a target node to be influenced by other nodes. To model the different roles, we exploit two representation vectors for each node v: $\mathbf{x}_v^s$ reflects the role of source node and $\mathbf{x}_v^t$ reflects the role of target node. The research problem is formally defined as follows.

Definition 1: **(Hyperbolic Network Representation)** Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we aim at representing each node $v \in \mathcal{V}$ with two vectors in a d-dimensional hyperbolic space, $\mathbf{x}_v^s \in R^d$ and $\mathbf{x}_v^t \in R^d$, such that the node proximities are preserved with the node representations.

IV. Node2LV Representation Model

A. Squared Lorentzian Distances

Hyperbolic spaces are spaces of constant negative curvature, and can be described by several models, such as the Poincaré ball model [11], Klein model [26], and hyperboloid model [13]. In this paper, we mainly consider the hyperboloid model, which is defined as $\mathcal{H}_\beta^d = \{\mathbf{x} \in R^{d+1} : \langle \mathbf{x}, \mathbf{x} \rangle_\mathcal{L} =$

$-\beta\}$. Here, $\mathbf{x} = (x_0, x_1, ..., x_d)$ is a $d+1$ dimensional vector with $x_0 = \sqrt{\beta + \sum_{i=1}^{d} x_i^2} > 0$. The curvature of the hyperbolic space is determined by the parameter $\beta > 0$. Given $\mathbf{x}_u \in R^{d+1}$ and $\mathbf{x}_v \in R^{d+1}$, $\langle \mathbf{x}_u, \mathbf{x}_v \rangle_{\mathcal{L}}$ denotes the Lorentzian scalar product of $\mathbf{x}_u$ and $\mathbf{x}_v$, and can be computed as

$$\langle \mathbf{x}_u, \mathbf{x}_v \rangle_{\mathcal{L}} = -x_{u,0} \cdot x_{v,0} + \sum_{i=1}^{d} x_{u,i} \cdot x_{v,i}. \tag{1}$$

For any two nodes $\mathbf{x}_u \in \mathcal{H}_\beta^d$ and $\mathbf{x}_v \in \mathcal{H}_\beta^d$, their Lorentzian scalar product $\langle \mathbf{x}_u, \mathbf{x}_v \rangle_{\mathcal{L}} \leq -\beta$.

Given two vectors $\mathbf{x}_u \in \mathcal{H}_\beta^d$ and $\mathbf{x}_v \in \mathcal{H}_\beta^d$, the squared Lorentzian distance [32] is defined as

$$D_{uv}^{\mathcal{SL}} = ||\mathbf{x}_u - \mathbf{x}_v||_{\mathcal{L}}^2 = -2\beta - 2\langle \mathbf{x}_u, \mathbf{x}_v \rangle_{\mathcal{L}}. \tag{2}$$

Compared with other hyperbolic metrics [11], [13], the squared Lorentzian distance can make the parent nodes have smaller L2-norms than their children [15]. Hence, it is able to effectively infer hierarchical structures by the L2-norms. Another benefit of the squared Lorentzian distance is that it can be approximately optimized by conventional gradient algorithms, e.g., stochastic gradient descent, because Equation 2 is well defined and has a smooth space structure.

B. Modeling Node Proximity

To learn the representations of nodes, we propose to use the squared Lorentzian distance to capture their proximity. Each node v is associated with two representation vectors: a source vector $\mathbf{x}_v^s \in R^d$ and a target vector $\mathbf{x}_v^t \in R^d$. For two nodes connected by a directed relation $(u \to v)$, their representations $\mathbf{x}_u^s \in R^d$ and $\mathbf{x}_v^t \in R^d$ should be close in the hyperbolic space.

Given a representation vector $\mathbf{x} \in R^d$, we convert it to a hyperboloid model by the following mapping function:

$$f : \mathbf{x} = (x_1, x_2, .., x_d) \to \mathbf{x}_{\mathcal{H}} = (x_0, x_1, x_2, .., x_d), \tag{3}$$

where $x_0 = \sqrt{\beta + \sum_{i=1}^{d} x_i^2} = \sqrt{\beta + ||\mathbf{x}||^2}$, and $\mathbf{x}_{\mathcal{H}} \in \mathcal{H}_\beta^d$ is the mapped vector in the hyperboloid model.

Using Equation 3, we convert $\mathbf{x}_u^s$ and $\mathbf{x}_v^t$ to the hyperboloid model. Next, combining Equations 1 and 2, the squared Lorentzian distance between $\mathbf{x}_u^s \in R^d$ and $\mathbf{x}_v^t \in R^d$ is defined as follows:

$$\begin{aligned} D_{uv} &= ||\mathbf{x}_{u\mathcal{H}}^s, \mathbf{x}_{v\mathcal{H}}^t||_{\mathcal{L}}^2 \\ &= -2\beta - 2\langle \mathbf{x}_{u\mathcal{H}}^s, \mathbf{x}_{v\mathcal{H}}^t \rangle_{\mathcal{L}} \\ &= 2\big(\sqrt{\beta + ||\mathbf{x}_u^s||^2}\sqrt{\beta + ||\mathbf{x}_v^t||^2} - \langle \mathbf{x}_u^s, \mathbf{x}_v^t \rangle - \beta\big). \end{aligned} \tag{4}$$

Here, $\langle \mathbf{x}_u^s, \mathbf{x}_v^t \rangle = \sum_{i=1}^{d} x_{u,i}^s \cdot x_{v,i}^t$ is the dot product of $\mathbf{x}_u^s$ and $\mathbf{x}_v^t$, and $||\cdot||$ denotes the corresponding L2-norm.

Given an observed relation $(u \to v)$, the probability $\Pr(v|u)$ can be formulated by a softmax function with the squared Lorentzian distance:

$$\Pr(v|u) = \frac{e^{-D_{uv}}}{Z(u)}, \tag{5}$$

where $Z(u) = \sum_{k \in \mathcal{V}} e^{-D_{uk}}$ is the normalization term. It is computationally expensive to directly compute Equation 5, since calculating $Z(u)$ needs to enumerate all nodes $k \in \mathcal{V}$. To alleviate this issue, we adopt the *negative sampling* technique [6], which is widely utilized to compute softmax functions. Specifically, for the relation $(u \to v)$, a small set of negative nodes $\mathcal{N} \subset \mathcal{V}$ are randomly sampled. Then the log probability can be estimated by:

$$\log \Pr(v|u) \approx \log \sigma(-D_{uv}) + \sum_{k \in \mathcal{N}} \log \sigma(D_{uk}), \tag{6}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the Sigmoid function. Intuitively, maximizing the log probability in Equation 6 should push unconnected nodes (u and k) away and pull connected nodes (u and v) close.

We exploit the skip-gram architecture [6], which involves predicting the neighborhood context of a given node. For a node u, its context C_u contains a set of nodes that have close relations to this node. Assuming that the nodes $\{v \in C_u\}$ are independent, the probability $\Pr(C_u|u)$ is defined as:

$$\Pr(C_u|u) = \prod_{v \in C_u} \Pr(v|u). \tag{7}$$

The objective of the Node2LV model is to maximize the log probability of observing all relations between nodes:

$$O = \sum_{u \in \mathcal{V}} \log \Pr(C_u|u) = \sum_{u \in \mathcal{V}} \sum_{v \in C_u} \log \Pr(v|u). \tag{8}$$

C. Generating Neighborhood Context

To learn node representations, the first step is to extract the neighborhood context, which is important for preserving the network structure [3]. To capture the node-node relations, Node2LV extracts the neighborhood context from the network topology, which exhibits implicit hierarchical structures, as described in Section III.

Network embedding models learn node representations by capturing node proximity. However, existing hyperbolic embedding models [11], [13], [15] only preserve the first-order connections between nodes, which may not reflect the complex neighborhood structure. To utilize richer proximity information, we adopt the random walk strategy [3]. In each generated random walk sequence $S = (v_1, v_2, ..., v_l)$, successive nodes $\{v_{i+1}, v_{i+2}, ..., v_{i+w}\}$ within a window of size w are extracted as the neighborhood context C_{v_i} of node v_i. The parameter w controls the size of the neighborhood. The extracted neighborhood context contains both first-order and higher-order neighbors.

D. Optimization

Due to the Riemannian manifold structure, most existing hyperbolic embedding methods [13], [24] need to calculate the Riemannian gradient using an exponential map in a hyperbolic space, which is time consuming. In contrast, the squared Lorentzian distance can be effectively optimized using a standard gradient method [15], which is much more efficient.

To optimize Equation 8, we exploit the stochastic gradient descent method to learn the node representations. The node representations are initialized randomly at the beginning. Then, we iteratively update them by capturing the node

proximity. Specifically, given a node pair (u, v), we update the corresponding representations Θ by calculating the gradient:

$$\Theta \leftarrow \Theta + \gamma \frac{\partial}{\partial \Theta} \left(\log(\Pr(v|u)) \right), \tag{9}$$

where γ is the learning rate and $\frac{\partial}{\partial \Theta}$ indicates the gradient of parameters Θ.

For each node pair, the training time complexity is $O(I \cdot |N| \cdot d)$, where I is the number of iterations until convergence, $|N|$ is the number of negative samples (typically 5-10), and d indicates the number of embedding dimensions. The total number of training node pairs is $O(\gamma \cdot l \cdot w \cdot |V|)$, where γ is the number of random walks per node, l is the length of the walk sequence, w is the window size, and $\mathcal{V}$ is the size of the node set. Therefore, the total time complexity of Node2LV is $O(\gamma \cdot l \cdot w \cdot |V| \cdot I \cdot |N| \cdot d)$.

V. Experiments

A. Experimental Settings

1) Datasets: We evaluate the performance of Node2LV on four publicly available datasets.

- **Protein** [33] is a graph of protein-protein interactions for Homo sapiens. The categories denote the biological states of proteins.
- **Wikipedia** [34] is a graph of occurrence words in a subset of the Wikipedia text dump. The categories are the part-of-speech tags of words.
- **BlogCatalog** [35] is a social network of bloggers from the BlogCatalog website. The categories represent the topics listed by bloggers.
- **Flickr** [35] is a social network of users from the photo sharing website Flickr. The categories denote the interest groups of the users.

The statistics of the four datasets are reported in Table I.

TABLE I
Statistics of four networks

Dataset	#Nodes	#Edges	#Average Degree
Protein	3,890	76,584	39.37
Wikipedia	4,777	184,812	77.37
BlogCatalog	10,312	667,966	64.77
Flickr	80,513	11,799,764	146.55

2) Evaluation Tasks: For performance evaluation, we consider two main research tasks as follows:

- **RT1: Link Prediction**. A good network embedding method is expected to have satisfactory generalization power to infer unknown links. We exploit the edges in the tuning data or test data as ground-truth positive links. An equal number of unconnected node pairs are randomly sampled and used as negative links.
- **RT2: Node Recommendation**. Network embedding models should identify the potential links for each individual node. Based on the learned embeddings, we attempt to recommend the top-K candidates with the highest proximity scores for a given node.

Given a network, we randomly choose 80% of the edges to train the embedding model, 10% to tune the hyperparameters, and the remaining 10% to evaluate the performance.

3) Evaluated Methods: We evaluate the performance of the following approaches.

- **DeepWalk** [4]: This is a widely used network embedding method. It first builds a binary tree to organize all the nodes, and then uses the hierarchical softmax technique [6] to learn the node representations.
- **Node2vec** [3]: This is another widely used network embedding model. It utilizes the word2vec framework with the negative sampling technique [6] to learn the node representations.
- **SDNE** [8]: This is a deep network embedding model, which utilizes an auto-encoder framework to preserve both first-order and second-order node proximity.
- **HHNE** [24]: This is the state-of-the-art hyperbolic network embedding for heterogeneous information networks. It exploits the Poincaré distance to model node proximity and utilizes the Riemannian stochastic gradient descent method to learn node embeddings. It uses meta-path guided random walks [36] to extract heterogeneous neighborhood context for each node. Since the datasets used in this work are homogeneous graphs, we use the random walks [4] to generate neighborhood context.
- **SLDE** [15]: This is the state-of-the-art hyperbolic representation model for hierarchy datasets. It only preserves first-order node proximity.
- **APP** [21]: This is an asymmetric proximity preserving graph embedding model. It utilizes the Euclidean space to capture higher-order and asymmetric similarities between node pairs.
- **Node2LV**: Our proposed hyperbolic network embedding model for preserving node proximity as well as capturing the implicit hierarchical structure.

4) Parameter Settings: To provide a fair comparison, for random walk based methods (DeepWalk, Node2vec, HHNE, APP, and Node2LV), we use the same settings to generate neighborhood context: number of walks per node $\gamma = 10$, walk length $l = 80$, and window size $w = 5$. Following previous studies [3], [4], [21], we set the number of dimensions $d = 128$ for all the embedding methods. The default number of negative samples N is set as 5 in the methods with negative sampling techniques. In SDNE, we use two hidden layers (with 1000 and 128 dimensions, respectively) in the encoder. We implement Node2LV in C++ and conduct experiments in an Ubuntu server with a 48-core Intel Xeon(R), 2.20 GHz CPU and 64 GB memory.

B. Experimental Results

We present the experimental results of the two research tasks on the four datasets.

1) RT1: Link Prediction: Following [3], [24], area under the curve (AUC) is used as the metric to evaluate the performance of evaluated methods on the link prediction task. The experimental results are reported in Table II. The proposed

TABLE II
THE AUC SCORES OF LINK PREDICTION ON THE FOUR DATASETS. BEST RESULTS ARE IN BOLDFACE.

Dataset	#Dimension	DeepWalk	Node2vec	SDNE	HHNE	SLDE	APP	Node2LV
Protein	8	0.6903	0.5675	0.7384	0.7927	0.8328	0.8318	**0.8399**
	32	0.6740	0.6511	0.7427	0.8016	0.8499	0.8322	**0.8696**
	128	0.6549	0.6355	0.7446	0.8019	0.8506	0.8327	**0.8720**
Wikipedia	8	0.5458	0.5368	0.6585	0.5942	0.6127	0.6773	**0.6867**
	32	0.5479	0.5370	0.6667	0.6266	0.6841	0.6861	**0.7204**
	128	0.5491	0.5585	0.6749	0.6364	0.6959	0.6831	**0.7557**
BlogCatalog	8	0.7172	0.6991	0.8101	0.7553	0.8274	0.8399	**0.8635**
	32	0.7394	0.7172	0.8145	0.7765	0.8564	0.8479	**0.8878**
	128	0.7991	0.7918	0.8239	0.7776	0.8578	0.8323	**0.8988**
Flickr	8	0.8516	0.8272	0.7779	0.9016	0.9388	0.9354	**0.9454**
	32	0.8662	0.8351	0.8262	0.9024	0.9438	0.9361	**0.9507**
	128	0.8562	0.8413	0.8446	0.9014	0.9419	0.9401	**0.9508**

TABLE III
THE RESULTS OF NODE RECOMMENDATION ON THE FOUR DATASETS. BEST RESULTS ARE IN BOLDFACE.

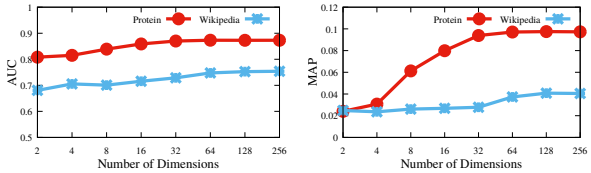
Dataset	#Dimension	DeepWalk	Node2vec	SDNE	HHNE	SLDE	APP	Node2LV
Protein	MAP	0.0336	0.0274	0.0219	0.0696	0.0921	0.0665	**0.0954**
	Pre@10	0.0122	0.0089	0.0126	0.0350	0.0449	0.0302	**0.0528**
	Rec@10	0.0453	0.0311	0.0308	0.1094	0.1427	0.1019	**0.1561**
Wikipedia	MAP	0.0198	0.0204	0.0237	0.0237	0.0302	0.0231	**0.0413**
	Pre@10	0.0077	0.0084	0.0163	0.0101	0.0138	0.0086	**0.0195**
	Rec@10	0.0330	0.0335	0.0319	0.0370	0.0429	0.0362	**0.0671**
BlogCatalog	MAP	0.0048	0.0067	0.0476	0.0095	0.0319	0.0175	**0.0568**
	Pre@10	0.0039	0.0055	0.0402	0.0076	0.0206	0.0119	**0.0454**
	Rec@10	0.0052	0.0103	0.0871	0.0168	0.0440	0.0217	**0.1037**
Flickr	MAP	0.0143	0.0137	0.0167	0.0116	0.0226	0.0238	**0.0277**
	Pre@10	0.0119	0.0158	0.0284	0.0139	0.0305	0.0299	**0.0413**
	Rec@10	0.0114	0.0148	0.0116	0.0111	0.0229	0.0312	**0.0349**

Node2LV obtains the highest AUC scores among all the evaluated methods. Overall, as shown in Table II, the AUC scores increase with the number of dimensions for all the embedding methods.

2) RT2: Node Recommendation: For a given node, we rank all the unconnected nodes by their proximity scores to this node. Mean average precision (MAP) is used to evaluate the ranking of ground-truths in the overall ranking list. We also consider a top-K recommendation setting by selecting the top-K ranked nodes from the ranking list, and then use precision@K (Pre@K) and recall@K (Rec@K) to evaluate the models. For fair comparison, the number of dimensions d is set to 128 for all the evaluated methods. The experimental results of recommending potential connections are reported in Table III. Node2LV significantly outperforms the Euclidean embedding models (DeepWalk, Node2vec, and SDNE), obtaining at least 12.9% higher Pre@10 scores on all four datasets. The performance of SLDE is not as good as Node2LV, which demonstrates the benefits of capturing higher-order node proximity. Node2LV performs much better than its Euclidean competitor APP. This indicates that the squared Lorentzian distance can effectively capture the network structures and has strong generalization power. To sum up, Node2LV consistently obtains the best performance, which demonstrates the superiority of our proposed hyperbolic embedding model.

C. Effect of Parameters

For the parameter sensitivity analysis, we employ Protein and Wikipedia as examples. We consider two metrics: AUC for the link prediction task and MAP for the node recommendation task. To investigate the effect of the number of dimensions d, we show the experimental results with different d in Figure 1. We observe that the performance increases with d. However, the performance becomes saturated when $d = 128$, after which the scores stop increasing. Empirically, the number of dimensions d can be set to 64-128.



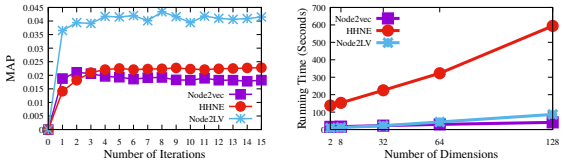
(a) AUC of link prediction (b) MAP of node recommendation

Fig. 1. The effect of number of dimensions d.

D. Training Efficiency

Next, we study the efficiency of the proposed optimization algorithm for Node2LV. We compare the running time of Node2LV with Node2vec and HHNE, which are Euclidean and hyperbolic embedding models, respectively. Note that we apply exactly the same settings to the three models for fair comparison. In Figure 2(a), we present the MAP scores with different numbers of iterations I for the node recommendation task on the Wikipedia dataset. Empirically, all three methods converge after 3-5 iterations with a learning rate of 0.01. Node2LV obtains higher MAP scores than Node2vec and HHNE, which demonstrates the advantage of the squared Lorentzian distance.

Furthermore, we evaluate the running time of different embedding methods with only one thread. In practice, a multi-thread framework can be exploited to enhance the computational efficiency. We report the running time of one iteration with different numbers of dimensions d on the Wikipedia dataset in Figure 2(b). For all methods, the running time linearly increases with the number of dimensions d. Node2vec is the fastest because it utilizes the inner product of two embeddings to reflect the node proximity. Both Node2LV and HHNE are hyperbolic embedding methods. As we can observe, Node2LV takes much less time than HHNE. Specifically, Node2LV requires 86 seconds while HHNE takes 594 seconds to complete one iteration when $d = 128$. Overall, Node2LV is 6–10x faster than HHNE on the four evaluated datasets. This is because HHNE uses the computationally expensive Riemannian stochastic gradient descent method, while Node2LV learns the node embeddings with an efficient approach. To sum up, the results in Figure 2 demonstrate the superiority of Node2LV over HHNE in terms of both effectiveness and efficiency in learning node embeddings.



(a) Convergence speed (b) Running time of one iteration

Fig. 2. Training efficiency analysis of different models on Wikipedia.

VI. Conclusion

In this paper, we study the problem of learning hyperbolic network embeddings, aiming at preserving both hierarchical structures and high-order node proximity in a low-dimensional hyperbolic space. Using the squared Lorentzian distance, we propose a new Node2LV model to capture the hierarchical structures of real-world networks. To the best of our knowledge, this is the first work to use the squared Lorentzian distance for capturing the higher-order node proximity. We conduct extensive experiments on four different types of networks. The results demonstrate that the proposed Node2LV model significantly outperforms various baseline methods.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant No.61602197, and in part by Equipment Pre-Research Fund for The 13th Five-year Plan under Grant No.41412050801.

References

[1] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, 2018.
[2] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, 2018.
[3] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016.
[4] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*, 2014.
[5] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*, 2015.
[6] T. Mikolov and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.
[7] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *AAAI*, 2016.
[8] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *SIGKDD*, 2016.
[9] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017.
[10] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.
[11] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," in *NeurIPS*, 2017.
[12] A. B. Adcock, B. D. Sullivan, and M. W. Mahoney, "Tree-like structure in large social and information networks," in *ICDM*, 2013.
[13] M. Nickel and D. Kiela, "Learning continuous hierarchies in the lorentz model of hyperbolic geometry," *ICML*, 2018.
[14] C. De Sa, A. Gu, C. Ré, and F. Sala, "Representation tradeoffs for hyperbolic embeddings," *ICML*, 2018.
[15] M. T. Law, R. Liao, J. Snell, and R. S. Zemel, "Lorentzian distance learning for hyperbolic representations," *ICML*, 2019.
[16] I. Chami, Z. Ying, C. Ré, and J. Leskovec, "Hyperbolic graph convolutional neural networks," in *NeurIPS*, 2019.
[17] Q. Liu, M. Nickel, and D. Kiela, "Hyperbolic graph neural networks," in *NeurIPS*, 2019.
[18] D. V. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguñá, "Hyperbolic geometry of complex networks," *Physical Review E*, 2010.
[19] C. Yang, M. Sun, Z. Liu, and C. Tu, "Fast network embedding enhancement via high order proximity approximation." in *IJCAI*, 2017.
[20] S. Bonnabel, "Stochastic gradient descent on riemannian manifolds," *IEEE Transactions on Automatic Control*, 2013.
[21] C. Zhou, Y. Liu, X. Liu, Z. Liu, and J. Gao, "Scalable graph embedding for asymmetric proximity," in *AAAI*, 2017.
[22] O. Ganea, G. Becigneul, and T. Hofmann, "Hyperbolic entailment cones for learning hierarchical embeddings," in *ICML*, 2018.
[23] R. Suzuki, R. Takahama, and S. Onoda, "Hyperbolic disk embeddings for directed acyclic graphs," in *ICML*, 2019.
[24] X. Wang, Y. Zhang, and C. Shi, "Hyperbolic heterogeneous information network embedding," in *AAAI*, 2019.
[25] O. Ganea, G. Bécigneul, and T. Hofmann, "Hyperbolic neural networks," in *NeurIPS*, 2018.
[26] C. Gulcehre, M. Denil, M. Malinowski, A. Razavi, R. Pascanu, K. M. Hermann, P. Battaglia, V. Bapst, D. Raposo, A. Santoro *et al.*, "Hyperbolic attention networks," *arXiv preprint arXiv:1805.09786*, 2018.
[27] A. Tifrea, G. Bécigneul, and O.-E. Ganea, "Poincaré glove: Hyperbolic word embeddings," *ICLR*, 2019.
[28] L. Vinh Tran, Y. Tay, S. Zhang, G. Cong, and X. Li, "Hyperml: A boosting metric learning approach in hyperbolic space for recommender systems," in *WSDM*, 2020.
[29] S. Feng, L. V. Tran, G. Cong, L. Chen, J. Li, and F. Li, "Hme: A hyperbolic metric embedding approach for next-poi recommendation," in *SIGIR*, 2020.
[30] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, 2002.
[31] E. Ravasz and A.-L. Barabási, "Hierarchical organization in complex networks," *Physical review E*, vol. 67, no. 2, p. 026112, 2003.
[32] J. G. Ratcliffe, S. Axler, and K. Ribet, *Foundations of hyperbolic manifolds*. Springer, 1994, vol. 3.
[33] B.-J. Breitkreutz, C. Stark, T. Reguly, L. Boucher, A. Breitkreutz, M. Livstone, R. Oughtred, D. H. Lackner, J. Bähler, V. Wood *et al.*, "The biogrid interaction database: 2008 update," *Nucleic acids research*, 2008.
[34] M. Mahoney, "Large text compression benchmark," *www.mattmahoney.net/dc/textdata*, 2011.
[35] L. Tang and H. Liu, "Relational learning via latent social dimensions," in *SIGKDD*, 2009.
[36] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *SIGKDD*, 2017.