Multi-Vector Embedding on Networks with Taxonomies

Yue Fan , **Xiuli Ma**[*]
Key Laboratory of Machine Perception (MOE), School of Artificial Intelligence, Peking University, Beijing, China
fanyue@pku.edu.cn, xlma@pku.edu.cn

Abstract

A network can effectively depict close relationships among its nodes, with labels in a taxonomy describing the nodes' rich attributes. Network embedding aims at learning a representation vector for each node and label to preserve their proximity, while most existing methods suffer from serious underfitting when dealing with datasets with dense node-label links. For instance, a node could have dozens of labels describing its diverse properties, causing the single node vector overloaded and hard to fit all the labels. We propose HIerarchical Multi-vector Embedding (HIME), which solves the underfitting problem by adaptively learning multiple 'branch vectors' for each node to dynamically fit separate sets of labels in a hierarchy-aware embedding space. Moreover, a 'root vector' is learned for each node based on its branch vectors to better predict the sparse but valuable node-node links with the knowledge of its labels. Experiments reveal HIME's comprehensive advantages over existing methods on tasks such as proximity search, link prediction and hierarchical classification.

1 Introduction

A network is able to depict the proximity among nodes. Extra node properties are described by labels organized as a taxonomy, which is often presented as a directed acyclic graph (DAG). For instance, in a Protein-Protein Interaction (PPI) network with Gene Ontology (GO) being the taxonomy, network edges reveal interactions among proteins, while different hierarchical GO terms of a protein tell its diverse biological properties. Generally, a node can have multiple label paths in the taxonomy, as shown in figure 1 (a).

Traditional heterogeneous network embedding methods serve as an option for co-embedding the network and the taxonomy. They learn a representation vector for each node and label, so that the proximity is preserved by the distances among the vectors. However, on datasets with dense node-label relationships, the single representation vector of a node could be overloaded and hard to fit the node's different labels,

[*]Corresponding author: Xiuli Ma.

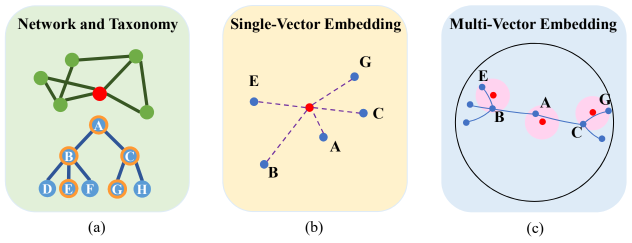

Figure 1: The difference between single-vector and multi-vector embedding. Given a network node with five circled labels in a taxonomy (a), single-vector embedding learns a vector in the middle of the five labels without being close to any of them, causing vague representation (b). However, by giving multiple vectors to a node, the underfitting problem can be overcome in a Poincaré ball (c).

which causes the underfitting problem [Yang *et al.*, 2020]. As illustrated in figure 1 (b), in order to minimize the overall distance to its labels, the node vector is embedded near the center of its five labels without being close to any of them. This underfitting problem seriously decreases the quality of the representation vectors, causing them containing vague or even misleading information.

To solve the underfitting problem, one way is to optimize the positions of the labels so as to minimize the probability of underfitting, which can be achieved by the Poincaré ball model [Nickel and Kiela, 2017]. A Poincaré ball is a hyperbolic space with constant negative curvature of -1, and can be viewed as a continuous version of trees. It naturally caters to the taxonomy, with high-level labels embedded near the origin while low-level ones near the ball surface, and labels in a label-path embedded in the same direction. Therefore, a node vector is less likely to underfit labels belonging to a single label path. But using the Poincaré ball alone does not necessarily tackle the underfitting problem, since a node can have multiple label paths embedded away from each other.

Based on the Poincaré ball, an effective way to solve the underfitting problem is to 'divide and conquer'. Specifically, we allow multiple 'branch vectors' for each node to fit different sets of its labels. A branch vector can reflect a group of closely related low-level characteristics, or indicate the general concepts that the node belongs to. Therefore, multi-vector embedding provides a more accurate and hierarchical representation as shown in figure 1 (c). The branch vector number is adaptive to the node's label distribution. Besides, the Least Recently Used (LRU) policy is employed to balance the loads among the branch vectors of the node.

The branch vectors can then be utilized in preserving sparse but valuable node-node relationships. In a gene regulatory network for instance, it is expensive and intractable to biologically examine all the regulations among up to 27000 human genes, making the label information crucial for predicting unseen regulations. Given a node with its label information learned by its branch vectors, we use a single 'root vector' aggregating these information to better preserve existing links and predict the potential ones.

We conclude our method as HIerarchical Multi-vector Embedding (HIME), which learns a representation vector for each label and multiple branch vectors for each node to fit its different label clusters. A single root vector is then learned for each node based on its branch vectors to preserve node-node proximity. Our work makes the following contributions: (1) We propose multi-vector embedding to solve the underfitting problem. (2) The LRU policy is applied to balance the loads among the branch vectors. (3) The root vector aggregated from the branch vectors better preserves the node proximity. (4) We conduct extensive experiments to show HIME's comprehensive advantages over existing methods.

2 Related Work

Network embedding aims at learning representation vectors for network nodes to preserve the node proximity. The methodologies of network embedding can be mainly categorized into matrix factorization, Skip-Gram with walk patterns [Grover and Leskovec, 2016; Tang *et al.*, 2015] and neural networks [Wang *et al.*, 2018]. These classic methods laid the foundation for later variants such as taxonomy-related embedding and multi-aspect embedding.

2.1 Taxonomy-Related Embedding

Most taxonomy-related embedding methods co-embed network nodes and taxonomy labels. Unlike the entity typing task [Ren *et al.*, 2016] in natural language processing that focuses on preserving node-label relationships, taxonomy-related network embedding should take both node-node and node-label relationships into account.

Poincaré [Nickel and Kiela, 2017] and Lorentz [Nickel and Kiela, 2018] both embed WordNet into a hyperbolic space so as to maintain the hypernym relationships. The hyperbolic space is later applied by Tag2Vec [Wang *et al.*, 2019] that co-embeds the network and the taxonomy by Skip-Gram with meta-paths. The three methods ensure the universality of the embedding vectors by using a single hyperbolic space, but they suffer from the underfitting problem. JOIE [Hao *et al.*, 2019] embeds nodes and labels into two separate spaces connected by a non-linear affine transformation. TaxoGAN [Yang *et al.*, 2020] spares an individual embedding space for every label where its nodes and its child labels are embedded by adversarial training, while its drawback is also obvious since the representation vectors only function locally in diverse spaces.

In general, these taxonomy-related methods are blocked by the trade-off between the universality and the accurateness of the representation vectors. By contrast, HIME generates a precise embedding in a single Poincaré ball.

2.2 Multi-Aspect Embedding

Several recent works have revealed the necessity of learning multiple vectors to capture different aspects of the node. For instance, PolyDW [Liu *et al.*, 2019] represents each facet of an item using a single vector, and Splitter [Epasto and Perozzi, 2019] allows multiple node embedding vectors to encode different communities. The drawback of them is that the aspects of the node need to be determined in advance and are fixed during the learning process, while Asp2Vec [Park *et al.*, 2020] employs random walks to dynamically assign aspects to each node according to its local context.

Though multiple vectors are learned for each node, given a specific aspect, these methods are in essence single-vector embedding since each node has only a single vector in an aspect. By contrast, the computation of the distance between a node and a label in HIME involves all the node's branch vectors and the label vector.

3 Problem Statement

HIME takes the following data as inputs:

- a network $\mathcal{N} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is the set of nodes, and $\mathcal{E} = \{e_{i,j}\}_{i,j}^{n}$ is the set of edges.
- a taxonomy $\mathcal{T} = \{\mathcal{L}, \mathcal{U}\}$, where $\mathcal{L} = \{l_1, l_2, ..., l_m\}$ is the set of labels, and $\mathcal{U} = \{(l_i, l_j) \mid l_j \in Pa(l_i)\}$ is the set of parent-child edges, where $Pa(l_i)$ is the set of l_i's parents in the DAG.
- The label assignment $\mathcal{A} = A_1 \cup A_2 \cup ... \cup A_n$, where A_i is the set of node-label links of node v_i. Once a label appears in A_i, its ancestors will also appear in A_i.

HIME learns a vector for each label and multiple vectors for each node. A node has several 'branch vectors' to fit its labels and a single 'root vector' to preserve its node-node proximity. Formally, given branch limit K, HIME learns:

- a set of label vectors $\mathcal{Q} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, ..., \boldsymbol{q}_m\}$, with $\boldsymbol{q}_i$ referring to the representation vector of label l_i.
- a set of branch vectors $\mathcal{B} = B_1 \cup B_2 \cup ... \cup B_n$, where $B_i = \{\boldsymbol{b}_{i,1}, \boldsymbol{b}_{i,2}, ..., \boldsymbol{b}_{i,k_i}\}$ is the set of k_i branch vectors of node v_i. $k_i \leq K$.
- a set of root vectors $\mathcal{R} = \{\boldsymbol{r}_1, \boldsymbol{r}_2, ..., \boldsymbol{r}_n\}$, with $\boldsymbol{r}_i$ being the root vector of node v_i.

4 Method

4.1 The Poincaré Ball

Let $\|\cdot\|$ denotes the Euclidean L^2 norm. A Poincaré ball with dimension d and radius 1 can be defined as the Riemannian manifold: $\mathcal{P}^{d,1} := \{\boldsymbol{x} \in \mathbb{R}^d \mid \|\boldsymbol{x}\|^2 < 1\}$, $g_{\boldsymbol{x}} = \lambda_{\boldsymbol{x}}^2 \boldsymbol{I}_d$, where g_x is the Riemannian metric tensor, $\lambda_{\boldsymbol{x}} = 2/(1 - \|\boldsymbol{x}\|^2)$ and $\boldsymbol{I}_d$ is the identity matrix. The distance between two points $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{P}^{d,1}$ can be computed as:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \operatorname{arcosh}\left(1 + 2\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{(1 - \|\boldsymbol{x}\|^2)(1 - \|\boldsymbol{y}\|^2)}\right). \quad (1)$$

With $\|\boldsymbol{x}-\boldsymbol{y}\|$ fixed, the distance $d(\boldsymbol{x},\boldsymbol{y})$ grows rapidly to infinity when $\|\boldsymbol{x}\|$ approaching the open boundary 1, similar to a tree that expands exponentially with the increase of the tree depth. [Nickel and Kiela, 2017] first used the Poincaré ball to embed hierarchical data.

4.2 Multi-Vector Embedding

The multi-vector embedding (MVE) procedure in HIME is to learn a representation vector for each label and an adaptive number of branch vectors for each node to fit its separate label sets. Inspired by the K-means algorithm, which clusters data points by assigning them to their closest centroids, we view the branch vectors as the centroids and the labels as the points to be clustered. The main difference is that the points are fixed in the clustering task, while the label vectors can be dynamically optimized in MVE.

Here we discuss the adaptiveness of MVE by examining the K-means algorithm. In the K-means, though the number K of the centroids is predetermined, the final number of clusters could be less than K if no data points are assigned to some centroids. It will happen when a poor centroid initialization is made or the point number has the same or close order of magnitude to K. The later situation indicates that the number of clusters can be adaptive to the distribution of a relatively small number of points, which is often the case in preserving about a dozen of labels by using several branch vectors. We present the details of MVE as follows.

The Mechanism of Multi-Vector Embedding

We first allocate K branch vectors $\boldsymbol{b}_{i,1}, \boldsymbol{b}_{i,2}, ..., \boldsymbol{b}_{i,K}$ for each node v_i, and initialize all the branch vectors and the label vectors close to the origin of the Poincaré ball. We define the node-label distance between node v_i and label l_j as the shortest Poincaré distance from all the branch vectors $\{\boldsymbol{b}_{i,1}, \boldsymbol{b}_{i,2}, ..., \boldsymbol{b}_{i,K}\}$ to the label vector $\boldsymbol{q}_j$:

$$D(v_i, l_j) = \min_{a=1,2,...K} d(\boldsymbol{b}_{i,a}, \boldsymbol{q}_j). \tag{2}$$

Parallel computation of this node-label distance can be easily implemented for a batch of node-label pairs. We then optimize the branch vectors and the label vectors by the following objective function with negative sampling:

$$O_{nl} = \sum_{(v_i,l_j)\in\mathcal{A}} \log\sigma(-D(v_i,l_j)) + \sum_{(v_i,l_j)\in\mathcal{A}_{neg}} \log\sigma(D(v_i,l_j)) \tag{3}$$

where $\mathcal{A}_{neg}$ contains negatively sampled node-label links. It should be pointed out that the positive set $\mathcal{A}$ can also include potential node-label links by performing random walks on meta-path 'Node-Node-Label' based on the dataset-specific assumption that connected nodes will have similar labels.

By maximizing O_{nl}, the negative samples serve as a strong force to push the branch vectors and the label vectors away from each other and the origin. By contrast, the positive samples has the clustering effect. Given a node-label pair (v_i, l_j), suppose $\boldsymbol{b}_{i,1}$ is the branch vector closest to $\boldsymbol{q}_j$ among all v_i's branch vectors, and in this way, $D(v_i,l_j) = d(\boldsymbol{b}_{i,1}, \boldsymbol{q}_j)$. During the back propagation process, only the gradients of $\boldsymbol{b}_{i,1}$ and $\boldsymbol{q}_j$ are updated by (v_i, l_j). Therefore, if (v_i, l_j) is a positive sample, $D(v_i, l_j)$ will be decreased, which means that both $\boldsymbol{b}_{i,1}$ and $\boldsymbol{q}_j$ will be updated closer. It can be interpreted in a way that given a node, a label is always assigned to the closest branch vector of the node, while a branch vector fits a cluster of labels, with all the vectors optimized dynamically.

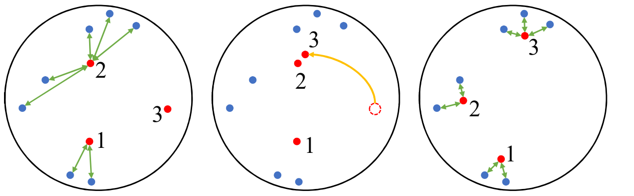

Figure 2: An illustration of the LRU policy. A node's three red branch vectors are allocated to fit the node's seven blue labels, with the third branch vector being inactive. After the vector replacement, the third branch vector attracts three labels from the busiest one, and finally each is assigned a small set of labels.

Algorithm 1 The LRU policy.

```
1: for $i$ in $n$ nodes do
2:     get $hit_{min}, id_{min} = \min(\mathbf{H}_{i,1}, \mathbf{H}_{i,2}, ..., \mathbf{H}_{i,K})$
3:     get $hit_{max}, id_{max} = \max(\mathbf{H}_{i,1}, \mathbf{H}_{i,2}, ..., \mathbf{H}_{i,K})$
4:     if $hit_{min} == 0$ then
5:         sample $\boldsymbol{\epsilon}$ from normal distribution $N(\mathbf{0}, 10^{-6})$
6:         $\boldsymbol{b}_{i,id_{min}} \leftarrow \exp_{\boldsymbol{b}_{i,id_{max}}}(\boldsymbol{\epsilon})$
7:     end if
8: end for
```

Tracing the Active Branch Vectors

We call a node's branch vector 'active' if at least one of the node's labels is assigned to it, otherwise we call it 'inactive'. The active branch vectors are those we want in the end. We maintain a hit matrix $\mathbf{H}^{n\times K}$, where $\mathbf{H}_{i,a}$ records the number of labels that $\boldsymbol{b}_{i,a}$ fits at the end of every epoch. Therefore, $\sum_{a=1}^{K} \mathbf{H}_{i,a} = \|A_i\|$, where A_i is the label set of v_i. $\mathbf{H}$ is set to $\mathbf{0}$ at the beginning of every epoch, and it can be simply maintained by the procedure computing the node-label distance. Given a positive pair (v_i, l_j), the procedure remembers the index id of the branch vector closest to $\boldsymbol{q}_j$ when computing the node-label distance, and plus $\mathbf{H}_{i,id}$ by 1 to record a hit. At the end of the MVE, we pick the active branch vectors according to their non-zero hit value in the last epoch.

The LRU Load Balancing Policy

Since all the vectors are dynamically updated, sometimes unbalanced loads are shared among a node's branch vectors. In figure 2, the hit values of the three red branch vectors are 2, 5 and 0 at the beginning. MVE almost degenerates into single vector embedding since the busiest branch vector fits a majority of labels. We use a least-recently-used (LRU) policy to balance the loads among the branch vectors.

The main idea is to split the most active vector into two vectors so that each is assigned a smaller set of labels. The two vectors are then optimized in the following epochs to fit its label clusters. With $\mathbf{H}$ providing the hit values, the 'splitting' can be achieved by replacing an inactive branch vector (LRU vector) close to the most active branch vector with a small random margin between them, so as to share the burden of the most active vector, as shown in figure 2. Algorithm 1 describes the LRU policy, where $\exp_{\boldsymbol{x}}(\boldsymbol{y})$ updates $\boldsymbol{x}$ with a step $\boldsymbol{y}$ in the Poincaré ball (see appendix[1]).

4.3 Refining the Label Hierarchy

MVE can roughly generate a hierarchical embedding of the taxonomy in a way that a high-level label having more nodes will be placed near the origin so as to minimize the overall Poincaré distance to its nodes, while low level labels with few nodes will be embedded near the boundary due to the effect of negative sampling. But with the ground-truth information given by $\mathcal{T}$, a more accurate embedding of the label hierarchy could be further achieved.

One thought is to simply use the Poincaré distance to reflect the relationships between two labels. However, though the Poincaré distance can reflect the proximity, it fails to reflect the partial order among the labels since the distance is symmetric for the two input vectors. Therefore, an asymmetric score function is required for directed parent-child edges.

Given two labels l_i and l_j, we call l_i 'is_a' l_j if $l_i \prec l_j$, meaning that there is a directed DAG path from l_j to l_i. We define the asymmetric score function of relation 'is_a' as:

$$S_{is\_a}(l_i, l_j) = -(1 + \alpha(\|\boldsymbol{q}_j\| - \|\boldsymbol{q}_i\|))d(\boldsymbol{q}_i, \boldsymbol{q}_j) \tag{4}$$

where α is a positive hyper-parameter set to 1. If $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$ are close in the Poincaré ball and $\boldsymbol{q}_j$ is closer to the origin than $\boldsymbol{q}_i$, then a high score $S_{is\_a}(l_i, l_j)$ will indicate the probability of $l_i \prec l_j$, particularly, l_i being the child of l_j. This score function is first introduced by [Nickel and Kiela, 2017] in the experimental evaluation, and here we apply it in the refinement of the label vectors:

$$O_{ll} = \sum_{(l_i,l_j)\in\mathcal{U}} \log\sigma(S_{is\_a}(l_i,l_j)) + \sum_{(l_i,l_j)\in\mathcal{U}_{neg}} \log\sigma(-S_{is\_a}(l_i,l_j)) \tag{5}$$

Where $\mathcal{U}$ is the parent-child edge set and $\mathcal{U}_{neg}$ is the set of negative edges sampled from $\{(l_i, l_j)|l_i \not\prec l_j\}$. By maximizing O_{ll}, the label vectors will be refined according to both the proximity and the partial order. The refinement of the label vectors is performed together with MVE.

4.4 Learning the Root Vector

A single root vector is learned for each node based on its active branch vectors to preserve the network proximity. This procedure is after MVE, since the active branch vectors of a node are not determined until the end of MVE.

Specifically, we use the hyperbolic aggregation over the branch vectors to represent $\boldsymbol{r}_i$. The hyperbolic aggregation is performed by the following steps: (1) Transform the Poincaré branch vectors into the Euclidean tangent space of the origin O using $\log_{\mathbf{o}}(\cdot)$ (see appendix). (2) Perform the aggregation in the tangent space. (3) Transform the result back to the Poincaré ball using $\exp_{\mathbf{o}}(\cdot)$. Formally,

$$\boldsymbol{r}_i = \exp_{\mathbf{o}}(\sum_{j=1}^{k_i} w_{ij} \log_{\mathbf{o}}(\boldsymbol{b}_{i,j}) + \boldsymbol{c}_i) \tag{6}$$

$$w_{ij} = \mathrm{softmax}_{j\in\{1,2,..,k_i\}}(p_{ij}) \tag{7}$$

[1]Code and Appendix: https://github.com/YueFan1014/HIME.

Algorithm 2 The learning process of HIME.

```
Input: $\mathcal{N}$, $\mathcal{T}$, $\mathcal{A}$, branch limit $K$, epoch $g$, LRU period $t$
Output: root vectors $\mathcal{R}$, branch vectors $\mathcal{B}$, label vectors $\mathcal{Q}$
 1: initialize $\mathcal{Q}$, $nK$ branch vectors $\mathcal{B}_{all}$, and $\mathbf{H}$
 2: for $epoch$ in $\{1, 2, ..., g\}$ do
 3:     $\mathbf{H} \leftarrow \mathbf{0}$
 4:     update $\mathcal{Q}$ by $O_{ll}$
 5:     update $\mathcal{Q}$ and $\mathcal{B}_{all}$ by $O_{nl}$, maintain $\mathbf{H}$
 6:     if $epoch$ mod $t == 0$ then
 7:         update $\mathcal{B}_{all}$ by the LRU with reference to $\mathbf{H}$
 8:     end if
 9: end for
10: obtain the active $\mathcal{B}$ from $\mathcal{B}_{all}$ according to $\mathbf{H}$
11: initialize $p$ and $\mathbf{c}$ to $\mathbf{0}$
12: for $epoch$ in $\{1, 2, ..., g\}$ do
13:     update $\mathcal{R}$ by optimizing $p$ and $\mathbf{c}$ through $O_{nn}$
14: end for
15: output $\mathcal{R}$, $\mathcal{B}$, $\mathcal{Q}$
```

where p and $\boldsymbol{c}$ are parameters that determine $\boldsymbol{r}$. Given a node v_i, since the weights of its branch vectors are all positive and satisfy $\sum_{j=1}^{k_i} w_{i,j} = 1$, this aggregation reflects the semantic meaning: the contributions of different label sets to the node-node proximity. Besides, the 'residual vector' $\boldsymbol{c}_i$ endows $\boldsymbol{r}_i$ with more flexibility in preserving the node-node proximity, while its influence on $\boldsymbol{r}_i$ should be minimized in order to fully extract the information learned by the branch vectors.

We initialize the parameters p and residual vectors $\boldsymbol{c}$ to $\mathbf{0}$. We optimize p and $\boldsymbol{c}$ by the following node-node objective:

$$O_{nn} = \sum_{e_{i,j}\in\mathcal{E}} \log\sigma(-d(\boldsymbol{r}_i, \boldsymbol{r}_j)) + \sum_{e_{i,j}\in\mathcal{E}_{neg}} \log\sigma(d(\boldsymbol{r}_i, \boldsymbol{r}_j)) - \lambda\sum_{i=1}^{n} \|\boldsymbol{c}_i\|^2 \tag{8}$$

where λ is a positive hyper-parameter which we set to 1, and $\mathcal{E}_{neg}$ is the negative sample set of the network edges. The norms of all the residual vectors are minimized by the third term in O_{nn} in order to maximize the contribution of the branch vectors to the root vectors.

4.5 The Whole Learning Process of HIME

Algorithm 2 shows the whole learning process of HIME. The space complexity of HIME is $O(K|\mathcal{V}| + |\mathcal{L}|)$, which is the space occupied by $\mathcal{R}$, $\mathcal{B}$ and $\mathcal{Q}$. Given the negative sampling number s, $(1+s)|\mathcal{U}|$ label-label pairs, $(1+s)|\mathcal{A}|$ node-label pairs and $(1+s)|\mathcal{E}|$ node-node pairs are passed to the model during each epoch. The extra cost of our method lies in the node-label distance computation and the hyperbolic aggregation, which are both $O(K)$. Therefore the total time complexity of learning for g epochs is $O(gs|\mathcal{U}| + gsK|\mathcal{A}| + gsK|\mathcal{E}|)$, which grows linearly with the increase of the branch limit K.

5 Experiments

5.1 Datasets

We evaluate HIME on three datasets from different domains.

	Network		Taxonomy			Assignment
Dataset	**nodes**	**links**	**labels**	**edges**	**levels**	**N-L links**
Protein_GO	13840	348658	4184	6848	15	205629
Gene_Pathway	5593	30290	241	288	9	28640
DBLP_ACM	12379	12164	268	268	4	50402

Table 1: The statistics of the three datasets.

Protein-GO. A human PPI network [Szklarczyk *et al.*, 2021] with the Cellular Components domain of Gene Ontology [Ashburner *et al.*, 2000] being the taxonomy. The GO terms of proteins are provided by GOA database. [Huntley *et al.*, 2015].

Gene-Pathway. A human gene regulatory network [Liu *et al.*, 2015] depicts regulations among genes, with each gene associated with several biological pathways given by CTD [Davis *et al.*, 2020]. All the pathways are organized as a pathway ontology [Petri *et al.*, 2014].

DBLP-ACM. We extract a dense subset of the DBLP co-authorship network with ACM key word taxonomy. A key word reflects an author's research interests.

Table 1 presents the statistics of the datasets. We split the node-node links with the ratio of 7:3 and the node-label links with the ratio of 9:1 for training and testing.

5.2 Methods for Comparison

In the experiments, **HIME_K** is the original version of HIME described in the paper, with the branch limit K being 2, 4 and 8. Besides, we include 8 existing methods for comparison and 2 variations of HIME for ablation studies.

We run **Node2Vec** [Grover and Leskovec, 2016] and **GraphGAN** [Wang *et al.*, 2018] dealing with plain networks by viewing labels as plain nodes and label-related edges as plain links. **GraphSAGE** [Hamilton *et al.*, 2017] and **PTE** [Tang *et al.*, 2015] are methods on heterogenous networks, while **JOIE** [Hao *et al.*, 2019] and **TaxoGAN** [Yang *et al.*, 2020] are designed for networks with taxonomies. **Poincaré** [Nickel and Kiela, 2017] and **Lorentz** [Nickel and Kiela, 2018] are two hyperbolic embedding methods.

For ablation study, **EUHIME** refers to the Euclidean version of HIME by replacing the hyperbolic distance function with the inner product score function, which is adopted by most existing methods. The 'free-root' version **RHIME** learns a node's root vector independently from the branch vectors, taking no label information into account.

In the experiments, we set the embedding dimensions of all methods to 256. The branch vector dimensions are 128, 64, 32 for HIME_2, HIME_4 and HIME_8 respectively so as to ensure that a node's total dimension is no greater than 256. All methods are tuned to the best and trained for 100 epochs.

5.3 Evaluation

Proximity search, hierarchical classification and link prediction are conducted to evaluate the comprehensive performances of the methods on the three datasets.

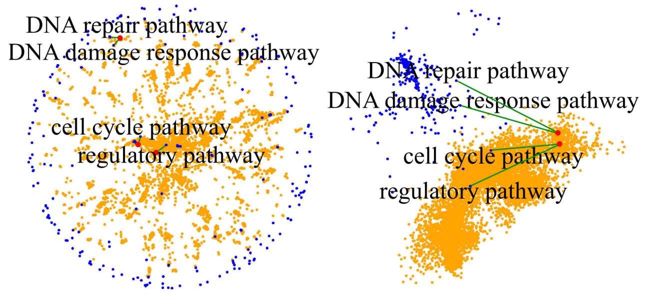

Figure 3: The 2D embeddings of HIME_4 and EUHIME_4 on Gene-Pathway. Blue points are labels and the orange points are the active branch vectors of all nodes. The red points are the active branch vectors of gene OR7G3 fitting the displayed pathways.

Proximity Search

Label Search (LS). Given a node, label search is to correctly return its seen or potential labels. The method ranks all the labels according to the node-label scores. The mean rank MR of the ground-truth labels is calculated for the node, which is averaged over all nodes to obtain $\overline{MR}$. We normalize it to $1 - \overline{MR}/|\mathcal{L}|$ as the metric.

Node Search (NS). Node search aims at recalling the nodes that a given label describes. Specifically, for a label, the method ranks all the nodes according to the node-label scores. $AUPRC$ is then computed for the ranking. The average $\overline{AUPRC}$ is calculated to represent the average performance on all labels.

Child Search (CS). Child Search evaluates a method's capability of preserving hierarchical label relationships. Given a parent label, the method ranks all other labels according to their 'is_a' scores. A mean rank MR of the label's children is then calculated. The MR for all parent labels are averaged to obtain $\overline{MR}$. $1 - \overline{MR}/|\mathcal{L}|$ is used to evaluate this task.

The LS, NS and CS columns in table 2 show the performance of all methods on proximity search.

On the LS task, HIME and EUHIME are the two best methods due to their multi-vector characteristics: each node has multiple branch vectors to minimize its distances to all its labels. Besides, the label vectors hierarchically organized in the hyperbolic space decrease the possibility of underfitting when searching a node's labels, which is proven by the better results of Poincaré and Lorentz compared with their Euclidean counterparts.

On the NS task, HIME significantly outperforms all other methods. Specifically, it outperforms Poincaré by 251%, 76% and 132% on the three datasets. Besides, there exists a positive correlation between HIME's performance and K, since the node-label distances of positive pairs will decrease when the underfitting problem is alleviated by large K, as illustrated by the small node-label distances between gene OR7G3 and its pathways in the Poincaré ball shown in figure 3. Furthermore, all the branch vectors are uniformly distributed in the whole Poincaré ball, reducing the density of branch vectors around a given label and leading to a more accurate node search near the label. By contrast, the branch

	Protein-GO				Gene-Pathway				DBLP-ACM			
Method	**LS**	**NS**	**CS**	**HC**	**LS**	**NS**	**CS**	**HC**	**LS**	**NS**	**CS**	**HC**
Node2Vec	85.29	18.32	72.93	10.31	79.81	47.51	73.27	23.76	59.09	12.57	71.38	18.32
GraphGAN	75.01	25.58	70.33	11.43	62.82	28.18	78.23	20.73	69.36	33.15	71.29	13.01
GraphSAGE	71.58	23.69	73.13	8.70	49.11	15.64	76.77	21.41	77.03	26.33	69.59	36.21
PTE	55.71	19.53	63.65	8.64	47.09	34.36	59.69	28.05	49.51	13.07	46.00	12.91
JOIE	87.81	28.46	79.87	14.84	63.10	32.50	75.74	29.10	67.10	25.34	70.00	17.83
TaxoGAN	43.55	4.55	51.73	**61.31**	58.23	6.10	50.17	41.15	53.09	4.90	48.95	43.38
Poincare	92.54	22.25	88.88	18.07	86.23	54.96	81.77	22.31	82.04	39.12	69.70	25.19
Lorentz	93.07	25.75	**89.29**	22.88	89.22	52.42	82.13	19.50	85.18	41.10	71.01	29.89
HIME_2	97.70	48.48	**90.19**	53.89	95.85	88.18	**85.65**	37.14	96.05	85.03	81.90	51.48
HIME_4	97.86	**70.55**	88.94	47.88	**96.07**	**93.52**	85.44	**55.84**	**96.92**	**89.19**	**82.18**	55.42
HIME_8	**98.02**	**78.17**	89.08	51.77	**95.91**	**96.85**	**85.93**	55.02	**96.93**	**90.98**	**82.07**	**57.73**
EUHIME_8	**98.56**	59.90	86.58	**60.84**	94.24	22.31	78.11	**69.06**	94.09	23.89	53.08	**62.02**

Table 2: The experiments on taxonomy-related tasks. All results are shown in percentage.

	Protein-GO		Gene-Pathway		DBLP-ACM	
Method	**PRC**	**ROC**	**PRC**	**ROC**	**PRC**	**ROC**
Node2Vec	62.65	79.26	36.13	66.19	53.28	69.96
GraphGAN	74.11	85.64	41.58	74.96	62.66	74.47
GraphSAGE	48.60	72.33	27.03	56.09	35.17	60.37
PTE	69.87	83.16	33.03	68.03	41.98	68.69
JOIE	70.44	83.28	35.10	65.99	42.49	66.30
TaxoGAN	73.97	85.56	40.62	73.63	63.19	73.26
Poincare	72.18	85.13	39.84	73.46	60.43	74.32
Lorentz	74.19	85.53	36.17	69.99	58.31	73.10
HIME_2	80.88	88.95	**45.73**	75.70	68.80	78.23
HIME_4	**82.93**	**89.10**	45.68	**76.88**	**70.79**	**79.16**
HIME_8	**81.99**	**89.13**	**47.54**	**77.08**	**71.01**	**79.04**
EUHIME_8	79.22	88.65	46.16	76.75	66.37	78.22
RHIME_8	70.22	83.65	39.25	74.39	63.10	75.80

Table 3: The experiments on node-node link prediction.

vectors and the labels in EUHIME are embedded into two separate groups due to its score function and the negative sampling, making it hard for EUHIME to correctly search the label's true branch vectors from a crowd of wrong ones.

On the CS task, HIME and Lorentz are the two best methods, followed by Poincaré and JOIE. These methods preserve the parent-child relationships by either defining an asymmetric score for the directed relationships or generating a more hierarchical embedding in hyperbolic spaces. Compared with HIME, EUHIME is less capable of generating an accurate taxonomy in Euclidean space, especially on DBLP-ACM with dense node-label links which dilute the effect of its symmetric label refinement on the label vectors.

Hierarchical Classification

Given a node, Hierarchical Classification (HC) is to predict a correct label path with the knowledge of the taxonomy. Starting from the root label, if a label is correctly predicted from its siblings, then the classification will continue to its child labels, otherwise it will stop. Acc is computed for each node as the ratio of the accurate predicted labels to the whole label-path. $\overline{Acc}$ evaluates the performance on all nodes.

On the HC task, EUHIME, HIME and TaxoGAN are the three leading methods.The good performance of TaxoGAN on HC is mainly due to the fact that it is specialized in predicting a node's label from the label's siblings by embedding them into a separate space. This ensures a more precise embedding but also sacrifices the universality of the vectors, which accounts for its notably poor performance on NS. Surprisingly, EUHIME is more capable than HIME of predicting a correct label path for a node, which can be explained by its inner-product score function: a branch vector can simultaneously increase the inner-product scores of all the labels in a path by increasing its norm or decreasing the angle to the label's directions. However, EUHIME also fails on the NS tasks as discussed before. Besides, its embedding provides less visual information compared with HIME that optimizes vectors according to their space distances.

Link Prediction

Link prediction tests a method's ability to predict potential node-node links. A method predicts unseen links from the negative ones by ranking the scores of node pairs. $AUPRC$ and $AUROC$ are calculated to evaluate the ranking.

Table 3 displays the results on link prediction. HIME has the highest $AUPRC$ and $AUROC$ compared with other methods, with EUHIME being the second and GraphGAN the third. RHIME has a relatively poor performance compared with HIME, indicating the effectiveness of using label information in predicting node-node links. Although the node vectors in other methods also contain label information, the abundant node-label links actually impose burdens on the node vectors other than providing information for predicting unseen node-node links, while these burdens are taken off by the multiple branch vectors in HIME.

6 Conclusions

The extensive experiments show the outstanding performance of HIME compared with existing methods. The ablation studies reveal the effectiveness of the Poincare ball in preserving the hierarchical taxonomy and the usefulness of branch-dependent root vectors in preserving the network proximity. HIME can be easily extended to diverse manifolds by equipping different distance functions. Techniques used by traditional methods such as random walks on meta-paths can be also applied by HIME.

Acknowledgments

We acknowledge the support by the High-Performance Computing Platform of Peking University, Beijing, China.

References

[Ashburner *et al.*, 2000] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

[Davis *et al.*, 2020] Allan Peter Davis, Cynthia J Grondin, Robin J Johnson, Daniela Sciaky, Jolene Wiegers, Thomas C Wiegers, and Carolyn J Mattingly. Comparative Toxicogenomics Database (CTD): update 2021. *Nucleic Acids Research*, 49(D1):D1138–D1143, 10 2020.

[Epasto and Perozzi, 2019] Alessandro Epasto and Bryan Perozzi. Is a single embedding enough? learning node representations that capture multiple social contexts. In *The World Wide Web Conference*, page 394–404. Association for Computing Machinery, 2019.

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

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[Hao *et al.*, 2019] Junheng Hao, Muhao Chen, Wenchao Yu, Yizhou Sun, and Wei Wang. Universal representation learning of knowledge bases by jointly embedding instances and ontological concepts. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 1709–1719, 2019.

[Huntley *et al.*, 2015] Rachael P Huntley, Tony Sawford, Prudence Mutowo-Meullenet, Aleksandra Shypitsyna, Carlos Bonilla, Maria J Martin, and Claire O'Donovan. The goa database: gene ontology annotation updates for 2015. *Nucleic acids research*, 43(D1):D1057–D1063, 2015.

[Liu *et al.*, 2015] Zhi-Ping Liu, Canglin Wu, Hongyu Miao, and Hulin Wu. RegNetwork: an integrated database of transcriptional and post-transcriptional regulatory networks in human and mouse. *Database*, 2015, 09 2015. bav095.

[Liu *et al.*, 2019] Ninghao Liu, Qiaoyu Tan, Yuening Li, Hongxia Yang, Jingren Zhou, and Xia Hu. Is a single vector enough? exploring node polysemy for network embedding. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 932–940, 2019.

[Nickel and Kiela, 2017] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[Nickel and Kiela, 2018] Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3776–3785, 2018.

[Park *et al.*, 2020] Chanyoung Park, Carl Yang, Qi Zhu, Donghyun Kim, Hwanjo Yu, and Jiawei Han. Unsupervised differentiable multi-aspect network embedding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1435–1445, 2020.

[Petri *et al.*, 2014] Victoria Petri, Pushkala Jayaraman, Marek Tutaj, G. Thomas Hayman, Jennifer R. Smith, Jeff De Pons, Stanley JF Laulederkind, Timothy F. Lowry, Rajni Nigam, Shur-Jen Wang, Mary Shimoyama, Melinda R. Dwinell, Diane H. Munzenmaier, Elizabeth A. Worthey, and Howard J. Jacob. The pathway ontology – updates and applications. *Journal of Biomedical Semantics*, 5(1):7, Feb 2014.

[Ren *et al.*, 2016] Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1369–1378, 2016.

[Szklarczyk *et al.*, 2021] Damian Szklarczyk, Annika L Gable, Katerina C Nastou, David Lyon, Rebecca Kirsch, Sampo Pyysalo, Nadezhda T Doncheva, Marc Legeay, Tao Fang, Peer Bork, et al. The string database in 2021: customizable protein–protein networks, and functional characterization of user-uploaded gene/measurement sets. *Nucleic acids research*, 49(D1):D605–D612, 2021.

[Tang *et al.*, 2015] Jian Tang, Meng Qu, and Qiaozhu Mei. 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*, pages 1165–1174, 2015.

[Wang *et al.*, 2018] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[Wang *et al.*, 2019] Junshan Wang, Zhicong Lu, Guojia Song, Yue Fan, Lun Du, and Wei Lin. Tag2vec: Learning tag representations in tag networks. In *The World Wide Web Conference*, page 3314–3320, 2019.

[Yang *et al.*, 2020] Carl Yang, Jieyu Zhang, and Jiawei Han. Co-embedding network nodes and hierarchical labels with taxonomy based generative adversarial networks. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 721–730. IEEE, 2020.