Taxonomy Embedding in Poincaré Space: A General Exploration

Luyu Gao, Yunyi Zhang, Xinwei He
{luyugao2, yzhan238, xhe17}@illinois.edu

Abstract

Graph embedding is an important branch in Data Mining and Machine Learning, and most of recent studies are focused on preserving the hierarchical structure with less dimensions. One of such models, called Poincare Embedding, achieves the goal by using Poincare Ball model to embed hierarchical structure in hyperbolic space instead of traditionally used Euclidean space. However, Poincare Embedding suffers from two major problems: (1) performance drops as depth of nodes increases since nodes tend to lay at the boundary; (2) the embedding model is constrained with pre-constructed structures and cannot be easily extended. In this paper, we first raise several techniques to overcome the problem of low performance for deep nodes, such as using partial structure, adding regularization, and exploring sibling relations in the structure. Then we also extend the Poincare Embedding model by extracting information from text corpus and propose a joint embedding model with Poincare Embedding and Word2vec.

1. Introduction

In the recent year, many embedding techniques are proposed to learn representation of a set of data in latent space. Word embedding such as word2vec and GloVe are proposed to learn representation of word in corpus. Graph embedding techniques such as DeepWalk are proposed to learn about node similarity. Instead of learning similarity, another type of embedding aims at capturing the structural information in the data. As an example, taxonomy embedding aims at capturing hypernymy-hyponymy relations within a taxonomy. Recently, one paper proposed to use a hyperbolic space (Poincare space) to capture hierarchical information in a taxonomy structure, whose distinct distance metric enable it to capture hierarchy with low dimension vector. In our project, we carefully examine the property of Poincare space and the method proposed in the original paper. We discovered the strength, weakness and problems of the original method and provide a few methods to deal with the original problems.

2. Poincare Embedding and Its Problems

Since taxonomy is a hierarchical structure, its number of nodes grows exponentially as the height of tree increases. As a result, there is a fundamental limitation of many traditional embedding methods that linear embedding of graphs may require a prohibitively large dimensionality to model certain types of relations. To solve the problem, a recent model, called Poincare model, changed the embedding space to hyperbolic space and was shown to be able to capture the hierarchical structure with relatively less dimensions.

2.1. Poincare Embedding

Poincare embedding uses hyperbolic space to represent the hierarchical structure. Hyperbolic space has a constant negative curvature, and, intuitively, it can be thought of as a continuous version of trees. Therefore, it can be naturally equipped to model hierarchical structure, and the number of dimensions needed drops significantly.

Poincare embedding chooses Poincare ball model among several existing models for hyperbolic space, since it is well-suited for gradient-based optimization.
The distance function in the Poincare ball model between two vectors \( u \) and \( v \) is given by:

\[
d(u, v) = \text{arccosh} \left( 1 + 2 \frac{\|u - v\|^2}{(1 - \|u\|^2)(1 - \|v\|^2)} \right),
\]

where \( \| \cdot \| \) denotes the Euclidean norm. It can be seen from the distance function, by placing root node of the hierarchical structure near the origin, it will have relatively small distances to all other nodes, and leaves nodes can be placed close to the boundary as the distance grows rapidly between points with norms close to one. In the paper, Wordnet noun phrases are used as training data, and due to the special property of the Poincare ball model, the transitivity closure is constructed for the Wordnet hypernymy hierarchy before fed into the training model. Figure 1 is a simple example of transitivity closure, where black arrows represent the original relations given by Wordnet and red arrows the relations constructed by transitivity closure.

![Figure 1. Transitivity Closure](image)

Poincare embedding uses the softmax function in its loss function:

\[
\mathcal{L}(\Theta) = \sum_{(u, v) \in \mathcal{D}} \log \frac{e^{-d(u, v)}}{\sum_{v' \in \mathcal{N}(u)} e^{-d(u, v')}} ,
\]

where \( \mathcal{D} = \{(u, v)\} \) is the set of observed hypernymy relation pairs and \( \mathcal{N} = v | (u, v) \notin \mathcal{D} \cup \{u\} \) is the set of negative examples for \( u \) (including \( u \)). During the training process, \( k \) negative samples (\( k = 10 \) in the original paper’s setup) would be randomly selected based on a multinomial distribution proportional to the number of times each node appears in the data set. The gradient is given by:

\[
\frac{\partial \mathcal{L}}{\partial u} = \frac{\partial d(u, v)}{\partial u} - \frac{1}{Z} \left( e^{-d(u, v)} \frac{\partial d(u, v)}{\partial u} \right)
\]

\[
\frac{\partial \mathcal{L}}{\partial v} = \frac{\partial d(u, v)}{\partial v} - \frac{1}{Z} e^{-d(u, v)} \frac{\partial d(u, v)}{\partial v},
\]

\[
\frac{\partial \mathcal{L}}{\partial v_i} = -\frac{1}{Z} e^{-d(u, v_i)} \frac{\partial d(u, v_i)}{\partial v_i},
\]

where we let \( Z = e^{-d(u, v)} + \sum_{i=1}^k e^{-d(u, v_i)} \). The partial derivative of the distance function is given by:

\[
\frac{\partial d(u, v)}{\partial u} = \frac{4}{\alpha \beta \sqrt{\gamma^2 - 1}} \left( 1 + \frac{\|u - v\|^2}{\alpha} \right) (u - v),
\]

\[
\frac{\partial d(u, v)}{\partial v} = \frac{4}{\alpha \beta \sqrt{\gamma^2 - 1}} \left( 1 + \frac{\|u - v\|^2}{\beta} \right) (v - u),
\]

where \( \alpha = 1 - \|u\|, \beta = 1 - \|v\|, \) and \( \gamma = 1 + \frac{2}{\alpha \beta} \|u - v\|^2 \).

### 2.2. Problems with Poincare Embedding

Although Poincare embedding can represent hierarchical structure with relatively less dimensions, it suffers from two major problems that largely limit its performance. The first problem is that most nodes lay at the boundary of the Poincare ball model, indicating that Poincare embedding cannot capture enough structural information for nodes that further form root node. Figure 2 is a visualized embedding result for the mammal subtree of Wordnet noun phrases provided in the original paper. Although the mammal tree’s depth is 9, it can be seen from the plot that, nodes with distance larger than 3 from the root node are all pushed to the boundary. Figure 3 shows the nDCG@10 measure for nodes with different depths, and it is clear that the performance drops significantly at depth = 4.
Another problem with Poincare Embedding is that only words that are contained in the pre-constructed taxonomy structure can be embedded by the model. The model largely depends on the quality of the existing taxonomy and cannot be easily extended to represent more words.

3. Taxonomy Induction Techniques

As Poincare space embedding of taxonomy structure is recently proposed new idea, we expect that by carefully choosing the training data setting and the training objective, it is possible to improve the performance of Poincare embedding. In this section, we will be introducing our methods to enhance Poincare embedding.

3.1. A naive approach: Partial link

As discussed before, original Poincare embedding suffers a performance drop as the the depth of the hypernymy nodes increase. In the original method, embedding training is performed on the transitivity closure of the original structure. In particular, it consists all the possible hypernym-hyponym tuples. We believe this is one of the reason for the performance drop. As all the links in the structure become explicit, nodes that are closer to the root are presented in more training tuples compared to the deeper ones, since a node will be connected to all the sub-nodes. To deal with this issue, we first introduced a naive method of direct data manipulation. For all the tuples in the transitivity closure, our model will first filter out all the tuples such the distance of the two entry nodes are greater or equal to $d$, where $d$ is a hyper-parameter parameter that can be tuned for particular model. Moreover, we believe this method can be extended to having different weight for different distance $d$, but in general the complexity of training of tuning will be greatly increased.

3.2. Regularization

In addition to data manipulation, we also found out that introducing loss to the problem can helped with the dropping of performance across layers. In general, it is pretty natural to use regularization in a machine learning model and in the past years, regularization based on different norm such as L1 and L2 setup are developed. To train the Poincare, we experimented with all such norms and as expected they do not help in the Poincare space. In contrary to popular regularization function that are space invariant, in the Poincare space, the distance metric between two points are space dependent, which is essential to its advantage for taxonomy embedding.

To tackle with this problem, we experimented with new formulations of regularization. Here we will introduce one that work in this particular problem. To calculate the regularization term, for a vector, take the points corresponds to the vector tip and calculate its distance to the origin by the Poincare metric. The
Taxonomy Embedding in Poincaré Space: A General Exploration

distance between a point \( u \) and the origin is given by:

\[
d(u,0) = \operatorname{arcosh}(1 + 2 \frac{||u||^2}{1 - ||u||^2}).
\]

In order to globally impose this loss, as usually we incorporate it into our loss function to enforce it during each update.

New loss function and gradients are given by:

\[
\mathcal{L}_1(\Theta) = \sum_{(u,v) \in D} \left( \alpha \log \frac{e^{-d(u,v)}}{\sum_{v' \in N(u)} e^{-d(u,v')}} + (1 - \alpha)(d(u,0) + d(v,0)) \right),
\]

\[
\frac{\partial \mathcal{L}_1}{\partial u} = \alpha \frac{\partial \mathcal{L}}{\partial u} + (1 - \alpha) \frac{\partial d(u,0)}{\partial u},
\]

\[
\frac{\partial \mathcal{L}_1}{\partial v} = \alpha \frac{\partial \mathcal{L}}{\partial v} + (1 - \alpha) \frac{\partial d(v,0)}{\partial v},
\]

\[
\frac{\partial \mathcal{L}_1}{\partial \nu_i} = \alpha \frac{\partial \mathcal{L}}{\partial \nu_i}.
\]

3.3. Sibling information

Another problem we noticed with transitivity closure is that while it fails to capture the sibling information implicitly encoded in the taxonomy structure. In a taxonomy, in addition to all the edges that are represented explicitly in the taxonomy structure, we observe that the structure also implicitly captures all the sibling relations between words. Being a information of the taxonomy, siblings can also be important information for taxonomy embedding. Though in general all the Is-A information is able to imply the rest of the sibling relations, during training in update, method like stochastic gradient descent may not be able to capture such information.

In our work, we proposed to explicitly incorporate such sibling relations by directly modeling it. In particular, in our attempts, one method that works pretty well is to model all the siblings as Euclidean clusters.

For training, as with the regularization, we will direct add a new sibling term to the loss function and jointly train both Is-A and sibling relations with a complementary filter to balance them.

New loss function and its gradients are:

\[
\mathcal{L}_2(\Theta) = \alpha \sum_{(u,v) \in D} \log \frac{e^{-d(u,v)}}{\sum_{v' \in N(u)} e^{-d(u,v')}} + (1 - \alpha) \sum_{(u,v) \in S} \log \frac{e^{-||u-v||}}{\sum_{v' \in N^*_c}(u) e^{-||u-v'||}},
\]

\[
\frac{\partial \mathcal{L}_2}{\partial u} = \alpha \frac{\partial \mathcal{L}}{\partial u} + (1 - \alpha) \frac{\partial \|u - v\|}{\partial u} - \frac{1}{Z} \left( e^{-\|u-v\|} \frac{\partial \|u-v\|}{\partial u} \right),
\]

\[
\frac{\partial \mathcal{L}_2}{\partial v} = \alpha \frac{\partial \mathcal{L}}{\partial v} + (1 - \alpha) \frac{\partial \|u - v\|}{\partial v} - \frac{1}{Z} e^{-\|u-v\|} \frac{\partial \|u-v\|}{\partial v},
\]

\[
\frac{\partial \mathcal{L}_2}{\partial \nu_i} = \alpha \frac{\partial \mathcal{L}}{\partial \nu_i} - \frac{1}{Z} e^{-\|u-v\|} \frac{\partial \|u-v\|}{\partial \nu_i},
\]

4. Corpus-based Poincaré Embedding

The previous sessions shows that Poincaré embedding is extremely effective in capturing hyponym-hypernym relationships but unlike other embedding method, e.g. Glove, word2vec, which are capable of providing all words appearing in a corpus with a unique embedding that captures each word’s latent expression, poincare embedding is only able to capture hierachical realtions of a set hyponym-hypernym pairs.

It would be really helpful for hyponym-hypernym induction tasks if we could design a model that automatically "grows" our taxonomy tree by giving each candidate hyponym term an embedding and placing it under the correct hypernym when a set of candiate hyponym and related corpus is provided.

4.1. Problem Description

Input data are a set of existing hypernym relation pairs \( D \), a set of candidate hyponym terms \( S \) and a corpus \( V \) s.t. \( S \subset D \). Our goal is to place each hyponym term "under" the correct corresponding hypernym existing in \( S_c \).
4.2. Main idea

Since Word2vec shows strong ability to capture similarities among words and is capable of capturing words in similar context. As both "dog" and "cat" are pets and mammals, it is conceivable that by word embedding of corpus related to dog and cat, we are able to automatically obtain a word embedding in which embedding for dog and cat are close to each other. As a result, we plan to first provide a hyperbolic version of Word2vec model so that both embedding models are able to be consistent in terms of distance metric. Then we will combine Poincare and hyperbolic Word2vec in a joint learning scheme so that they are able to mutually enhance each other.

4.3. Construction of ”Hyperbolic Word2vec”

1. The original Word2vec loss function:

\[ L_{\text{original}}^{w2v} = \sum_{w \in V} \sum_{v \in C(w,l)} n_{v,C(w,l)} \log \sigma(w \cdot v^T) \]

\[ + k_{v \sim p_n(v')} \log \sigma(-w \cdot v') \]

where \( V \) is the given corpus, \( C(w,l) \) is the neighboring words of \( w \) with window size \( l \), \( n_{v,C(w,l)} \) is the number of times \( v \) appears in \( C(w,l) \), and \( \sigma(x) \) represents the sigmoid function.

2. (a) Dot product and Euclidean distance

\[ \|x - y\|_2^2 = (x - y)^T \cdot (x - y) = \|x\|_2^2 + \|y\|_2^2 - 2x^T \cdot y \]

\[ \Rightarrow w \cdot v^T \propto -\|w - v\|_2^2 \]

(b) Euclidean to hyperbolic

\[ -\|w - v\|_2^2 \rightarrow -d(w, v) \]

3. By relationships we derive above, we plan to replace dot product in original Word2vec directly by hyperbolic distance as shown below:

\[ L_{\text{hyper}}^{w2v} = \sum_{w \in V} \sum_{v \in C(w,l)} n_{v,C(w,l)} \log \sigma(-d(w, v)) \]

\[ + k_{v \sim p_n(v')} \log \sigma(d(w, v)) \]

4. The combined model is:

\[ L_{\text{joint}} = -\alpha \sum_{w \in V} \sum_{v \in C(w,l)} [\log e^{-d(w, v)} - \log \sum_{v' \in N_1(w)} e^{-d(w, v')}] \]

\[ + (1 - \alpha) \sum_{w \in V} \sum_{v \in C(w,l)} n_{v,C(w,l)} \log \sigma(-d(w, v)) \]

\[ + k_{v \sim p_n(v')} \log \sigma(d(w, v)) \]

5. Gradients of the hyperbolic Word2vec part:

\[ \frac{\partial L_{w2v}}{\partial w} = -\sum_{v \in C(w,l)} n_{v,C(w,l)} (1 - \sigma(-d(w, v))) \frac{\partial d(w, v)}{\partial w} \]

\[ + k \sum_{v' \in N_2(w)} n_{v',N_2(w)} p_n(v') (1 - \sigma(d(w, v'))) \frac{\partial d(w, v')}{\partial w} , \]

where \( N_2(w) \) is the set of negative samples of \( w \)

6. Riemannian Gradient Descent

After obtaining the above Euclidean gradients, we can transform them into the Riemannian gradients and do a projected Riemannian gradient descent:

\[ \theta_{t+1} = \text{proj} \left( \theta_t - \eta \left( \frac{1 - \|\theta_t\|^2}{4} \frac{\partial L}{\partial \theta_t} \right) \right), \]

for \( x \in \{u, v, v_1, \cdots, v_k\} \), where

\[ \text{proj}(\theta) = \begin{cases} \frac{\theta}{\|\theta\|} - \epsilon & \text{if } \|\theta\| > 1, \\ \theta & \text{otherwise.} \end{cases} \]

4.4. Implementation details

Similar to w2v model, we traverse each word sequentially in the corpus and, for each target word, we obtain its corresponding context word by keeping a fixed size window. If the target word does not appear in hyponym-hypernym pairs then we update its gradient model by following the hyperbolic word2vec model; otherwise, we update its gradient by calculating each corresponding gradient of Word2vec and Poincare model and obtain the final gradient by a linear combination.
4.5. Case Study

Here is an interesting case found by our model, the \{(canine, dog), (dog, basenji dog), (dog, leonberg dog), (dog, sausage dog)\} hierarchical structure belongs to the existing hyponym-hypernym pair sets and the we successfully incorporate "raccoon dog" to the existing taxonomy tree by joint learning of poincare and hyperbolic word2vec model on a corpus that include "raccoon dog".

4.6. Result

Although our corpus-based Poincare embedding model gives us some interesting result, it still does not perform well in most other the cases, below I summarize several potential reasons.

1. In the construction of our "hyperbolic word2vec model", the analysis that lead us to switch from "dot product measure" to "hyperbolic measure" is overly-simplified

2. The training procedure placed too much weight to word that does not appeared in the hyponym-hypernym pairs set compared to the words that do, as most of the word that appears in the corpus does not appear in hyponym-hypernym pairs set. As a result, the effect of Poincare embedding model is downplayed

3. Directly switching euclidean distance in word2vec model to hyperbolic distance may lead to unexpected phenomenon; it is not guaranteed that we could use exactly the same manner to analyze relationships of pre-trained word embedding in hyperbolic space.

5. Evaluation

In this section, we evaluate the performance of the improved Poincare Embedding on its capability of taxonomy reconstruction using both of the whole Wordnet noun phrases and the mammal subtree. We embed fully observed hypernym pairs and reconstruct the original taxonomy from the embedding result. Following two evaluation metrics are used:

**Normalized discounted cumulative gain**: We use all the hypernyms in the original taxonomy for each hyponym as the ground truth, and calculate the nDCG@10 for each hyponym according to its distance to all of other nodes. We use the normalized version DCG since each hyponym has different number of true hypernyms.

**Mean average precision**: For each observed relation \((u, v) \in D\), we rank the distance \(D(u, v)\) among all the negative examples \(\{d(u, v')|(u, v') \notin D\}\), and then calculate the mean rank precision for the whole data set.

5.1. Evaluation on Mammal Subtree

Table 1 shows the evaluation results for both dimension is 5 and 10. The first row is the performance of the original Poincare Embedding implemented by Gensim. It can be seen that Poincare Embedding with sibling information achieve the best result.

We also plot nDCG@10 for nodes with different depth for each of the improved Poincare Embedding. From the plots, We can also notice that different methods improve the performance of Poincare Embedding in different manners.

5.2. Entire Wordnet taxonomy

We did not have enough time to fully tune all the hyper-parameters, and Table 2 shows the best results we have got for each improved Poincare Embedding model on the whole Wordnet taxonomy structure.
Table 1. Result on Mammal Subtree

<table>
<thead>
<tr>
<th>Method</th>
<th>Dimension = 5</th>
<th>Dimension = 10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>nDCG@10</td>
<td>MAP</td>
</tr>
<tr>
<td>Poincare</td>
<td>0.39</td>
<td>0.32</td>
</tr>
<tr>
<td>Poincare w/ Partial Link</td>
<td>0.58</td>
<td>0.49</td>
</tr>
<tr>
<td>Poincare w/ Regularization</td>
<td>0.51</td>
<td>0.42</td>
</tr>
<tr>
<td>Poincare w/ Partial Link + Regularization</td>
<td>0.59</td>
<td>0.50</td>
</tr>
<tr>
<td>Poincare w/ Sibling</td>
<td>0.69</td>
<td>0.61</td>
</tr>
</tbody>
</table>

Table 2. Evaluation on Entire Wordnet Taxonomy

<table>
<thead>
<tr>
<th>Method</th>
<th>Dimension = 10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>nDCG@10</td>
</tr>
<tr>
<td>Poincare</td>
<td>0.44</td>
</tr>
<tr>
<td>Poincare w/ Partial Link</td>
<td>0.51</td>
</tr>
<tr>
<td>Poincare w/ Regularization</td>
<td>0.49</td>
</tr>
<tr>
<td>Poincare w/ Partial Link + Regularization</td>
<td>0.52</td>
</tr>
<tr>
<td>Poincare w/ Sibling</td>
<td>0.61</td>
</tr>
</tbody>
</table>

Figure 5. Poincare w/ Partial Link

Figure 7. Poincare w/ Partial Link + Regularization

Figure 6. Poincare w/ Regularization

Figure 8. Poincare w/ Sibling
6. Future Work

1. We can combine our methods in improving Poincare Taxonomy embedding. In the meantime, we should consider better ways to integrate sibling information into the embedding. In addition, in our experiment, we notice that the real global structure of the hierarchy have profound influence on the embedding performance and it is possible to develop methods that can adapt itself based on the global structure.

2. We should modify loss function of hyperbolic word2vec and find more appropriate ways in designing our joint learning algorithm, in order to improve the overall performance on more general domains.

3. We plan to further improve our model by incorporating more fine-grained relationship between terms in taxonomy tree. e.g. currently only sibling relationships in exploited in the enhanced poincare model, but we could also include ”generalized sibling relationships”(hyponyms with a common grandparents) in the model construction.

4. Exploit relationships between regularization of norm of the taxonomy embedding and the regularization of euclidean sibling distance of the hyponym terms in the taxonomy tree.

7. Reference


