Geometric Methods in Machine Learning

Poincaré Embeddings for Learning Hierarchical Representations

M. Nickel & D. Kiela (Facebook AI Research)

Samuel RITCHIE (samuel.ritchie@ensae.fr)
Elvire ROBLIN (elvire.roblin@ensae.fr)

May 19, 2018

Teacher : **Marco Cuturi**

Contents

Introduction **3**

1 Hyperbolic embeddings **4**
1.1 Hyperbolic space 4
1.2 The Poincaré ball-model 4
1.3 The optimization procedure 5
1.3.1 The Poincaré Ball as a Riemannian manifold structure 5
1.3.2 The adapted RSGD 5

2 Implementation **6**
2.1 Datasets used 6
2.2 Results 6

Conclusion **10**

References **10**

Introduction

Among the variety of approaches that exist in machine learning, representation learning (also called feature learning) has recently been widely applied to some specific datasets, such as text and graphs. The general idea behind representation learning is to allow a system to automatically discover the representations needed for feature detection or classification from raw data. This is of great interest as it eventually may allow us to replace manual feature engineering by an automatic task. One of the most famous subfield of representation learning is undoubtedly word embedding, where words or phrases from vocabulary dictionnaries are mapped to vectors in a context of Natural Language Processing, for example regarding sentiment analysis. The algorithm Word2Vec, developed by Google Inc. in 2013, was an important milestone in the litterature regarding text representation, following the famous quotation of the linguist John Rupert Firth that *"You shall know a word by the company it keeps"*. Similarly, algorithms such as Node2Vec have been widely applied in a matter of community detection and link prediction in social (or computational) networks.
The main objective of such embedding techniques is thus to organize symbolic objects (such as words in a text or nodes in a graph) in a way that their similarity is reflected by their distance in the embedding space. Even though these methods have for a long time been considered as the state-of-the-art in this context, they present a major drawback : modeling complex relations between words or in graphs inherently induce a large dimensionality of the embedding space, which causes obvious computational issues.

In their article "Poincaré Embeddings for Learning Hierarchical Representations" (2017), M. Nickel & D. Kiela consequently focus on finding a way to increase the representation capacity of embedding methods in order to model both large-scale and complex patterns in structural data. To do so, the authors limit themselves to datasets that may be organized according to a latent-hierarchy, which is often the case in complex datasets such as texts. The main innovation in their approach is that rather than computing embeddings in a classical Euclidian space, they do so in hyperbolic spaces. Hyperbolic geometry, that we will detail further on, has shown itself particularly adapted to embedding tree-based structures, thus fitting our latent hierarchical context. More specifically, authors use the Poincaré ball model, insofar as it enables usual gradient optimization to be performed.

After having introduced basic notions of hyperbolic spaces in a first part, we will detail the algorithm proposed by the authors on the Poincaré ball and resume their main findings. In a third part, we will apply this approach to the word 'music_genre' thanks to WordNet database, as well as on a Network of Jazz Musicians.

1 Hyperbolic embeddings

1.1 Hyperbolic space

The framework of Nickel & Kiela's studies in their article refers to non-Euclidian geometry, on which we will give some basic insights in order to better understand Poincaré Embeddings.
In opposition to Euclidian geometry, which consists in the intersection of metric and affine geometry, the non-Euclidian field appears when relaxing one of the postulates of the latter, i.e. respectively the metric requirement or the parallel postulate. Relaxing the parallel postulate (also called Euclid's fifth postulate) leads to hyperbolic and elliptic geometry, depending on the curvature considered. More precisely, a simple way to differentiate Euclidian from hyperbolic geometry is to consider two straight lines indefinitely extended in a two-dimensional plane that are both perpendicular to a third line : in Euclidian geometry, these two lines remain at constant distance and never cross (parallel postulate) while in Hyperbolic geometry, they "curve away" from each other.[1]

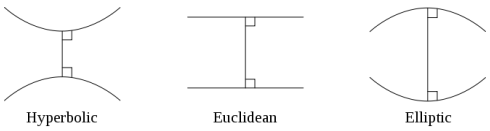
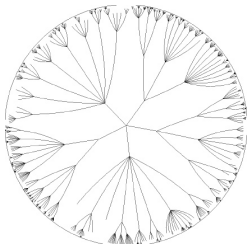


Figure 1: Differences between Euclidian and non-Euclidian geometry

Figure 2: Hyperbolic tree



To put in a nutshell, hyperbolic geometry can consequently be seen as a non-Euclidian setting for spaces of negative curvature. The reason hyperbolic geometry is of a great use for Poincaré embeddings is that they are particularly adapted to the visualization of large hierarchies such as trees, as showed by Lamping et al. in their article [2]. The original issue in efficiently visualizing trees comes from the fact that the number of nodes per level usually grows exponentially.[2] In Euclidian geometry, we often represent trees linearly in a plane, descending from parents to children, which is quickly difficult to interpretate. If we wanted to represent trees in circles, Euclidian geometry would also quickly come to its limits, insofar as circle length and disc area only grow respectively linearly and quadratically with the radius r. On the contrary, hyperbolic geometry has the nice property of having a disc area and circle length growing exponentially with the radius, enabling an efficient rendering of the exponential growth of childrens in trees.
The fact that hyperbolic space intrinsically has "more room" than Euclidean space (exponential in the radius rather than polynomial) is also extremely important regarding computational issues. Indeed, in the previous Euclidian setting, the only way to model both large-scale and complex patterns was to increase the dimensionality of the embedding space, which led to obvious problems of runtime and memory. The use of hyperbolic setting is thus an answer to this limitation, as well as a powerful tool regarding representation of the data.

1.2 The Poincaré ball-model

Once the hyperbolic framework set, the idea is to find an appropriate embedding that enables reflecting semantic similarity of words by their distance in the embedding space. A fundamental assumption of the authors to do so is to consider the existence of a latent hierarchy organizing words. We will discuss this assumption latter when describing the dataset used in our implementation. Moroever, even if in previous litterature, various authors have shown the usefulness of considering pairs as ordered, Nickel & Kiela do not make this assumption, i.e. orders in pairs do not matter.
The motivations in using hyperbolic geometry is two-fold :

- improving performances of the algorithm thanks to an appropriate structure
- gaining new information relationships thanks to a friendly representation in the embedding space

[1] While in elliptic geometry, they "curve toward" and intersect
[2] For a basic binary-tree, the maximum number of nodes at a level n is 2^n

Multiple models exist in hyperbolic space, among which the most famous ones are the Beltrami-Klein model and the two Poincaré models. Authors decide to use the Poincaré ball model because it is adapted to gradient-based optimization, which is an essential argument as we will see in the following subsection. The Poincaré ball is a generalization in n-dimensions of the Poincaré disk. The latter employs the interior of the unit circle, and lines are represented by arcs of circles that are orthogonal to the boundary circle, plus diameters of the boundary circle. A visual advantage of this model is that is preserves angles (which is why it is also called 'conformal disk').[3]
This definition easily generalizes to a d-dimensional unit ball $\mathcal{B}^d = \{x \in \mathbb{R}^d s.t. ||x|| < 1\}$ where $||\cdot||$ corresponds to the Euclidian norm. The Poincaré ball is equiped with the following metric :

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

1.3 The optimization procedure

1.3.1 The Poincaré Ball as a Riemannian manifold structure

M. Nickel and D. Kiela assimilate the Poincaré ball to a Riemannian manifold structure. A Riemannian manifold can be define by a pair (M,g), where M is a smooth manifold[4] and g is a Riemannian metric. More precisely, a Riemannian metric g on a smooth manifold M is a smoothly chosen inner product $g_x : T_xM \times T_xM \longrightarrow \mathbb{R}$ on each tangent spaces T_xM of M. In other words, for each $x \in M, g = g_x$ satisfies :

- $g(u,v) = g(v,u) \quad \forall u, v \in T_xM$;
- $g(u,u) \geq 0 \quad \forall u \in T_xM$;
- $g(u,u) = 0 \iff u = 0$.

A Riemann manifold carries the structure of a metric space whose distance function is the arclength of a minimizing path between 2 points. In this structure, a geodesic is define as a curve of minimum length joining sufficiently close x and y.

1.3.2 The adapted RSGD

We can now define the stochastic gradient descent in the context of a Riemannian manifold : this is the Riemannian stochastic gradient descent (RSGD) [1]. More speficially, we define it in the context of the Riemannian manifold $(\mathcal{B}^d, g_\theta)$, that is the open unit ball equipped with the Riemannian metric tensor. The RSGD follows 4 main steps :

1. Computation of the stochastic gradient ∇_E. ∇_E represents the Euclidean gradient, that is the gradient with the Eucliedean norm. More precisely, $\nabla_E = \frac{\partial \mathcal{L}(\theta)}{\partial d(\theta,x)} \frac{\partial d(\theta,x)}{\partial \theta}$ where $\mathcal{L}$ is the loss function (see 2. Implementation for an explicit loss function adapted to database used).
2. Correction of the Euclidean gradient by the Riemannian metric tensor to derive the Riemannian gradient: $g_\theta^{-1}\nabla_E$ where $g_\theta = (\frac{2}{1-||x||^2})^2 g^E$
3. Update rule: $\theta_{t+1} \longleftarrow \theta_t - \eta_t g_\theta^{-1}\nabla_E$
4. Projection of the result onto $\mathcal{B}$ at θ. This ensures that the solution remains within the Poincaré ball.

In order to implement in an easier way the gradient descent, one can re-write the updating rule. By setting $\alpha = 1-||\theta||^2$, $\beta = 1-||x||^2$ and $\gamma = 1 + \frac{2}{\alpha\beta}||\theta - x||^2$, the partial derivate of the Poincaré distance regarding θ can be calculated (we will not demonstrate this result, given in the article p.4) and we can consequently show that :

$$g_\theta^{-1}\nabla_E = \left(\frac{1-||x||^2}{2}\right)^2 \frac{\partial d(\theta,x)}{\partial \theta} = \frac{1}{\alpha\beta\sqrt{\gamma^2-1}}\left[\frac{||x||^2 - 2\langle\theta,x\rangle + 1 + \alpha x}{\alpha\beta}\right]$$

[3] On the contrary, Beltrami-Klein conservs straight lines but distorts angles
[4] a manifold is a topological space that locally resembles Euclidean space near each point

Working one more step on this expression enables us to calculate the gradient step only given metrics of two words w_1 and w_2 as well as the loss function derivative. We respectively call these two components left and right derivatives in the python script.
The two authors propose two other training details that are not commonly used with the RSGD procedure. First, they initialize all embeddings randomly from the uniform distribution $\mathcal{U}(-0.001, 0.001)$. This ensures that embeddings are close to the origin of $\mathcal{B}^d$ at the beginning of the training phase. Secondly, they define a specific "burn-in" procedure by reducing the learning rate η to $\frac{\eta}{c}$ for a certain number of initial epochs.

Regarding their results, authors interest themselves borth in reconstruction capacity from the embedding obtained, and the link prediction between vertexes of the network. Their results outperform the ones obtained in Euclidian geometry, especially for high dimensionality.

2 Implementation

Please refer to the GitHub `https://github.com/SamuelRitchie/GMML` dedicated to this project for the implementation, led in Python3. Running the Notebook will enable the reader to run analyses on either of the datasets we used, which are described in the following. Command line runnning is also possible.

2.1 Datasets used

To illustrate the method described above, we implement the Poincaré Embeddings in two different contexts : taxonomies and a given network.
Regarding taxinomies, we used a subset from the same database as the authors, that is Wordnet. Wordnet is a lexical database for the English language. It groups English words into sets of synonyms called synstes, and records a number of relations among these synonym sets. Words are separeted into nouns, verbs, adjectives, and adverbs. We focuse here on nouns and on two specific types of relations, hyponymy and hypernymy. Hyponymy is usually called a "is-a relation" : for instnace, x is the hyponym of y if "A x is-a y" . All the nouns are organized in a hierarchy that can be assimilated to a tree. The authors want to find this hierarchy back using a dataset where the hierarchical structure is not directly visible.
In this context, they define a specific loss function to minimize using the RSGD, which is called *negative sampling*:

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

$\mathcal{D} = \{(u, v)\}$ is the set of hypernymy relations between noun pairs. More specifically, if we take the word "music_genre", we build a subset based on this noun in the tree hierachy. $\mathcal{D}$ is the set of hypernymy relations between all the words of the subset. Furthermore, $\mathcal{N}(u) = \{v^{'}|(u, v^{'}) \notin \mathcal{D}\} \bigcup \{v\}$ is the set of negative examples. These are words that are not related to the word u. For the training, the authors randomly sample 10 negative examples per positive example.
The procedure led for the word "music_genre" can easily be changed to any other word present in WordNet, see Notebook for details.

Regarding our other dataset, we used the jazzmen connections datasets, scrapped from the LinkedJazz project, in order to see how Poincaré embeddings could find and represent interesting structures in the network. This network is composed of jazz musicians which are linked if they have played together in a band.

2.2 Results

The results presented here are build on the subtree built on the noun "music_genre". We plotted the two-dimensional Poincaré embeddings on figure .
We can analyze the quality of the embedding obtained here. There are two level of hierarchies that determine a good embedding. Locally, the children of each node should be spread out on the sphere around the parent. Globally, substrees should be separated from each other. To check for the local quality of the embedding, the authors use MAP. MAP captures how well each vertex's nieghborhoods are preserved. Figure 3 illustrates different levels of local quality of the embedding.

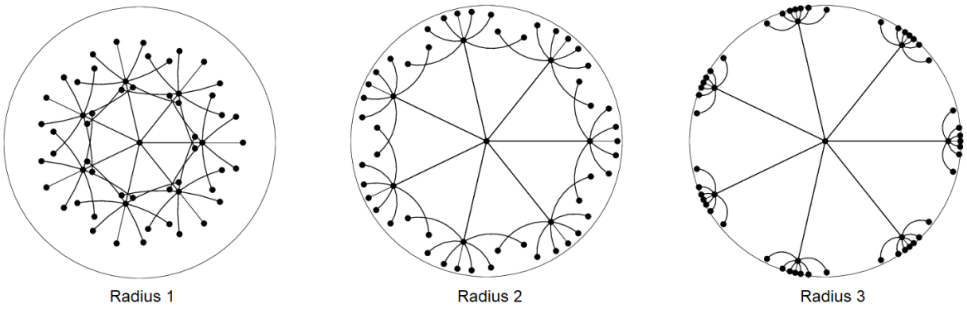


Figure 3: Evolution of the MAP value

On the left, with poor separation, when the embedding is not well trained yet, some nodes are closer to nodes from other substrees than to their actual neighbors, leading to a poor MAP value. The central embedding has a better MAP value. Finally, the last one on the right side is perfelcty well-separeted : the MAP has a value of one. We can see this evolution from a bad to a perfectly trained embedding through the training steps.

Below are presented our results obtained on the word "music_genre" in WordNet, respectively after 20 epochs (above) and 200 epochs (below), for a number of negative sampling equal to 10 per step, and a learning rate of 0.1 (except for the first first epochs were it is 0.01 as suggested by the authors).

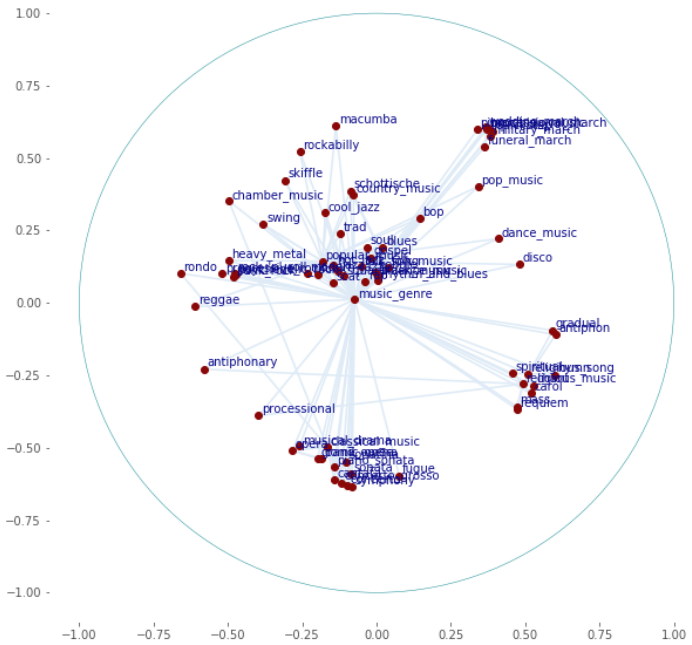


Figure 4: Results for 'music_genre' after 20 epochs

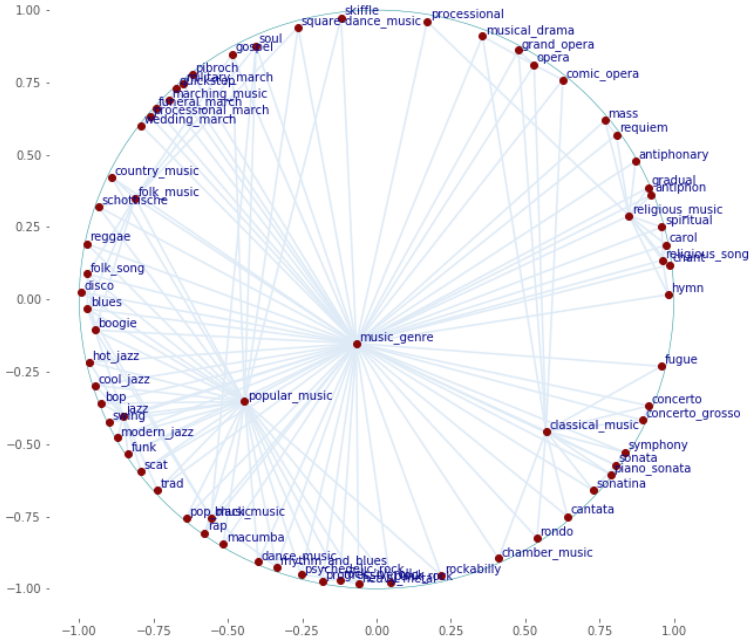


Figure 5: Results for 'music_genre' after 200 epochs

Words start to cluster after 20 epochs, but they are still all located in the center of the Riemannian ball. As expected, words organise themselves with the number of epochs. Results obtained after 200 epochs are satisfactory : we can clearly see that musical genre cluster well on the boundary of the unitary ball. Moreover, the intermediate labels such as 'popular music', 'classical music' or 'jazz' are nearer from the center, and the initial label 'music$_genre'isextremelyclosefromthecenter$.

Regarding the Jazzmen network dataset, we limited ourselves to 300 links for better vizualisation. Results obtained after 200 epochs are shown below.

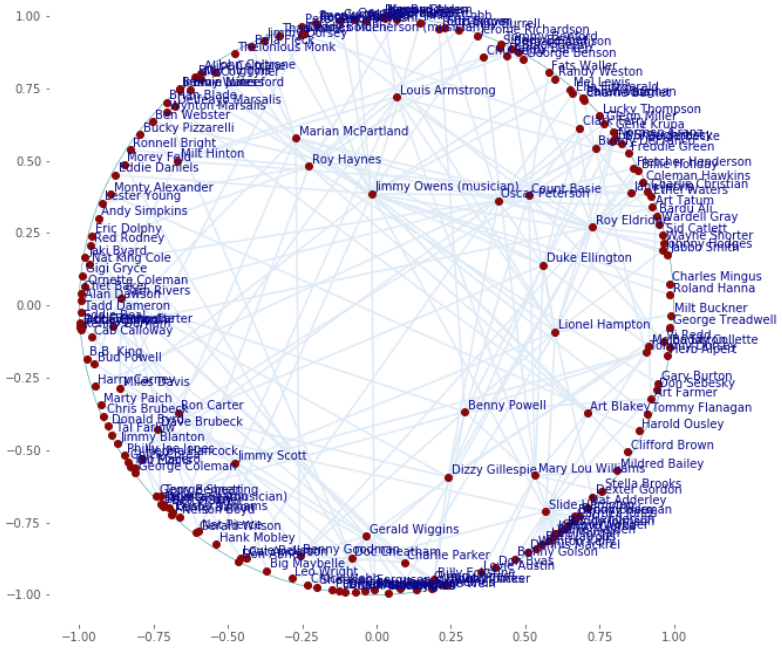


Figure 6: Results for the jazzmen network after 200 epochs

Results are difficult to interpretate because of the high number of artists. Addind metadata such as birth decade, jazz style, living area or the instrument played would be interesting to see if Poincaré Embeddings finds some interesting latent hierarchy in the network organization. However, we see that influent jazz musicians such as Louis Armstrong, Duke Ellington, Lionel Hampton or Benny Powell are situated nearer from the center than others, with appears logical.

Conclusion

We presented here a new method developped by M. Nickel and D. Kiela. It is based on the Poincaré ball-model. The authors assimilate the Poincaré ball to a Riemannian manifold, enabling them to use a Riemanian Stochastic Gradient Descent (RSGD) to optimize Poincaré embeddings. The results are quite interesting, achieving better results than in usual Euclidian geometry setting. Poincaré embeddings proved their ability to automatically capture hierarchical structure from the data.
When applying their algorithm to the word music_genre in WordNet, we found that is efficiently structured the data after 200 epochs and in a reasonable amount of time. Regarding network analysis, one would have to complete Poincaré Embeddings with labels to see how well this method hierarchises the network.
Besides, our implementation was led with a Numpy backend. For larger datasets, the implementation would undoubtedly have to be done with higher level backends such as PyTorch or Tensorflow to achieve quick results.
Also, although it may be true that hyperbolic spaces are better suited than Euclidean spaces to capture the semantic or functional relationships in tree-structured data, there is, in general, no reason to assume a priori that the underlying space has to be exactly hyperbolic. A better approach would be to learn the appropriate metric, rather than assuming it. We could still initialize it close to a hyperbolic metric, but making it flexible would allow us to capture possible deviations from hyperbolicity.

References

[1] S. Bonnabel. “Stochastic gradient descent on Riemannian manifolds”. In: 58(9) (2013), pp. 2217–2229.

[2] P. Pirolli J. Lamping R. Rao. “A focus+context technique based on hyperbolic geometry for visualizing large hierarchies”. In: 11 (1995), p. 11.