A hyperbolic approach for learning communities on graphs

Thomas Gerald, Hadi Zaatiti, Hatem Hajri, Nicolas Baskiotis, Olivier Schwander

A Hyperbolic approach for learning communities on graphs

Thomas Gerald · Hadi Zaatiti · Hatem Hajri · Nicolas Baskiotis · Olivier Schwander

Abstract Detecting communities on graphs has received significant interest in recent literature. Current state-of-the-art approaches tackle this problem by coupling Euclidean graph embedding with community detection. Considering the success of hyperbolic representations of graph-structured data in the last years, an ongoing challenge is to set up a hyperbolic approach to the community detection problem. The present paper meets this challenge by introducing a Riemannian geometry based framework for learning communities on graphs. The proposed methodology combines graph embedding on hyperbolic spaces with Riemannian K-means or Riemannian mixture models to perform community detection. The usefulness of this framework is illustrated through several experiments on generated community graphs and real-world social networks as well as comparisons with the most powerful baselines. The code implementing hyperbolic community embedding is available online https://github.com/tgeral68/HyperbolicGraphAndGMM.

Keywords Manifolds, Representation Learning, Graph-structured data, Hyperbolic space, Community embedding

T. Gerald, N. Baskiotis, O. Schwander
Science Sorbonne University, Lip6, 75005, Paris, France
E-mail: first.lastname@lip6.fr

H. Hajri
Institute of Research and Technology SystemX
8 avenue de la Vauve, 91120, Palaiseau, France
E-mail: first.lastname@irt-systemx.fr

H. Zaatiti
Airbus Defence and Space (Airbus DS SAS)
1 Bd Jean Moulin, 78990, Elancourt, France
E-mail: first.lastname@airbus.com

1 Introduction

In recent years, the idea of embedding data in new representation spaces has proven its effectiveness in many applications. Indeed, several techniques have become very popular due to their great ability to represent data while reducing the complexity and dimensionality of the space. For instance, Word2vec [28] and Glove [33] are widely used tools in natural language processing, Nod2vec [20], Graph2vec [3] and DeepWalk [34] are commonly used for community detection, link prediction and node classification in social networks [17].

The present paper is concerned with learning Graph Structured Data (GSD). Examples of such data include social networks, hierarchical lexical databases such as Wordnet [29] and Lexical entailment datasets such as Hyperlex [43]. GSD are increasing in popularity mainly due to being widely available and easily obtainable from online resources and social networks. For instance, graphs can represent links in social networks where nodes are users and edges are relationships between them. Representing GSD is an important topic related to many applications such as community detection, node classification and graph visualisation.

In the state-of-the-art, one can distinguish two different approaches to cluster GSD. The first one applies pure clustering techniques on graphs such as spectral clustering algorithms [39], power iteration clustering [25] and label propagation [46]. The second one is in two steps and may be called Euclidean clustering after (Euclidean) embedding. First it embeds data in Euclidean spaces using techniques such as Nod2vec, Graph2vec and DeepWalk and then applies traditional clustering techniques such as K-Means algorithms. This approach appeared notably in [13,41,44]. More recently the *ComE* algorithm [13] achieved state-of-the-art performances in detecting communities on graphs, the main idea is to alternate between embedding and learning communities by means of Gaussian mixture models.

Motivation. Learning GSD has known major achievements in recent years thanks to the discovery of hyperbolic embeddings. Although it has been speculated since several years that hyperbolic spaces would better represent GSD than Euclidean spaces [19,23,10], it is only recently that these speculations have been proven effective through concrete studies and applications [30,14, 37]. As outlined by [30], Euclidean embeddings require large dimensions to capture certain complex relations such as the Wordnet noun hierarchy. On the other hand, this complexity can be captured by a simple model of hyperbolic geometry such as the Poincaré disk of two dimensions [37]. Additionally, hyperbolic embeddings provide better visualisation of clusters on graphs or trees than Euclidean embeddings [14,30].

Recently, the *ComE* and *ComE++* [13,12] frameworks oriented the paradigm of graph representation towards learning communities of graphs. Despite few works applying deep learning methods to represent GSD on hyperbolic spaces such as Hyperbolic Graph Neural Networks [26] or Hyperbolic Convolutionnal Neural Networks [15] which provided competitive results compared to their

Euclidean counterparts, the effectiveness of hyperbolic spaces for learning communities has not been proven or explored yet.

Given the success of hyperbolic embedding to obtain faithful representations, it seems relevant to set up an approach that applies pure hyperbolic techniques in order to learn nodes and more generally community representations on graphs. This motivation is in line with several recent works that have demonstrated the effectiveness of hyperbolic tools for different applications in Brain computer interfaces [8], Computer vision [35] and Radar processing [4].

Contribution. Motivated by the success of hyperbolic embeddings on the one hand and hyperbolic learning algorithms on the other hand, we propose in this paper a new hyperbolic approach to the problem of learning node and community representations on graphs. Two different applications are targeted:
Unsupervised learning on GSD. We present a scheme to learn communities based on Poincaré embeddings [30], the Riemmannian K-Means algorithm and the recent formalism of Riemannian EM algorithm [36].
Supervised learning on GSD. We propose a supervised framework that uses community-aware embeddings of graphs in hyperbolic spaces.

For both the unsupervised and supervised frameworks, our proposed methods are evaluated on real-data social networks and compared with the *ComE* approach [13] and recent geometric methods [16].

This paper is organised as follows. Section 2 reviews the geometry of the Poincaré ball and related tools which will be used after: Riemannian barycentre and Riemannian Gaussian distributions. In Section 3, we review Poincaré embedding [30] and present our approach to learn GSD. Section 4 provides experiments and comparisons between our approach and state-of-the-art baselines. Finally Section 5 provides a conclusion of the paper and perspectives for the future.

2 The Poincaré Ball model: Geometry and Tools

In this section, we first review the geometry of the Poincaré ball model as a hyperbolic space, that is a Riemannian manifold with constant negative sectional curvature. In particular, we recall expressions of the exponential and logarithmic maps which will be used in the sequel. Then, we review definitions and properties related to Riemannian barycentre and Riemannian Gaussian distributions. Our approach presented in the next section strongly relies on algorithms K-Means and EM developed from these two concepts.

2.1 Geometry of the Poincaré ball

The Poincaré ball model of m dimensions $(\mathbb{B}^m, g^{\mathbb{B}^m})$ is the manifold $\mathbb{B}^m = \{x \in \mathbb{R}^m : ||x|| < 1\}$ equipped with the Riemannian metric:

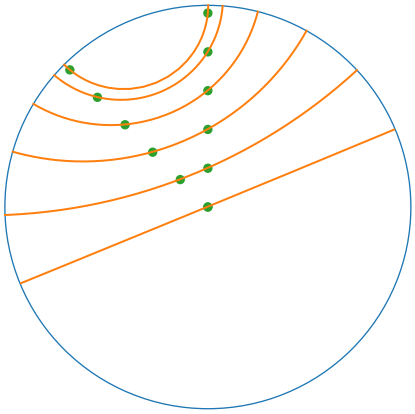


Fig. 1: Geodesics passing through two points in the *Poincaré disk*.

$$g_x^{\mathbb{B}^m} = \frac{4}{(1-||x||^2)^2} g^{\mathbb{E}}$$

where $g^{\mathbb{E}}$ is the Euclidean scalar product. The Riemannian distance between two points $x, y \in \mathbb{B}^m$, induced by this metric is given by:

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

Figure 1 shows different geodesics linking two points on $\mathbb{B}^2$. Notice that geodesics passing through the center are straight lines and as points get closer to the boundary, geodesics become more and more curved.

Endowed with this metric, $\mathbb{B}^m$ becomes a hyperbolic space [21,2]. [18] gave explicit expressions for the Riemannian exponential and logarithmic maps associated to this distance as follows. First, define Möbius addition $\oplus$ for $x, y \in \mathbb{B}^m$ as:

$$x \oplus y = \frac{(1 + 2\langle x, y\rangle + ||y||^2)x + (1 - ||x||^2)y}{1 + 2\langle x, y\rangle + ||x||^2||y||^2}$$

For $x \in \mathbb{B}^m$ and $y \in \mathbb{R}^m \setminus \{0\}$, the exponential map is defined as:

$$\operatorname{Exp}_x(y) = x \oplus \left(\tanh\left(\frac{||y||}{1-||x||^2}\right)\frac{y}{||y||}\right)$$

Intuitively, the exponential map is a generalisation to manifolds of the addition between a point and a vector. The addition of a point x with a vector y tangent at x, results in a point z belonging to the geodesic initiated from x and following the tangent vector.

Exp_x is a diffeomorphism from $\mathbb{R}^m$ to $\mathbb{B}^m$. Its inverse, called the logarithmic map is defined for all $x \neq y \in \mathbb{B}^m$ as:

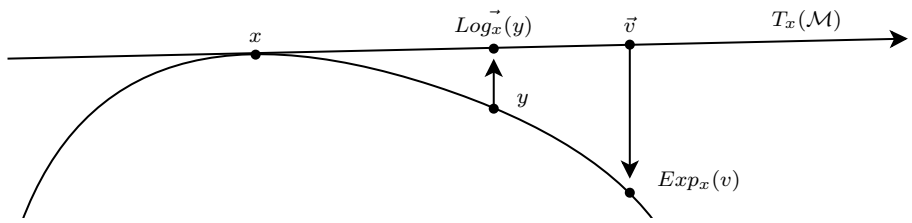


Fig. 2: Representation of the logarithmic and exponential maps on a manifold $\mathcal{M}$: Given two points $x, y \in \mathcal{M}$, $Log_x(y)$ is a tangent vector at x pointing towards y; and given some tangent vector v at x, then $Exp_x(v)$ is the point on the geodesic initiated from x that follows v.

$$\mathrm{Log}_x(y) = (1 - ||x||^2)\tanh^{-1}\left(||-x \oplus y||\right)\frac{-x \oplus y}{||-x \oplus y||}$$

The logarithmic map is generalisation of a subtraction to manifolds. Given, two points x and y, the result is the tangent vector from x oriented towards y. Figure 2 is an illustration of the Riemannian logarithmic and exponential maps on some manifold.

Another important concept for optimisation on Riemannian manifolds is parallel transport, that transports a tangent vector from one point to another alongside a geodesic curve. For the Poincare ball, the parallel transport of a vector v from point x to y is defined by

$$\varphi_{x \to y}(v) = \texttt{gyration}(y, -x, v)\frac{\lambda_x}{\lambda_y} \tag{2}$$

with the `gyration` function given in [18,42] (Equation 1.27) and $\lambda_x = \frac{2}{1-||x||^2}$ the conformal factor. The parallel transport is a key ingredient for optimisation using adaptive methods [9] such as *RAdam* or *RAMSGrad*. Figure 3 illustrates the parallel transport of a vector alongside a geodesic on the Poincaré disk.

2.2 Riemannian barycentre

As a Riemannian manifold of negative curvature, $\mathbb{B}^m$ enjoys the property of existence and uniqueness of the Riemannian barycentre [1]. More precisely, for every set of points $\{x_i, 1 \leq i \leq n\}$ in $\mathbb{B}^m$, the empirical Riemmanian barycentre

$$\hat{\mu}_n = \mathrm{argmin}_{\mu \in \mathbb{B}^m}\left(\sum_{i=1}^{n} d^2(\mu, x_i)\right)$$

exists and is unique. Several stochastic gradient algorithms can be applied to numerically approximate $\hat{\mu}_n$ [11,5,7,4,6]. Riemannian barycentre has given

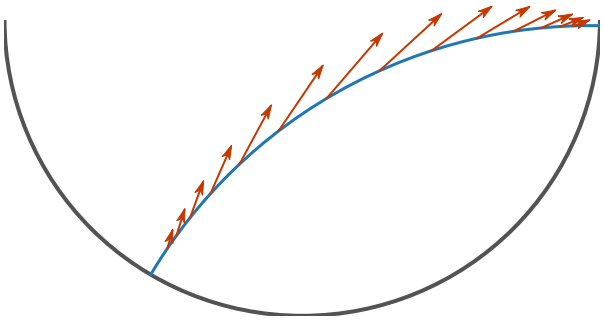


Fig. 3: Parallel transport of a vector (red) along a geodesic (blue) in the Poincaré disk; parallel transport preserves the angle between the vector and the disk as well as the vector direction.

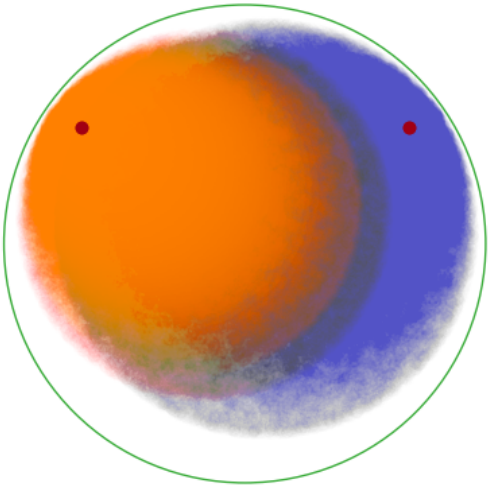


Fig. 4: Visualisation of two Gaussian distributions and their probability density functions as orange (small variance) and blue (large variance), in the Poincaré Ball with their Riemannian means (red points). Due to the curvature of the manifold, the means do not correspond to the centers of each area (orange and blue).

rise to several developments in particular, K-Means and EM algorithms which will be used in this paper.

2.3 Riemannian Gaussian Distributions

Gaussian distributions have been extended to manifolds in various ways [38,32, 36]. In this paper we rely on the recent definition provided in [36] as it comes up with an efficient learning scheme based on Riemannian mixture models. This

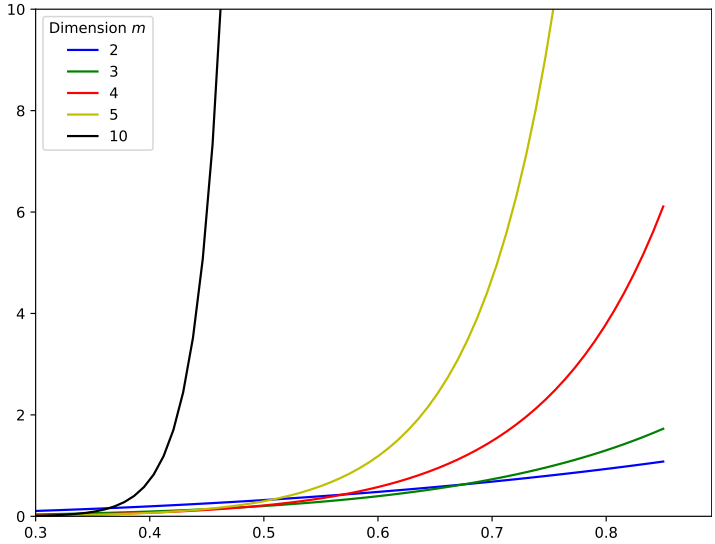


Fig. 5: Normalisation coefficient $\sigma \mapsto \zeta_m(\sigma)$ for different values of the dimension m of the Poincaré ball

distribution was particularly used in [27] to generalise variational-auto-encoders to the Poincaré Ball. Moreover [31] proposed an extension of Wasserstein auto-encoders based on the same definition of Gaussian distributions.

Given two parameters $\mu \in \mathbb{B}^m$ and $\sigma > 0$, respectively interpreted as theoretical mean (or barycentre) and standard deviation, the Riemannian Gaussian distribution $\mathcal{G}(\mu, \sigma)$ on $\mathbb{B}^m$, is given by its density:

$$f(x|\mu, \sigma) = \frac{1}{\zeta_m(\mu, \sigma)} \exp\left[-\frac{d^2(x, \mu)}{2\sigma^2}\right]$$

with respect to the Riemannian volume $dv(x)$ [36]. Figure 4 illustrates the density of two Gaussian distributions on $\mathbb{B}^2$. A first interesting and practical property is that $\zeta_m(\mu, \sigma)$ does not depend on μ:

$$\zeta_m(\mu, \sigma) = \zeta_m(0, \sigma) = \zeta_m(\sigma) = \int_{\mathbb{B}^m} \exp\left[-\frac{d^2(x, 0)}{2\sigma^2}\right] dv(x)$$

A more explicit expression of $\zeta_m(\sigma)$ has been given recently in [27] as follows:

$$\zeta_m(\sigma) = \sqrt{\frac{\pi}{2}} \frac{\sigma}{2^{m-1}} \sum_{k=0}^{m-1} (-1)^k C_{m-1}^k e^{\frac{p_k^2 \sigma^2}{2}} \left(1 + \mathrm{erf}(\frac{p_k \sigma}{\sqrt{2}})\right)$$

with $p_k = (m-1) - 2k$. Figure 5 displays some plots of the function $\sigma \mapsto \zeta_m(\sigma)$.

Recall Maximum Likelihood Estimation (MLE) of the parameters μ and σ, based on independent samples $x_1, \ldots, x_n$ from $\mathcal{G}(\mu, \sigma)$ given as follows:

– The MLE $\hat{\mu}_n$ of μ is the Riemannian barycentre of $x_1, \ldots, x_n$.
– The MLE $\hat{\sigma}_n$ of σ is

$$\hat{\sigma}_n = \Phi\left(\frac{1}{n}\sum_{i=1}^{n} d^2(\hat{\mu}_n, x_i)\right)$$

where $\Phi : \mathbb{R}_+ \longrightarrow \mathbb{R}_+$ is a strictly increasing bijective function given by the inverse of $\sigma \mapsto \sigma^3 \times \frac{d}{d\sigma} \log \zeta_m(\sigma)$.

3 Learning Hyperbolic Community Embeddings

The goal of embedding GSD is to provide a faithful and exploitable representation of the graph structure. It is mainly achieved by preserving *first-order* proximity that enforces nodes sharing edges to be close to each other. It can additionally preserve *second-order* proximity that enforces two nodes sharing the same context (i.e., nodes that are neighbours but not necessarily directly connected) to be close.

Consider a graph $G(V, E)$ where V is the set of nodes and $E \subset V \times V$ is the set of edges.

First-order proximity. It is preserved by optimising a loss function similar to [30]:

$$O_1 = -\sum_{(v_i, v_j) \in E} log(\sigma(-d^2(\phi_i, \phi_j))) \tag{3}$$

with $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function, $\phi_i \in \mathbb{B}^m$ is the embedding of the i-th node of V and d is the Riemannian distance.

Second-order proximity. It is preserved by forcing the representation of a node to be close to the representations of its context nodes. For this, we adopted the negative sampling approach [28] and considered the loss:

$$O_2 = -\sum_{v_i \in V} \sum_{v_j \in C_i} \left[log(\sigma(-d^2(\phi_i, \phi'_j))) + \sum_{v_k \sim \mathcal{P}_n} log(\sigma(d^2(\phi_i, \phi'_k))) \right] \tag{4}$$

with C_i the nodes in the context of the i-th node, $\phi'_j \in \mathbb{B}^m$ the embedding of $v_j \in C_i$ and $\mathcal{P}_n$ the negative sampling distribution over V given by $\mathcal{P}_n(v) = \frac{deg(v)^{3/4}}{\sum_{v_i \in V} deg(v_i)^{3/4}}$. The next paragraph introduces a third objective function allowing to detect and embed communities rather than individual nodes.

3.1 Expectation-Maximisation for Hyperbolic GMM

The Riemannian EM algorithm introduced in [36] has similarities with the usual EM algorithm on Euclidean spaces. It is employed to approximate distributions on manifolds by a mixture of standard distributions. We recall the density F of a GMM on the Poincaré ball

$$F(x|\mu,\sigma) = \sum_{k=0}^{K} \pi_k f(x|\mu_k,\sigma_k)$$

where $x \in \mathbb{B}^m$, π_k the mixture coefficient, μ_k, σ_k the parameters of the k-th Riemannian Gaussian distribution. The Riemannian EM algorithm computes the GMM that best fits a set of given points $x_1, \cdots, x_N$. This is done by maximising the log-likelihood of the joint probability of observing each point under the hypothesis that observations are independent. Given an initialisation of π, μ and σ, performing EM numerically translates to alternating between Expectation and Maximisation steps. Fitting a Gaussian mixture model on Riemannian manifolds have some similarities with its Euclidean counterpart. The main complexity is that there are no known closed forms for the log-likelihood maximisation of the mean and standard deviation. In the next section, we show how the EM algorithm was used to detect communities on graph embeddings.

3.1.1 Expectation

The expectation step estimates the probability of each sample x_i to belong to a particular Gaussian cluster. The estimation is based on the posterior distribution $\mathbb{P}(z_i = k|x_i)$, i.e. the probability that the sample x_i is drawn from the k-th Gaussian distribution:

$$w_{ik} := \mathbb{P}(z_i = k|x_i) = \frac{\pi_k \times f(x_i|\mu_k,\sigma_k)}{\sum_{j=1}^{N} \pi_k \times f(x_j|\mu_k,\sigma_k)} \tag{5}$$

In this expression $z_1, \cdots, z_N$ represent the latent variables associated to the mixture model.

3.1.2 Maximisation

The maximisation step estimates the mixture coefficient π_k, mean μ_k and standard deviation σ_k for each Gaussian component of the mixture model.

Mixture coefficient. The mixture coefficients are computed by :

$$\pi_k = \frac{1}{N} \sum_{i=1}^{N} w_{ik} \tag{6}$$

Mean. Updating the μ parameter relies on estimating weighted barycentres. It was therefore required to approximate the weighted barycentre $\hat{\mu_k}$ of the k-th cluster:

$$\hat{\mu_k} = \arg\min_{\mu} \sum_{i=1}^{N} w_{ik} d^2(\mu, x_i) \tag{7}$$

via Riemannian optimisation. We used Algorithm 1 of [4] which has proven effective for Radar applications. This algorithm returns an estimate of $\hat{\mu}_k$.

Algorithm 1 barycentre computation

Require : $W = (w_{ik})$ weight matrix, $\{x_1, \cdots, x_N\}$ a subset of $\mathbb{B}^m$, ϵ convergence rate (small), λ barycentre learning rate

1: Initialisation of μ_k^0
2: **do**
3: $\quad \mu_k^{t+1} \leftarrow \mathrm{Exp}_{\mu_k^t}\left(\lambda \frac{2}{\sum_{i=1}^{N} wik} \sum_{i=1}^{N} w_{ik} \mathrm{Log}_{\mu_k^t}(x_i)\right)$
4: **while** $d(\mu_k^t, \mu_k^{t+1}) > \epsilon$ ▷ t is the iteration index

return μ_k^{t+1}

Standard deviation. Recall the MLE of the standard deviation of the Gaussian distribution is previously defined in Section 2.3. Thanks to the property of Φ being strictly increasing and bijective, estimation of σ_k was given by solving the following problem :

$$\hat{\sigma_k} = \underset{\sigma_s}{\mathrm{argmin}} \left| \left(\frac{1}{\sum_{i=1}^{N} w_{ik}} \sum_{i=1}^{N} d^2(\mu_k, x_i) w_{ik} \right) - \Phi^{-1}(\sigma_s) \right| \tag{8}$$

We used a grid-search to find an approximation of σ_k by computing its values for a finite number of σ_s. To compute $\Phi^{-1}(\sigma_s)$ which involves the term $\frac{d}{d\sigma} log \zeta_m(\sigma)$ we used automatic differentiation algorithm provided by the back-end. The procedure for choosing the state space of σ_s are provided and discussed in Section 4.

3.2 Learning with Communities

Since our main objective is to learn embeddings for the purpose of community detection and classification, we need to ensure that embedded nodes belonging to the same community are close. Therefore, similarly to the statement made in *ComE* [13] we connect node embedding (first and second-order proximity via the minimisation of O_1 and O_2) together with community awareness using

a *third-order* loss function O_3. The later is named the community loss, which we write as:

$$O_3 = -\sum_{i=1}^{N}\sum_{k=0}^{K} w_{ik} log\left(\frac{1}{\zeta_m(\sigma_k)} e^{-\frac{d^2(x_i,\mu_k)}{2\sigma_k^2}}\right) \tag{9}$$

To solve community and graph embedding jointly, we finally minimise:

$$L = \alpha.O_1 + \beta.O_2 + \gamma.O_3$$

with α, β, γ the weights of respectively first, second-order and community losses.

3.3 Optimisation

We now focus on the Riemannian optimisation of the loss functions O_1, O_2 and O_3. Following the idea of [18], we used the Riemannian Gradient Descent (RGD) Algorithm, which first computes the gradient on the tangent space and then projects the new values on the ball

$$\phi^{t+1} = \mathrm{Exp}_{\phi^t}\left(-\eta \frac{\partial O}{\partial \phi}\right) \tag{10}$$

where O is a function to optimise, ϕ is a parameter, $t \in \{1, 2, \cdots\}$ is the iteration number and η is a learning rate. Recall from [4] the formula giving the gradient of $d^p(\phi_i, \phi_j)$ where $\phi_i, \phi_j \in \mathbb{B}^m$:

$$\nabla_{\phi_i} d^p(\phi_i, \phi_j) = -p * d^{p-1}(\phi_i, \phi_j) * \frac{\mathrm{Log}_{\phi_i}(\phi_j)}{d(\phi_i, \phi_j)} \tag{11}$$

Using the chain rule, the gradient of a function $h = g \circ d^p$ (g being a differentiable function) can be computed as follows:

$$\nabla_{\phi_i} h = g'(d^p(\phi_i, \phi_j)) \nabla_{\phi_i} d^p(\phi_i, \phi_j)$$

where the expression of $\nabla_{\phi_i} d^p(\phi_i, \phi_j)$ is given in Equation (11). In the attached code (detailed in the Appendix), optimisation was performed by redefining the gradient of the distance and then using usual auto-derivation tools provided by PyTorch backend.

To avoid division by $d(\phi_i, \phi_j)$, which becomes computationally difficult when ϕ_i and ϕ_j are close, we adopted $p = 2$ which experimentally revealed to be numerically more stable. Explicit forms for the gradient of O_1, O_2 and O_3 are as follows:

Update of O_1. It is based on the computation of the gradient of O_1 (given in (3)) with respect to ϕ_i obtained as follows:

$$\nabla_{\phi_i} O_1 = -2 \times \mathrm{Log}_{\phi_i}(\phi_j) \times \sigma(d^2(\phi_i, \phi_j))$$

In this last formula, we used the fact that $log(\sigma(-x))' = -\sigma(x)$. By symmetry, $\nabla_{\phi_j} O_1 = -2 \times \mathrm{Log}_{\phi_j}(\phi_i) \times \sigma(d^2(\phi_i, \phi_j))$.

Update of O_2. The updates of ϕ_i, ϕ'_j and ϕ'_k (the embeddings of v_i, v_j and v_k where v_i is a node, v_j belongs to the context C_i of v_i and v_k a negative sample of v_i) in (4) are given as follows:

- Update of ϕ'_j is done exactly as in O_1 since it occurs only in the first term of the sum.
- Update of ϕ'_k is based on the gradient computation

$$\nabla_{\phi'_k} O_2 = -2 \times \mathrm{Log}_{\phi'_k}(\phi_i) \times \sigma(-d^2(\phi_i, \phi'_k))$$

- Update of ϕ_i is based on the gradient computation

$$\nabla_{\phi_i} O_2 = \sum_{v_j \in C_i} \Big[-2 \times \mathrm{Log}_{\phi_i}(\phi'_j) \times \sigma(d^2(\phi_i, \phi'_j)) + \sum_{k=1}^{n_{neg}} \big(2 \times \mathrm{Log}_{\phi_i}(\phi'_k) \times \sigma(-d^2(\phi_i, \phi'_k))\big) \Big]$$

where n_{neg} is the number of negative samples.

Update of O_3. Recall the expression of O_3 from (9). The updates of $w_{ik}, \zeta_m(\sigma_k)$ and σ_k were detailed in Subsection 3.1. The update of ϕ_i uses the gradient

$$\nabla_{\phi_i} O_3 = \sum_{k=0}^{K} \frac{w_{ik}}{2\sigma_k^2} \nabla_{\phi_i}(d^2(\phi_i, \mu_k)) \tag{12}$$

Optimisation. To update each parameter, we use the RGD algorithm (Equation 10) and its adaptive counterpart RAMSGrad (Algorithm 2) [9]. We observed better performances with the adaptive method which is relied on in the next section.

Algorithm 2 Riemannian RAMSGrad

Require : f function to optimise, η learning rate, β_1, β_2 tradeoff parameters between the weight attributed to the gradient history and its current value, m_t^x, v_t^x, τ_t^x internal variables from previous update

procedure UPDATE $x_t(x_t, f, \eta, \beta_1, \beta_2, m_t^x, v_t^x, \tau_t^x)$
  $\nabla_{x_t}^{T_{x_t}\mathbb{B}} f(x_t) \leftarrow$ gradient on the tangent space in x_t
  $m_{t+1}^x \leftarrow \beta_1 \tau_t^x + (1-\beta_1)\nabla_{x_t}^{T_{x_t}\mathbb{B}} f(x_t)$
  $v_{t+1}^x \leftarrow \beta_2 v_t^x + (1-\beta_2)||\nabla_{x_t}^{T_{x_t}\mathbb{B}} f(x_t)||^2$
  $v_{t+1}^x \leftarrow max\{v_{t+1}^x, v_t^x\}$
  $x_{t+1} \leftarrow \mathrm{Exp}_{x_t}(-\eta \frac{m_{t+1}^x}{\sqrt{v_{t+1}^x}})$
  $\tau_{t+1} \leftarrow \varphi_{x_t \to x_{t+1}}(m_{t+1}^x)$ ▷ ϕ being the parallel transport function from Equation 2
end procedure

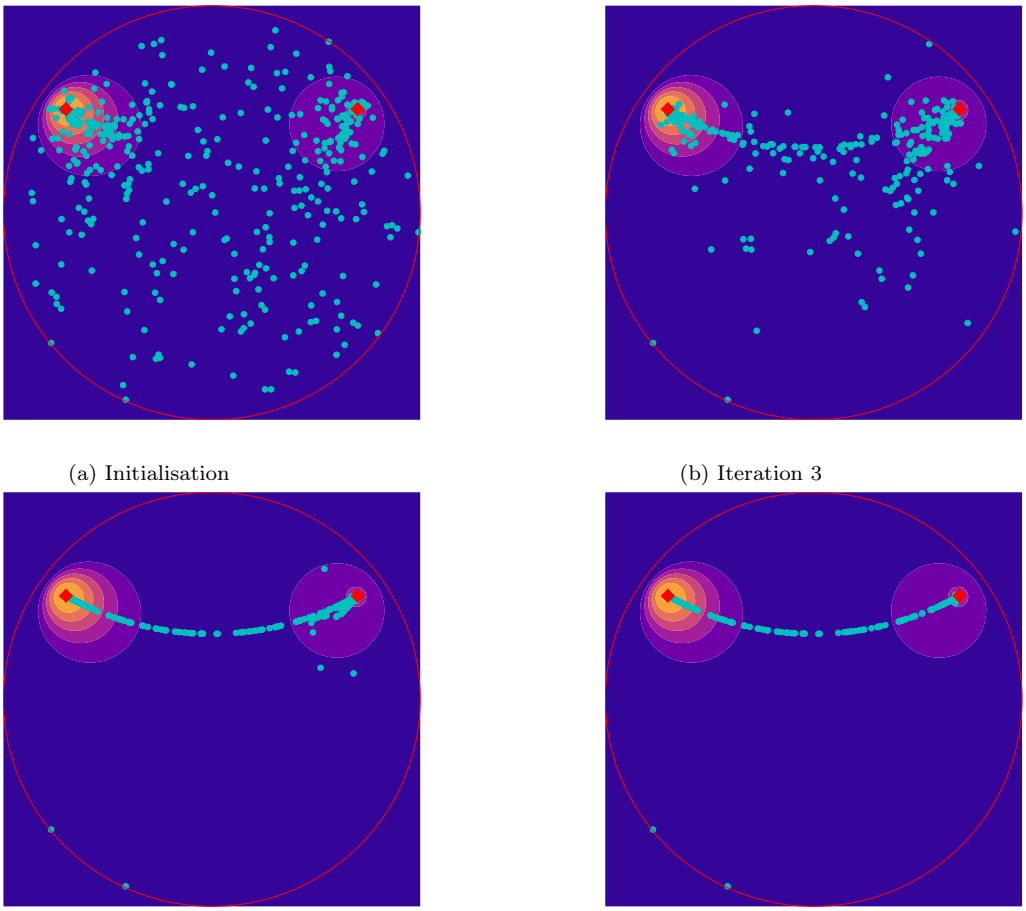
Hyperbolic community learning algorithm. Algorithm 3 presents the high level scheme for learning community-aware embeddings of graph data on hyperbolic spaces. It proposes to learn embeddings based on O_1 and O_2 first, detect communities by applying the EM algorithm second, then perform community embedding by optimising O_3. In fact, since initialisation is random, communities will first need to be formed by optimising O_1 and O_2. Only then, the EM algorithm would be successful in capturing representative communities. Experimentally, we pre-trained the embeddings by optimising O_1 and O_2 for several iterations before using the complete loop. When the EM algorithm starts capturing communities, the optimisation of O_3 with respect to node embeddings uses the captured communities to embed the nodes by moving each node closer to the most likely community, i.e., in the direction maximising the probability density function of the GMM.

To visualise the effect of community embedding, we trained a GMM with two components, distant barycentres and different variances. Figure 6 shows the effect of optimising O_3 via Equation (12) for randomly generated points at different iterations. Notice how the points settle at the geodesic linking the two barycentres except for points that were initially close to the boundary of the disk (considered at infinity). Indeed, the direction towards the geodesic is the one maximising the probability density function of the GMM. In this experiment, the posterior probabilities w_{ik} are not updated. Consequently the points settle at the geodesic without moving further towards one of the barycentres.

Algorithm 3 Community learning and detection on graphs using hyperbolic embeddings

Require : $G(V, E)$ graph data, K number of communities, m dimension of the hyperbolic space, l random walk length, n_w number of walks from some node, n_c context size, n_{neg} number of negative samples, `max_iter1,2,3` full loop, community detection, respectively, community embedding maximum iterations
Ensures : ϕ node embeddings

Initialisation of ϕ, μ, σ, π randomly from a uniform distribution with fixed bounds
for `iter<max_iter1` **do**
  for $(i, j) \in E$ **do**
    Update ϕ_i, ϕ_j via RGD based on $\alpha.O1$
  end for
  for $p \in P$ **do**
    for $i \in p$ **do**
      Update ϕ_i and all ϕ_j in the context of ß via RGD of $\beta.O_2$
        ▷ Select context and negatively sampled nodes of size n_c and n_{neg} by generating n_w Random walks of length l from each node as in *DeepWalk* [34]
    end for
  end for
  for `iter<max_iter2` **do**
    Update μ, σ, π ▷ Community detection by alternating between Exptectation and Maximisation steps as described in Subsection 3.1
  end for
  for `iter<max_iter3` **do**
    Update ϕ_i given the detected communities ▷ Community embedding by applying the update of Equation 12
  end for
end for



(a) Initialisation

(b) Iteration 3

(c) Iteration 15

(d) Iteration 40

Fig. 6: Visualising the effect of optimising the community embedding loss function: Initialisation of random points (cyan) on the hyperbolic space and a Gaussian mixture model (with probability density function illustrated as a purple to yellow heat map) with two components (red squares representing the two barycentres)

3.4 Prediction

The previous paragraphs presented our approach to learn community-aware embeddings of graph data. In what follows, we propose evaluations protocols for algorithms on unsupervised and supervised community problems. This will allow us in the next section to experimentally evaluate the efficiency of our approach.

Unsupervised learning.

- Hyperbolic K-Means (H-KM): This algorithm finds K cluster centroids and labels points by minimising the intra-cluster variances. Based on the notion of Riemannian barycentre, Algorithm 4 illustrates Riemannian K-Means.

Algorithm 4 Hyberbolic K-Means

Require : K number of clusters, $\phi \subset \mathbb{B}^{n\times m}$ node embeddings, `max_iter` maximum iterations
Ensures : I labels of each sample, μ cluster centroids

Initialisation of centroids $\mu \subset \mathbb{B}^{K\times m}$ randomly from a uniform distribution
for $t \in \{1, 2, 3, \dots$ `max_iter`$\}$ or *convergence* **do**
  for $i \in \{1, 2, \dots, n\}$ **do**
    $I_i^t \leftarrow \text{argmin}_j\, d(\mu_j, \phi_i)$
  end for
  for $k \in \{1, 2, \dots, K\}$ **do**
    Update barycentre $\mu_k \leftarrow RiemanianBarycentre(\{\phi_j | I_j = k\}$
  end for
  $convergence \leftarrow$ True iff $\forall i \in \{1, 2, \dots, n\}$, $I_i^{t-1} = I_i^t$
end for

- Hyperbolic Expectation-Maximisation (H-EM): We apply several EM iterations to obtain a well converged GMM. Then, the probability for a node to belong to some community (modelled by one component of the GMM) is computed. Each node is then labelled with the community giving it the highest posterior probability.

Supervised learning. Assuming to know the communities of some nodes, the objective is to predict the communities of unlabelled nodes while using the computed community-aware embedding. The nodes are split into a test set and a validation set. We will use three different methods to predict the labels of the validation set:

- Hyperbolic Barycentres (H-B): This method uses a supervised version of K-Means by computing the Riemannian barycentre of nodes known to belong to a given community. Each barycentre is considered as a cluster centroid representing the community. Then nodes from the validation set are associated to the communities of the n nearest barycentres (depending on the considered *Precision@n* discussed in the next section).

– Hyperbolic GMM (H-GMM): Given the known labels of the train data, a GMM is estimated. The parameters of the GMM are obtained by applying one maximisation step given the train nodes (Section 3.1.2) using the following estimate:

$$w_{ik} = \frac{y_{i,k}}{\sum_{k=0}^{K} y_{i,k}}$$

with $y_{i,k} = 1$ if node i belongs to the community k and $y_{i,k} = 0$ otherwise. To predict the community $\hat{k}$ of a given node embedded as $x_i \in \mathbb{B}^m$, we use Bayes decision rule:

$$\hat{k} = \text{argmax}_k \mathbb{P}(k|x_i) = \text{argmax}_k w_{ik} f(x_i|\hat{\mu_k}, \hat{\sigma_k})$$

with $\hat{\mu_k}, \hat{\sigma_k}$ the estimated parameters of the k-th Gaussian distribution.

– Hyperbolic Logistic Regression (H-LR): In order to further compare the baseline *ComE* with our proposed method developed in this paper, we propose similarly to [13] to learn a classifier. For the baseline *ComE*, we use the usual Euclidean logistic regression. For its Riemannian version, we rely on the hyperbolic logistic regression proposed in [18]. From this paper, recall that for a given hyper-plane $H_{a,p}$ defined by a point $p \in \mathbb{B}^m$ and a tangent vector $a \in T_p\mathbb{B}^m$, the distance of a given $x \in \mathbb{B}^m$ to $H_{a,p}$ is:

$$d(x, H_{a,p}) = sinh^{-1}\left(\frac{2| < -p \oplus x, a_k > |}{(1 - || - p \oplus x||^2)||a_k||}\right) \tag{13}$$

Then, the probability of a node x to belong to the community k is modelled by

$$\mathbb{P}(z = k|x) = \sigma\big(sign(-p \oplus x)d(x, H_{a,p})\big) \tag{14}$$

with z the community latent variable.

3.5 Complexity and Scalability

Complexity. With N the number of nodes and $|E|$ the number of edges, the time complexity of embedding GSD for first and second-order proximity are respectively in $\mathcal{O}(|E|)$ and $\mathcal{O}(N)$. Assuming a fixed number of iterations for all gradient descent computations, performing embedding and Riemannian barycentre based K-Means has $\mathcal{O}(N.m.K+|E|)$ complexity thus is in $\mathcal{O}(N+|E|)$ for large graphs. Similarly, the time complexity of community embedding with EM loop is in $\mathcal{O}(N + |E|)$. Table 1 provides a complexity comparison for node and community learning algorithms from the literature. The table shows that our work introduces no further complexity compared with the strongest state-of-the-art community learning frameworks.

	Complexity
DeepWalk [34]	$\mathcal{O}(Nlog(N))$
Node2vec [20]	$\mathcal{O}(Nlog(N) + N.a^2)$
LINE [40]	$\mathcal{O}(a\|E\|)$
SDNE [45]	$\mathcal{O}(a.N)$
ComE[13]	$\mathcal{O}(N + \|E\|)$
Hyperbolic Community Learning (This work)	$\mathcal{O}(N + \|E\|)$

Table 1: Complexity comparison between hyperbolic community learning (our work) and other frameworks from the literature (N the number of graph nodes, $|E|$ the number of edges and a the average graph degree)

Scalability. The GSD embedding update process for O_1 and O_2 in $\mathbb{B}^m$ is a linear function of m. Therefore, embedding the GSD scales well to large datasets. To accelerate the updates of O_1, O_2, O_3, we use batch gradient descent algorithm mainly for large datasets.

Although run-times of Riemannian K-Means and EM are higher than their Euclidean versions (mainly due to RGD), scalability is not affected by the change of the underlying space: each operation has higher run-time but the total number of operations as function of the dataset size scales similarly to Euclidean spaces. Notice that the computation of the normalisation factor $\zeta_m(\sigma)$ makes fitting hyperbolic GMM computationally challenging, this can be visually perceived in Figure 5. Thus, numerically we have to deal with the out of bound numbers due to the terms $e^{\frac{(m-1)^2\sigma^2}{2}}$. To keep the computation time-bounded, we fixed a finite number of values for σ for which we pre-computed ζ_m. To avoid the out of bound issue we restricted our work to 10 dimensions while increasing the floating-point precision (see Section 4). Similarly, as explained in the last paragraph of Section 3.1.2 for computing the variance and its gradient, we precompute them numerically in static tables instead of on demand computations. Experimentally, precomputing these parameters helped us reduce some of the computation burden and increase scalability.

4 Experiments

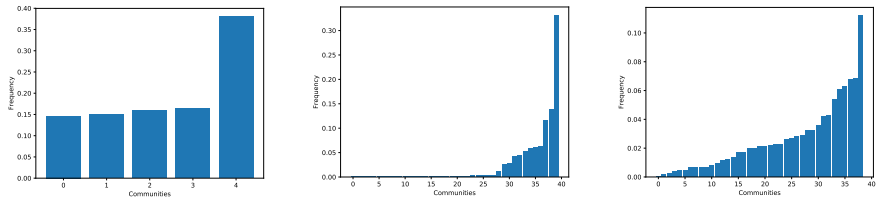
In this section, we compare our approach with recent works from the literature based on several experiments[1].

Datasets. As a first step to validate the framework, we present experiments on generated synthetic community graphs. The *Lancichinetti–Fortunato–Radicchi* benchmark (*LFR*) is a graph generation framework [24] that relies on the construction of community graphs. The LFR benchmark provides solutions to many problems that previous community graph generation methods encountered such as: nodes of generated networks having same degrees, communities

[1] `https://github.com/tgeral68/HyperbolicGraphAndGMM`

being of the same size and the generated networks being generally small. To tackle these issues, the authors of [24] generate the degree for each node (N nodes) according to a power law distribution. The support of this distribution is chosen according to k_{min} and k_{max}, the minimal, respectively, the maximal degree (or k_{mean}). Similarly, the size of each community is modelled by another power law distribution. The number of communities is also chosen from a support $s_{min} > k_{min}$ and respectively $s_{max} > k_{max}$ (or s_{mean}). Each node shares edges with its community at a probability of Λ and with other communities at $1 - \Lambda$.

Secondly, we present experiments on *DBLP*[2] a graph of scientific papers, *Wikipedia*[3] a graph built on Wikipedia dump, *BlogCatalog*[4] a blog social network graph and *Flickr*[5] a graph based on Flickr users. We also experiment on several low scale datasets and provide visualisations of the learned embeddings when the dataset allows it. Figure 7 depicts the community distribution for the DBLP, Wikipedia and BlogCatalog graphs, it shows the number of nodes (Y-axis) belonging to each community identified by an ID number (X-axis). For all the experiments presented below, we used 2080 TI RTX GPU during training to accelerate processing time. Table 2 shows the characteristics of the used



(a) DBLP (b) Wikipedia (c) BlogCatalog

Fig. 7: The ground-truth community distributions for *DBLP*, *Wikipedia* and *BlogCatalog* datasets. X-axis is identification number of some community and Y-axis the number of nodes belonging to the community.

datasets.

Hyper-parameters selection. Some parameters involved in the learning process are difficult to know a priori. Therefore, we performed several grid-search to save the hyper-parameters with the best cross-validation performances in terms of conductance, defined below, when running our algorithm and the baseline *ComE*. In particular, the parameter λ, the learning rate, c, the size of the context window and t, the number of negative sampling nodes appeared to

[2] https://aminer.org/billboard/aminernetwork
[3] http://snap.stanford.edu/node2vec/POS.mat
[4] http://socialcomputing.asu.edu/datasets/BlogCatalog3
[5] http://socialcomputing.asu.edu/datasets/Flickr

Table 2: Characteristics of the datasets used during experiments. N is the number of nodes, $|E|$ is the number of edges, K is the number of communities and ML is whether or not the dataset is multi-label (whether or not a node can belong to several communities).

Corpus	N	$\|E\|$	K	Multi-Label
Karate	34	77	2	no
Polblogs	1224	16781	2	no
Books	105	441	3	no
Football	115	613	12	no
LFR	800	-	13	no
DBLP	13,184	48,018	5	no
Wikipedia	4,777	184,812	40	yes
BlogCatalog	10,312	333,983	39	yes
Flickr	80,513	5,899,882	195	yes

be the most influencing on the results. For our algorithm [6], the parameters c and t were selected from the set $\{5, 10\}$, β and $\alpha \in \{0.1, 1\}$, $\gamma \in \{0.01, 0.1\}$ and λ from $[1e-2, 1e-4]$. For all experiments, we generated, for each node, 10 random walks, each of length 80. For the first 10 epochs, embeddings are trained using only O_1 and O_2 and then using the complete loop as described in Section 3.2.

As for running *ComE*, the tested parameters are $\lambda \in \{0.1, 0.01\}$, $\beta \in \{1, 0.1\}$ and $\gamma \in \{0.1, 0.01\}$, then the experiments giving the best performances (according to conductance) are relaunched 5 times to obtain variances and means. All others parameters are the default ones[7].

Variance search space procedure. Choosing the search space for the values of σ follows a method based on experimental results. We pre-compute Φ^{-1} for a range of σ values. The main objective is to cover, at best, a wide range of values of σ. We proceed as follows: For each Gaussian component, for $\sigma \in \mathbb{R}^+$ we select a range of values $A \in \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ with $\sigma_i = a + bi$ with $a = 5e^{-3}$ and $b = 1e-3$, and $k = 2000$, the values of a and b have been empirically chosen and tuned according to the communities. Additionally, we noted that the value of σ required for achieving high performances in representing communities never exceeded the value 2, a value that models largely scattered data. As for the computation time, since ζ_m is only computed once at the first step of the Expectation-Maximisation algorithm, the computation time is not affected (we could also store the value instead of computing it at run time). Retrieving the correct Φ^{-1} for each θ is computationally cheap and costs only a search in a sorted table which has a negligible complexity of $O(log(k))$.

[6] For Flickr dataset we only run one experiment for each parameter, moreover, the number of parameters tested is lower than those tested for others datasets

[7] *ComE* code repository is available at https://github.com/vwz/ComE

Evaluation metrics. We used the following metrics to assess the relevance of the proposed hyperbolic approach for learning communities on graphs.

• *Conductance:* measures the number of edges shared between separate clusters. The lower the conductance is, the less edges are shared between clusters. Let C_i be the set of nodes for the cluster i, A the adjacency matrix of the GSD, the mean conductance MC over clusters is given by:

$$MC = \frac{1}{K}\sum_{i=1}^{K}\frac{\sum\limits_{j\in C_i, k\notin C_i} A_{jk}}{\min\left(\sum\limits_{j\in C_i, k\in V} A_{jk}, \sum\limits_{j\notin C_i, k\in V} A_{jk}\right)} \tag{15}$$

• *Normalised Mutual Information (NMI):* Let G_{ij} be the number of common nodes belonging to both the predicted cluster i and the real cluster j, N being the number of nodes, G_i^p the number of elements in the i-th predicted cluster and G_i^t the number of elements in the i-th real community. The NMI is given by:

$$\text{NMI} = \frac{-2\sum\limits_{i=0}^{K}\sum\limits_{j=0}^{K} G_{ij} log\left(\frac{G_{ij}N}{G_i^P G_j^t}\right)}{\sum\limits_{i=0}^{K} G_i^p log\left(\frac{G_i^p}{N}\right) + \sum\limits_{j=0}^{K} G_j^t log\left(\frac{G_j^t}{N}\right)} \tag{16}$$

• *Precision@1*: Since the real labels of communities are known, we propose a supervised measure derived from the precision metric that uses the true community labels.

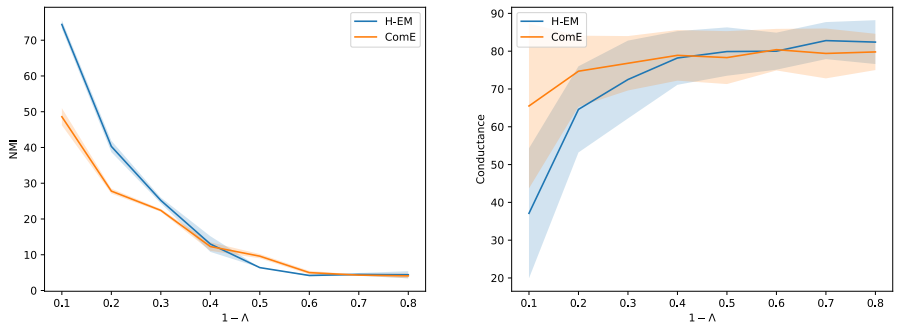
A problem we encountered is that the association between predicted labels and true ones is unknown. For a small number of communities, all possible combinations are computed and the best performance is reported. This solution is not tractable when the number of communities grows. A greedy approach is then used: the predicted labels of the largest cluster is first associated to the known true labels of the dominant class. Similarly the second largest cluster is associated with the second dominant class with known labels and so on. The process is repeated until all clusters are treated. Finally, for mono-label datasets, the *Precision@1* corresponds to the mean percentage of the correctly guessed labels. For multi-label datasets, an element is considered correctly labelled if the inferred community corresponds to one of its true known communities.

4.1 Unsupervised Clustering and Community Detection

In this section, we present experiments to validate the relevance of the hyperbolic community-aware embeddings based on the H-K-Means and H-EM frameworks for the task of unsupervised community detection.

First, we performed several experiments on the synthetic graph generated from *LFR*. Figure 8a and 8b depicts respectively the NMI and conductance for the generated graphs. The performances are reported for several values of

the community density parameter Λ while the rest of the parameters are fixed. One should note that the hyperbolic framework reaches better performances when considering hugely connected communities (i.e., small values of $1 - \Lambda$). Figure 9 shows the embedding resulting from the proposed learning framework. Figure 9a is coloured according to the real communities while Figure 9b is coloured according to the unsupervised H-EM algorithm.



(a) NMI (higher is better)

(b) Conductance (lower is better)

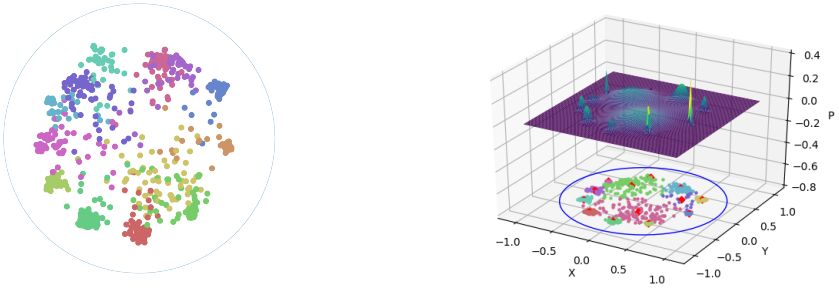
Fig. 8: A performance comparison of the unsupervised approach H-EM with the baseline ComE as function of $1 - \Lambda$ (higher Λ means more edges a node shares with its true community) applied to embeddings of the synthetic *LFR* graph in two dimensional space.

Second, for each real world dataset, we performed 5 experiments. The average mean of the performance metrics are shown in Table 3.

4.2 Supervised Classification

In this section, we present experiments to assess the effectiveness of hyperbolic community learning in the supervised context. The evaluation relies on three different ways to predict labels described previously in Section 3.4: H-B, a supervised version of K-Means based on the Riemannian barycentre, H-GMM, hyperbolic Gaussian mixture models, or H-LR, hyperbolic Logistic Regression based on geodesics. For all experiments, we applied a 5-cross-validation process with 20% of the dataset used for validation. The *Precision@1* is reported for mono-label datasets (i.e., each element belongs to a unique community), additionally *Precision@3* and *5* are reported for multi-label datasets (i.e., each element can belong to several communities). Table 4 reports the mean and standard deviation of the *Precision* over 5 runs.

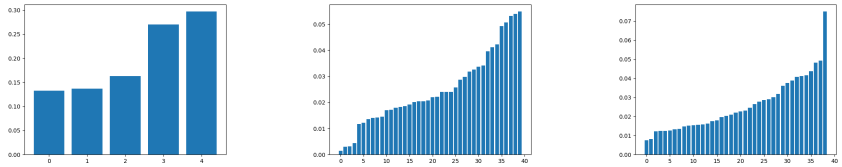
Hyperbolic SVM Comparison. To support the relevance of the proposed classification approaches, we compare in Table 5 our results with *Hyperbolic* SVM



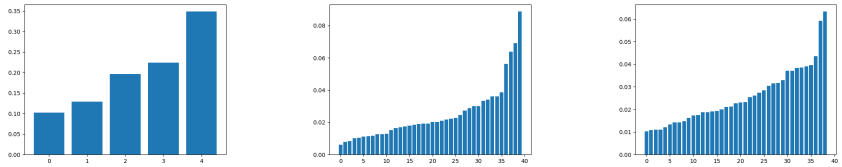
(a) Ground Truth: Each point represents a graph node, each color is a true known community

(b) Prediction model via Hyperbolic Expectation-Maximisation: each color is the predicted community of some node

Fig. 9: Visualisation of unsupervised approach applied on LFR with parameter $\mu = 0.9$.



(a) DBLP (H-GMM) (b) Wikipedia (H-GMM) (c) BlogCatalog (H-GMM)



(d) DBLP (*ComE*) (e) Wikipedia (*ComE*) (f) BlogCatalog (*ComE*)

Fig. 10: Example of predicted community distributions considering 10 dimensional embeddings for hyperbolic GMM and ComE

[16]. The performances are reported for embeddings of small datasets in two dimensions, using 5 folds cross-validation. The context size was set to 3 for *Football* and 2 for the other datasets. The reported results are directly taken from the H-SVM paper (where 5 folds cross-validation are used as well).

Table 3: Unsupervised community detection performances for Hyperbolic K-Means (H-KM) and Expectation-Maximisation (H-EM) in comparison with state-of-the-art method *ComE*.

		Precision@1		
Dataset	m	H-KM	H-EM	ComE
DBLP	2	78.5±1.8	**78.6**±4.8	60.4±5.5
	5	79.6±2.4	**81.2**±2.1	78.0±4.7
	10	71.4±13.1	**81.5**±0.1	78.2±1.2
Wikipedia	2	8.6±0.8	**16.1**±4.0	8.8±0.2
	5	9.7±0.4	**10.1**±0.7	**10.1**±0.2
	10	9.3±0.3	**11.9**±1.1	**12.1**±0.7
BlogCatalog	2	8.1±0.2	**9.8**±0.3	7.0±0.1
	5	**13.4**±0.3	12.6±0.7	12.3±0.3
	10	**18.9**±0.6	16.5±0.8	15.9±0.5
Flickr	2	8.0±0.2	**12.9**±0.8	6.2±0.
	5	13.0±0.1	**13.4**±0.1	10.2±0.2
	10	13.8±0.1	14.0±0.2	13.2±0.
		Conductance		
Dataset	m	H-KM	H-EM	ComE
DBLP	2	6.8±4.2	**6.7**±4.4	14.1±15.8
	5	**4.6**±3.4	4.8±3.8	7.4±4.3
	10	6.0±4.6	**5.2**±4.0	6.5±4.2
Wikipedia	2	96.6±3.0	96.6±5.1	**94.9**±3.2
	5	93.7±3.4	93.8±4.4	**92.9**±3.7
	10	91±4.1	**90.5**±4.7	91.4±4.3
BlogCatalog	2	**92.5**±6.0	93.1±8.1	93.4±4.3
	5	88.4±6.6	**87.8**±7.5	87.9±8.8
	10	85.9±7.5	**84.7**±7.8	87.4±8.6
Flickr	2	**93.5**±10.0	94.4±13.0	96.4±2.9
	5	**89.7**±12.8	**89.7**±13.9	91.2±10.6
	10	89.5±12.1	**88.2**±14.5	**88.8**±11.8
		NMI		
Dataset	m	H-KM	H-EM	ComE
DBLP	2	66.1±3.4	**66.2**±5.8	50.5±2.1
	5	**71.3**±2.4	69.7±2.1	63.6±4.4
	10	65.4±8.9	**69.3**±0.6	62.5±0.8
Wikipedia	2	**6.9**±0.8	5.5±2.1	6.5±0.
	5	8.8±0.3	8.6±0.3	**10.1**±0.5
	10	8.6±0.0	8.6±0.1	9.3±0.2
BlogCatalog	2	**4.4**±0.0	4.1±0.0	3.3±0.
	5	**10.4**±0.5	10.1±0.5	**10.5**±0.3
	10	**14.6**±0.2	14.0±0.3	13.7±0.1
Flickr	2	**24.8**±0.6	**24.7**±0.5	21.7±0.
	5	**31.8**±0.1	**31.8**±0.1	29.6±0.2
	10	32.7±0.1	32.7±0.1	**33.0**±0.

4.3 Results discussion

Unsupervised community detection results. Table 3 reports the results related to the unsupervised community detection experiments. For the three corpora DBLP, BlogCatalog and Flickr we obtained better results with Hyperbolic clustering methods with only few exceptions. Although hyperbolic embedding

Table 4: Results for the different supervised classification methods, with : H-B the supervised hyperbolic K-Means based on hyperbolic barycentre; H-GMM the supervised hyperbolic Gaussian mixture model; H-LR the logistic regression in hyperbolic space; *ComE* [13].

Dataset	Dim	H-B	H-GMM	H-LR	*ComE*
DBLP	2	86.6±1.6	**88.7**±1.1	82.9±8.0	67.1±1.3
	5	90.0±0.5	90.9±0.5	**91.2**±0.6	89.5±0.5
	10	90.2±0.5	90.7±0.4	**91.6**±0.5	90.9±0.3
	128	-	-	-	92.0
Wikipedia	2	4.9/1.6/1.0	45.1/28.2/21.5	**47.2/28.7/21.3**	46.9/28.4/21.0
	5	5.1/1.7/1.1	45.1/28.6/21.3	47.4/29.1/21.5	**48.3/29.7/22.0**
	10	8.3/2.8/1.7	44.6/29.1/21.4	47.5/29.6/21.9	**50.0/30.8/22.5**
	128	-	-	-	48.9/30.1/22.1
BlogCatalog	2	3.8/2.8/4.0	**17.5/12.6/10.8**	16.8/12.6/10.7	16.2/12.4/10.7
	5	13.3/5.9/5.5	25.1/15.8/12.6	24.8/16.0/12.6	**26.1/16.3/12.7**
	10	23.0/9.2/7.2	33.1/19.3/14.4	**35.5/20.2/15.0**	34.0/19.7/14.6
	128	-	-	-	42.6/22.3/16.1
Flickr	2	5.9/2.5/1.7	**23.3/14.3/10.8**	18.7/11.2/8.4	20.3/12.1/9.2
	5	12.8/4.8/3.1	26.8/16.5/12.6	**28.3/16.4/12.2**	26.4/15.6/11.8
	10	14.8/5.4/3.5	26.3/16.3/12.7	**31.4/18.3/13.6**	31.0/17.8/13.1

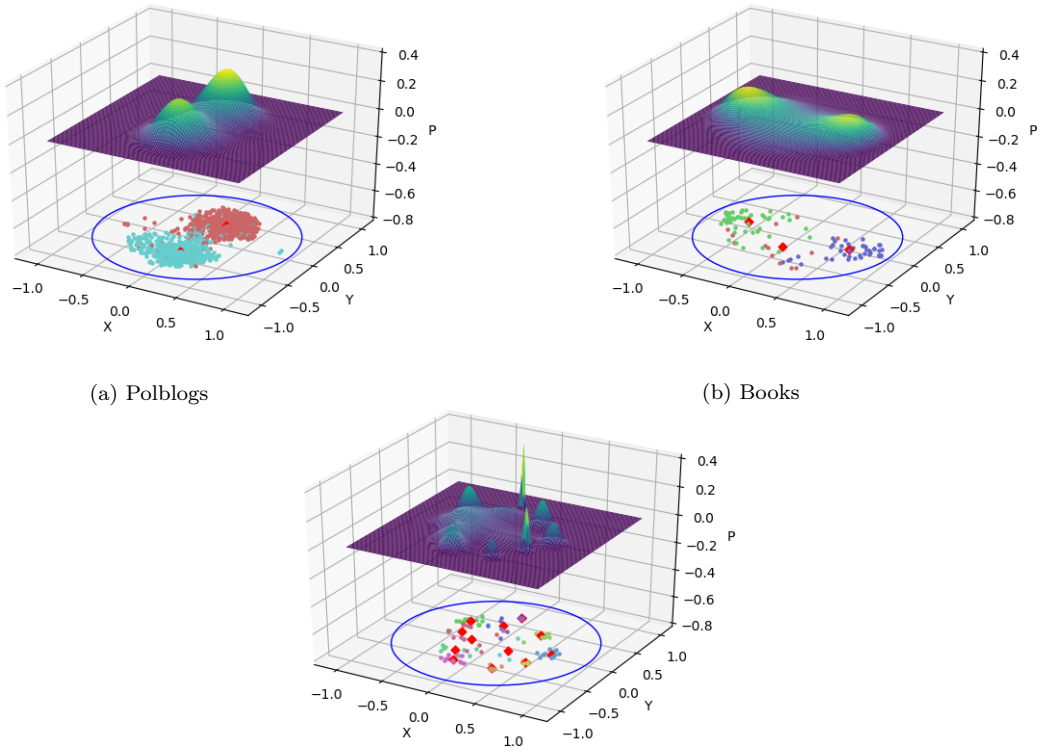
Table 5: Performances obtained by our method compared to Hyperbolic-SVM[16] (H-SVM) for small scale datasets for 2-dimensional embedding. Results are the means for 5-folds cross-validation for 5 experiments

Dataset	H-SVM	H-B	H-GMM	H-LR
Karate	86±3	**92**±11.	91±11.	87±15.
Polblogs	93±1	95±0.9	95±0.9	**96**±1.2
Books	73±4	83±8.3	**83**±7.5	82±7.4
Football	24±3	**80**±7.8	79±8.0	39±11.

seems to perform better than Euclidean embedding, it remains inconclusive which of H-EM or H-KM is better suitable: for DBLP, Wikipedia and Flickr the H-EM approach showed better results, however for *BlogCatalog* K-Means algorithm performed better.

In addition, Figure 10 shows the predicted distributions for *ComE* and H-GMM. When visually comparing these two figures with the real distribution of communities (Figure 7), we state that the obtained H-GMM distributions look closer to the real ones for DBLP and BlogCatalog than *ComE*. However, our framework struggles with the Wikipedia dataset where the true community distribution shows many small communities; this matter is further investigated in the next subsections. We recall that for all datasets except for DBLP each node can belong to many different communities; in the previous figure only the most likely community labels the node.

Supervised community classification results. Table 4 shows that the proposed embedding method often obtains better or similar performances for the different datasets especially in low dimensions. H-GMM is particularly advantageous



(a) Polblogs

(b) Books

(c) Football

Fig. 11: Visualization of Polblogs, Books and Football embeddings with their associated GMM, node colors represent the true known community labels

for *DBLP* and *Flickr* with superior performances when using two dimensional embedding. Using Riemannian logistic regression outperforms the baseline for *Flickr*, *DBLP* in 5 dimensions and reaches similar performances for the remaining datasets, demonstrating the effectiveness of our method to learn community representations. Notice that for *DBLP*, results of *ComE* in 128 (92%) dimensions are comparable to our results in only 10 dimensions. However, for Wikipedia and BlogCatalog, *ComE* remains better in 128 dimensions reaching respectively 49% and 43% of *Precision@1*.

For the small datasets in Table 5, H-B and H-GMM outperform H-SVM and H-LR especially when the number of communities is large (e.g., the *football* dataset). We empirically show that geodesics separators are not always suited for classification problems; this matter is further investigated in Section 4.4. Differences in results between H-SVM and H-LR are mainly due to the quality of the learned embeddings. Indeed, [16] uses embeddings preserving first and second-order proximity only without taking into account community awareness.

Figure 11 displays the learned embeddings with the known true labels of nodes (in colors), barycentres of each community (red squares) and the associated GMM components.

Convergence and sensibility to initialisation. Convergence for node embedding is reached when the value of the loss function becomes quasi-constant. The community detection algorithm is sensitive with respect to the initial parameters used for embedding and initial values of the EM algorithm as well. Relying on previous works, we considered the initialisation suggested by [30]. For the EM algorithm, we perform a K-Means initialisation to deduce initial means while variances are random.

4.4 Suitability of Classification Algorithms to Particular Datasets

In this section, we discuss and give insights regarding the reasons for which a classification algorithm could be suitable or not for a particular dataset. Additional experiments and comparisons are provided to explain why some datasets obtained poor performances regardless of the used classification algorithm. We propose therefore to compare our method with the baseline Most Common Community (MCC, described next). This comparison will illustrate difficulties for classification algorithms to deal with GSD exhibiting imbalanced community distributions when embedded in low dimensions. In particular, this is the case for *Wikipedia* and *Blogcatalog* datasets where some communities are reduced to a single node. Finally, relying on the visualisation of the predictions of the different supervised algorithms, we justify the relevance of using Gaussian distributions in some situations for classification rather than classifiers based on geodesics such as the Logistic Regression.

4.4.1 Comparison with Most Common Community (MCC)

In this section, we comment the results and discuss the reasons for which classification algorithms obtained low performances for two-dimensional representations when applied to some datasets. While both the unsupervised and supervised settings are concerned, we note better scores in the supervised case. The main intuition to explain the performance gap is the tendency of supervised classification methods to annotate all nodes with the most common community in the dataset (i.e., the community that labels the highest number of nodes). To better illustrate this claim, we propose to visualise in Table 6 the Most Common Community baseline (MCC). MCC associates each node with the community withdrawn from the probability distribution of the known true labels and can be seen as a naive classification method. Formally, the

probability vector of the known label $\hat{y}$ is written :

$$\hat{y} = \frac{\sum_{i=1}^{N} y_i}{\sum_{i=0}^{N} \sum_{k=0}^{K} y_{ik}} \tag{17}$$

where $y_{ik} = 1$ if the true label of node v_i is k and 0 if not, and y_i the labelling vector of node v_i (i.e., the k-th component of y_i is 1 if v_i belongs to community k).

Table 6: Precisions at 1, 3, 5 for H-LR (Hyperbolic Logistic Regression) for 2 and 10 dimensional embeddings, MCC-CV (mean of most common community using 5-cross validation sets) and MCC-A (most common community for the entire dataset)

Dataset	H-LR (2D)	H-LR (10D)	MCC-CV	MCC-A
Karate	87	-	53.3	50.0
PolBlogs	96	-	52.0	51.9
Books	82	-	46.6	46.6
Football	39	-	2.6	11.3
DBLP	79	92	38.1	38.0
Wikipedia	47/29/21	47/30/22	47/28/20	47/28/20
BlogCatalog	17/13/11	35/20/15	16/12/10	16/12/10
Flickr	19/11/8	31/18/14	17/10/7	17/10/7

Table 6 empirically shows that in 2 dimensions the H-LR method is unable to efficiently capture the community structures of *Wikipedia* and *BlogCatalog* since it achieves equal performances as MCC. Moreover, for *Wikipedia* the most common community labels nearly half of the graph nodes as shown in Figure 7. Taking a closer look at the same Figure, we notice how the community distributions are largely unbalanced for *Wikipedia* and *BlogCatalog*. Furthermore, for *Wikipedia*, some communities are represented by a single node (see Figure 10e), making this dataset particularly difficult to capture the least represented communities by classifiers. On the contrary, the fact that communities of *DBLP* are more uniformly distributed contributes in achieving better performances with H-LR in only two dimensions.

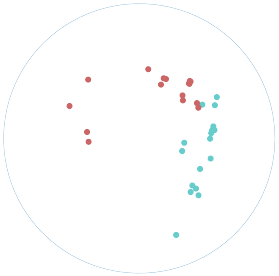
4.4.2 Visualisation

In this part, we provide visualisations of the learned embeddings for different classification algorithms and discuss advantages and disadvantages of each classifier.

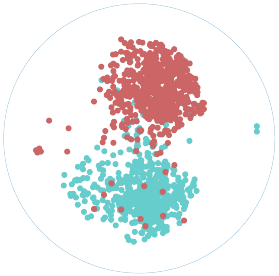
Figure 12 shows the learned community-aware embeddings for the graph datasets *Karate*, *PolBlogs*, *Books*, *Football* and *DBLP* in the two-dimensional hyperbolic space. Each embedded node is represented as a point within the disk.

The colour of each node corresponds to its real community. We can visually see how well the embedding was able to accurately separate the communities by perceiving whether or not colours have been mixed up within a group of clustered points. For instance, the embedding of *Polblogs* (2 communities) shows two clusters each with clear distinct colours while having only a few points not coloured as the rest of the cluster. Therefore, a clustering algorithm applied to this embedding should be able to easily retrieve the communities and should achieve good performances in correctly labelling the nodes.

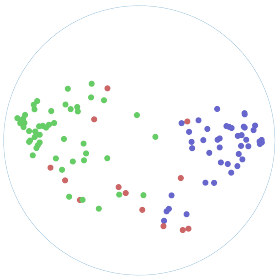
Figures 13 and 14 show the exact same embeddings for each dataset as Figure 12, but colours now represent labels obtained from respectively H-LR and H-GMM. Notice how H-LR struggled to find geodesics separating accurately communities of *Football*. Indeed, the learned two-dimensional embeddings were non-separable using geodesics for *Football*. This difficulty was however better handled by H-GMM. Moreover, H-LR performances on *Football* achieved $\approx 39\%$ while reaching $\approx 80\%$ with H-GMM, thus supporting the benefit of making use of classification with Gaussian distributions.



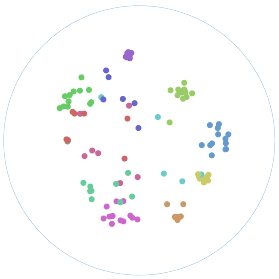
(a) Karate



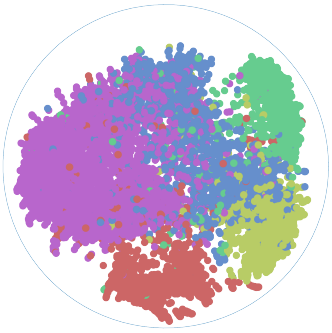
(b) PolBlogs



(c) Books

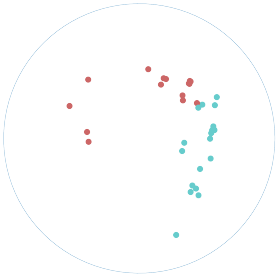


(d) Football

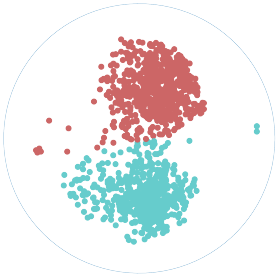


(e) DBLP

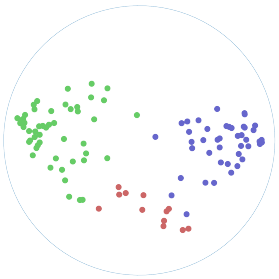
Fig. 12: Embedding visualisations on the Poincaré disk, each point is a graph node, colours represent ground truth communities



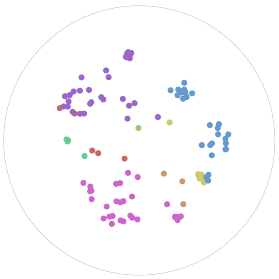
(a) Karate



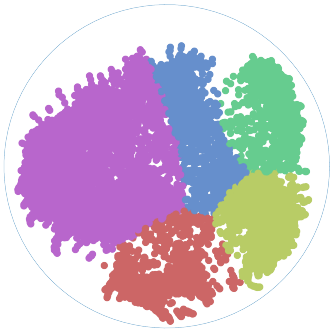
(b) PolBlogs



(c) Books

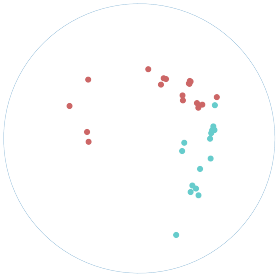


(d) Football

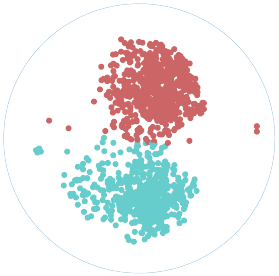


(e) DBLP

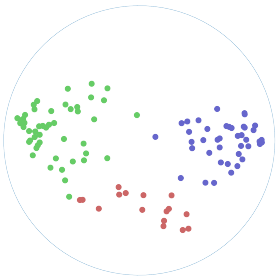
Fig. 13: Prediction obtained using H-LR classification (best view in colour), colours represent community prediction



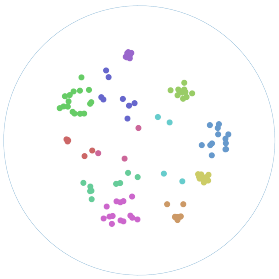
(a) Karate



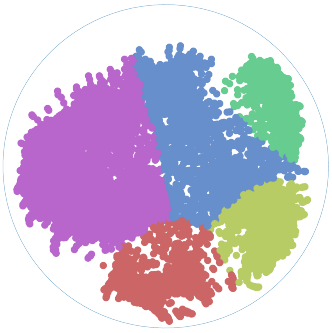
(b) PolBlogs



(c) Books



(d) Football



(e) DBLP

Fig. 14: Prediction obtained using *GMM* classification method (best view in colour), colours represent community prediction.

5 Conclusion

In this paper, we presented a methodology that combines graph embeddings together with clustering techniques on hyperbolic spaces to learn communities. To this end, we designed a framework to learn community-aware embeddings on the Poincaré Ball. To assess the relevance of the proposed hyperbolic approach, we experimented with both supervised and unsupervised applications. We used adaptations to the Riemannian setting of K-Means, Expectation-Maximisation clustering algorithms for unsupervised learning and Riemannian barycentre, Gaussian mixture models and Logistic Regression for the supervised one.

Our proposed framework was compared with the-state-of-art ComE method and with a recent hyperbolic SVM approach. Several metrics were reported and revealed similar or better results for our approach when using the same dimensions as Euclidean spaces except for the Wikipedia dataset.

In upcoming work, we plan to adapt the presented approach to larger dimensional hyperbolic spaces, held back today by the difficulty to efficiently approximate the normalisation factor of the Riemannian Gaussian distributions. We also plan to adapt Bayesian inference criteria to hyperbolic spaces and experiment with estimating the number of communities. Moreover, since Gaussian mixture models used in the ComE approach have the advantage of using full-covariance matrices, we believe that extending hyperbolic GMM to handle full covariance matrices would help improve performances by offering a more expressive mean to capture communities.

By empirically demonstrating the effectiveness of hyperbolic spaces in learning communities on large graphs, we hope to encourage their use in more applications.

References

1. B. Afsari. Riemannian L^p center of mass: existence, uniqueness and convexity. *Proceedings of the American Mathematical Society*, 139(2):655–6673, 2011.
2. D. Alekseevskij, E. B. Vinberg, and A. Solodovnikov. Geometry of spaces of constant curvature. In *Geometry II*, pages 1–138. Springer, 1993.
3. N. Annamalai, C. Mahinthan, V. Rajasekar, C. Lihui, L. Yang, and J. Shantanu. graph2vec: Learning distributed representations of graphs. In *Proceedings of the 13th International Workshop on Mining and Learning with Graphs (MLG)*, 2017.
4. M. Arnaudon, F. Barbaresco, and L. Yang. Riemannian medians and means with applications to radar signal processing. *Journal of Selected Topics in Signal Processing*, 7(4):595–604, 2013.
5. M. Arnaudon, C. Dombry, A. Phan, and L. Yang. Stochastic algorithms for computing means of probability measures. *Stochastic Process and their Applications*, 58(9):1473–1455, 2012.
6. M. Arnaudon and L. Miclo. Means in complete manifolds : uniqueness and approximation. *ESAIM Probability and statistics*, 18:185–206, 2014.
7. M. Arnaudon, L. Yang, and F. Barbaresco. Stochastic algorithms for computing p-means of probability measures, geometry of Radar Toeplitz covariance matrices and applications to HR Doppler processing. In *International Radar Symposium (IRS)*, pages 651–656, 2011.
8. A. Barachant, S. Bonnet, M. Congedo, and C. Jutten. Multiclass brain-computer interface classification by Riemannian geometry. *IEEE Transactions on Biomedical Engineering*, 59(4):920–928, 2012.
9. G. Becigneul and O.-E. Ganea. Riemannian adaptive optimization methods. In *International Conference on Learning Representations (ICLR)*, 2019.
10. M. Boguná, F. Papadopoulos, and D. Krioukov. Sustaining the internet with hyperbolic mapping. *Nature communications*, 1(1):1–8, 2010.
11. S. Bonnabel. Stochastic gradient descent on Riemannian manifolds. *IEEE Transactions on Automatic Control*, 122(4):2217–2229, 2013.
12. S. Cavallari, E. Cambria, H. Cai, K. C. Chang, and V. W. Zheng. Embedding both finite and infinite communities on graphs [application notes]. *IEEE Computational Intelligence Magazine*, 14(3):39–50, 2019.
13. S. Cavallari, V. W. Zheng, H. Cai, K. C.-C. Chang, and E. Cambria. Learning community embedding with community detection and node embedding on graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM)*, pages 377–386. ACM, 2017.
14. B. Chamberlain, M. Deisenroth, and J. Clough. Neural embeddings of graphs in hyperbolic space. In *Proceedings of the 13th International Workshop on Mining and Learning with Graphs (MLG)*, 2017.
15. I. Chami, Z. Ying, C. Ré, and J. Leskovec. Hyperbolic graph convolutional neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 4868–4879. Curran Associates, Inc., 2019.
16. H. Cho, B. DeMeo, J. Peng, and B. Berger. Large-margin classification in hyperbolic space. volume 89 of *Proceedings of Machine Learning Research*, pages 1832–1840. PMLR, 16–18 Apr 2019.
17. P. Cui, X. Wang, J. Pei, and W. Zhu. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 31(5):833–852, 2019.
18. O. Ganea, G. Becigneul, and T. Hofmann. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems 31 (NIPS)*, pages 5345–5355. Curran Associates, Inc., 2018.
19. M. Gromov. *Hyperbolic Groups*, pages 75–263. Springer New York, New York, NY, 1987.
20. A. Grover and J. Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22th ACM International Conference on Knowledge Discovery & Data Mining (SIGKDD)*, pages 855–864, 2016.
21. S. Helgason. *Differential geometry, Lie groups, and symmetric spaces*. American Mathematical Society, 2001.

22. S. Heuveline, S. Said, and C. Mostajeran. Gaussian distributions on riemannian symmetric spaces, random matrices, and planar feynman diagrams. *arXiv preprint arXiv:2106.08953*, 2021.
23. D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguñá. Hyperbolic geometry of complex networks. *Physical Review E*, 82:036106, Sep 2010.
24. A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008.
25. F. Lin and W. W. Cohen. Power iteration clustering. In *Proceedings of the 27th Intenational Conference on Machine Learning (ICML)*, 2010.
26. Q. Liu, M. Nickel, and D. Kiela. Hyperbolic graph neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8230–8241. Curran Associates, Inc., 2019.
27. E. Mathieu, C. Le Lan, C. J. Maddison, R. Tomioka, and Y. W. Teh. Continuous hierarchical representations with poincaré variational auto-encoders. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 12565–12576. Curran Associates, Inc., 2019.
28. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pages 3111–3119. Curran Associates, Inc., 2013.
29. G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
30. M. Nickel and D. Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 6338–6347. Curran Associates, Inc., 2017.
31. I. Ovinnikov. Poincaré Wasserstein autoencoder. In *Bayesian Deep Learning Workshop of Advances in Neural Information Processing Systems (NIPS)*, 2018.
32. X. Pennec. Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127, 2006.
33. J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.
34. B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, KDD '14, pages 701–710, 2014.
35. S. Said, L. Bombrun, Y. Berthoumieu, and J. H. Manton. Riemannian Gaussian distributions on the space of symmetric positive definite matrices. *IEEE Trans. Information Theory*, 63(4):2153–2170, 2017.
36. S. Said, H. Hajri, L. Bombrun, and B. C. Vemuri. Gaussian distributions on Riemannian symmetric spaces: Statistical learning with structured covariance matrices. *IEEE Trans. Information Theory*, 64(2):752–772, 2018.
37. F. Sala, C. D. Sa, A. Gu, and C. Ré. Representation tradeoffs for hyperbolic embeddings. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 4457–4466, 2018.
38. L. T. Skovgaard. A riemannian geometry of the multivariate normal model. *Scandinavian journal of statistics*, pages 211–223, 1984.
39. D. A. Spielmat. Spectral partitioning works: Planar graphs and finite element meshes. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, FOCS '96, page 96. IEEE Computer Society, 1996.
40. J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.
41. C. Tu, X. Zeng, H. Wang, Z. Zhang, Z. Liu, M. Sun, B. Zhang, and L. Lin. A unified framework for community detection and network representation learning. *IEEE Transactions on Knowledge and Data Engineering*, 31(6):1051–1065, 2019.
42. A. A. Ungar. A gyrovector space approach to hyperbolic geometry. *Synthesis Lectures on Mathematics and Statistics*, 1(1):1–194, 2008.

43. I. Vulić, D. Gerz, D. Kiela, F. Hill, and A. Korhonen. HyperLex: A large-scale evaluation of graded lexical entailment. *Computational Linguistics*, 43(4):781–835, Dec. 2017.
44. D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. In *Proceedings of the 22Nd ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1225–1234. ACM, 2016.
45. D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.
46. X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, 2002.

A Implementation

The package available at https://github.com/tgeral68/HyperbolicGraphAndGMM (soon to be released to the public) provides python code for performing graph embedding in the Poincaré Ball $\mathbb{B}^m$ for all dimensions $m \leq 10$ and to apply Riemannian versions of Expectation-Maximisation (EM) and K-Means algorithm as detailed in the paper. The `Readme` details the required dependencies to operate the code, how to reproduce the results of the paper as well as effective ways to run experiments (produce a grid search, evaluate performances and so on). The code uses PyTorch backend with 64-bits floating-point precision (set by default) to learn embeddings. In this section, further details of the procedures are presented in addition to those given in the paper and the `Readme`.

A.1 Centroids initialisation

To initialise both the centroids of the K-Means and the means of the Gaussian mixture models, one can use smart initialisations or random ones. A common way to initialise means or centroids is the K-Means++ algorithm using a smart initialisation instead of a purely random one. The main principle relies on selecting centroids that are far away from each other, consequently improving the initialisation. To this end, before running K-Means or Expectation-Maximisation algorithms, the means or centroids are selected as follow:

1. For the first iteration, choose the first centroid c_1 randomly from the embedded points according to a uniform distribution.
2. For a future iteration t, sample a new centroid c_t according to $p(x|c_1, c_2, ..., c_{t-1})$, that associates a low probability for choosing x if it is associated to one of the existing centroids ($\{c_1, ..., c_{t-1}\}$. Technically, the distribution is implemented by finding for each point the closest centroid in $\{c_1, c_2, \dots, c_t\}$ such that $f(x) = \min_{\{c_1, c_2, ..., c_t\}} d_h(x, c_i)$ then computing for each point $p(x|c_1, c_2, ..., c_{t-1}) = \frac{d_h(x, f(x))^2}{\sum_{z \in \mathcal{D}} d_h(z, f(z))^2}$
3. Repeat until K centroids have been chosen

The K-Means++ is implemented in the framework as a hyper-parameter of K-Means.

A.2 EM Algorithm

- **Weighted Barycentre:** We set for variable λ (learning rate) and ϵ (convergence rate) in Algorithm 1 of the paper the values $\lambda = 5e-2$ and $\epsilon = 1e-4$.
- **Normalisation coefficient :** We compute the normalisation factor of the Gaussian distribution for σ in the interval [1e-3,2] with step size of 1e-3. This parameter is a quite important since if the minimum value of sigma is too high then unsupervised precision is better on datasets for which it is difficult to separate clusters in small dimensions (*Wikipedia*, *BlogCatalog* mainly) as discussed in the previous MCC subsection (In Subsection 4.4.1, the most common community labelled $\approx 47\%$ of the nodes for *Wikipedia* and $\approx 17\%$ for *BlogCatalog*).
- **EM convergence :** In the provided implementation, the EM is considered to have converged when the values of w_{ik} change less than 1e-4 w.r.t the previous iteration, more formally when:

$$\frac{1}{N}\sum_{i=0}^{N}\frac{1}{K}\sum_{k=0}^{K}(|w_{ik}^{t} - w_{ik}^{t+1}|) < 1e-4$$

  For instance, using *Flickr* the first update of *GMM* distribution converged in approximately 50 to 100 iterations.

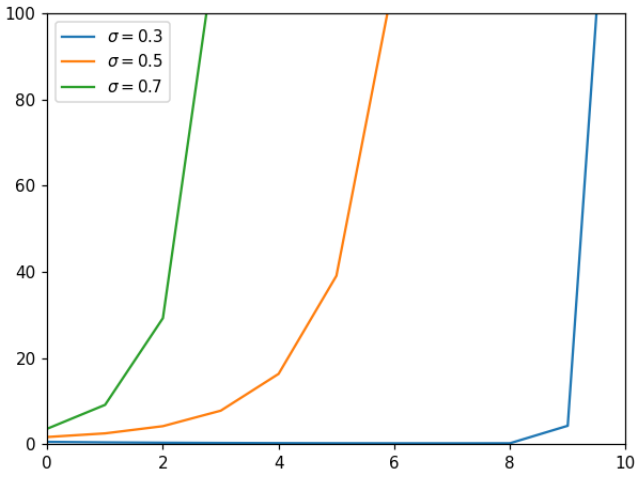


Fig. 15: Normalisation coefficient (y-axis) $\sigma \mapsto \zeta_m(\sigma)$ of Gaussian distributions on Hyperbolic space for different values of the dimension m (x-axis) and of the variance σ.

A.3 Learning embeddings

- **Optimisation :** In some cases, due to the Poincaré ball distance formula, the updates of the form $\text{Exp}_u(\eta \nabla_u f(d(u,v))$ can reach a norm of 1 (because of floating point precision). In this special case we do not take into account the current gradient update. If it occurs too frequently, we recommend to lower the learning rate.
- **Moving context size :** Similarly to *ComE*, we use a moving size window on the context instead of a fixed one; thus we uniformly sample the size of the window between the max size given as input argument and 1.

A.4 Limitations for going in high dimensions

Numerical instabilities when computing the normalisation coefficient $\zeta_m(\sigma)$ in Hyperbolic space. Recall that in Euclidean space, the normalisation coefficient of the multivariate isotropic Gaussian distribution is a linear function of the dimension m: $(2\pi\sigma)^{m/2}$. As for the Poincaré space used in the paper, the expression of the normalisation factor is:

$$\zeta_m(\sigma) = \sqrt{\frac{\pi}{2}} \frac{\sigma}{2^{m-1}} \sum_{k=0}^{m-1} (-1)^k C_{m-1}^k e^{\frac{p_k^2 \sigma^2}{2}} \left(1 + \text{erf}(\frac{p_k \sigma}{\sqrt{2}})\right)$$

Referring to Figure 15, we can observe that $\zeta_m(\sigma)$ increases exponentially as a function of m. Therefore, when increasing the dimension from 1 to 128 the computed probabilities $f(x|\mu,\sigma)$ where

$$f(x|\mu,\sigma) = \frac{1}{\zeta_m(\mu,\sigma)} \exp\left[-\frac{d^2(x,\mu)}{2\sigma^2}\right]$$

become the result of a division with an ever-increasing number causing out-of-bound issues and numerical instabilities. This problem seems to be equally faced by the authors of [27], who adapted variational auto-encoders to the Poincaré space. Their experiments are performed using 20 dimensions at most.

The ComE approach is however capable of reaching high dimensions without dealing with numerical instability issues.

A possible solution to this problem could be to reconsider the mathematical model at higher dimensions, and provide computationally efficient formulas that deals with floating representations. The very recent paper [22] seems to have found new asymptotic formulas of the normalising factor on typical manifolds when the dimension grows and is definitively worth investigating to overcome the current difficulty.