

# Report on Hyperbolic Community detection

October 3, 2019

## 1 Introduction

We plan to adapt the paper "Learning Community Embedding with Community Detection and Node Embedding on Graphs" using Hyperbolic space instead of euclidean one. Today community graph embedding rely on projecting data from a graph (such as adjacency matrix) on an continuous space, and the retrieve community by clustering based algorithms. The projection often involve a small representation space, moreover severall communauties are overlapping each others. In particular hierarchical community can exists, taking it into account can be relevant. If mainly used representation space is euclidean, some others manifolds can be used. Particularly hyperbolic manifolds have prooven their efficiency to embed hierarchical data and performs well in low dimenssion space. In this document we report results and experiments in progress experiments on community detection using hyperbolic riemnmanian manifolds.

## 2 Learning Community Embedding with Community Detection an Node Embedding Graph

Before describing our approach we sum up the main related articles. The paper propose to face Community detection in jointly learning detection process and embedding projection function.

### 2.1 Community Detection embedding

In order to retrieve community, authors propose to associate for each examples embeddings a community probability by  $p(z_i = k)$  the probability of node  $v_i$  to belong to the community indexed by  $k$ . They model the prosteriot  $p(v_i|z, \Theta)$  by a multivariate gaussian distritbution, thus we can write :

$$p(v_i|z_i = k; \phi_i, \psi_k, \Sigma_k) = \mathcal{N}(\phi_i|\psi_k, \Sigma_k)$$

With  $\psi_k, \Sigma_k$  respectively the mean and variance of the gaussian distribution.

### 2.2 Node Embedding

The embeddings are obtained by rapproaching neighbors in terms of graph distance, the first cost function associate is :

$$O_1 = -\alpha_1 \sum_{(v_i, v_j) \in E} \log(\sigma(z_i^t z_j^t))$$

with  $E$  the vertices. Unfortunately nodes at distance one are not always in the same community and nodes at a further distance can be in a same community. Thus the *DeepWalk* algorithm propose to perform a random walk over the graph allowing to find neighbors. The neighborhood of a node  $v_i$  is called context of  $v_i$ , denoted  $C_i$ :

$$O_2 = -\alpha_2 \sum_{v_i \in V} \sum_{v_j \in C_i} \left[ \log(\sigma(z_i^t z_j^t)) + \sum_{t=1, v_l \notin C_i}^T \log(\sigma(z_i^t z_l^t)) \right]$$

Once the model is fit we must ensure that it fit the multivariate gaussian prior. To connect both, the node embedding must take into account detection process, thus they add an additional loss fixing all except the embedding. This is given by :

$$O_3 = -\alpha_3 \sum_{v_i \in V} \log \left[ \sum_{k=1}^K p(z_i = k) p(v_i|z_i = k; \phi_i, \psi_k, \Sigma_k) \right]$$

## 3 Adapting to Hyperbolic

### 3.1 Node Embedding without a priori on Community detection

In this section we plan to find an alternative to  $O_1$  and  $O_2$  loss function because we can not use  $O_1$  and  $O_2$  without considering the poincaré ball distance. To keep the same meaning we can modelize the propabilities of users being in the same community  $P((v_i, v_j) \in C|z_i, z_j)$  we can thus simply modelise it by an exponential distribution :

$$p((v_i, v_j) \in E|z_i, z_j) = \lambda e^{-\lambda - d_h(z_i, z_j)}$$

with  $d_h$  the distance associate to the hyperbolic space. Thus we can use the following loss function considering that items sharing edge often are in the same community :

$$O_{h,1} = -\alpha_1 \sum_{(v_i, v_j) \in E} \log(p((v_i, v_j) \in C|z_i, z_j))$$

With  $\lambda$  a parameter that may be selected by grid search, however we curently use in experiments  $\lambda = 1$ .

Dimension	performance hyperbolic	performance euclidean
2	$74.6 \pm 2.9$	$69.9 \pm 4.5$
3	$81.5 \pm 3.9$	$79.2 \pm 3.2$
4	$82.9 \pm 3.6$	$84.4 \pm 2.2$
5	$86.2 \pm 2.5$	$86.8 \pm 3.0$
10	$88.6 \pm 2.1$	$87.7 \pm 2.9$

Table 1: KMeans football 10 kmeans init

Dimension	performance hyperbolic	performance euclidean
2	$93.2 \pm 1.4$	$85.8 \pm 17.3$
3	$96.1 \pm 1.4$	$92.9 \pm 1.5$
4	$91.7 \pm 7.4$	$94.1 \pm 0.0$
5	$94.1 \pm 0.0$	$94.1 \pm 0.0$
10	$94.1 \pm 0.0$	$94.1 \pm 0.0$

Table 2: KMeans karate 10 kmeans init

We also need to adapt a loss using negative sampling and randomWalks

$$\begin{aligned}
O_{h,2} &= -\alpha_2 \sum_{(v_i, v_j) \in R} \log \left[ \frac{p((v_i, v_j) \in C | z_i, z_j)}{\sum_{v_k \in N_i \cup v_j} p((v_i, v_k) \in E | z_i, z_k)} \right] \\
&= \alpha_2 \sum_{(v_i, v_j) \in R} \log \left[ 1 + \sum_{v_k \in N_i \cup v_j} \lambda e^{-\lambda(d_h(z_i, z_j) - d_h(z_i, z_k))} \right]
\end{aligned}$$

For optimization we use gradient descent with retraction similarly to the Nickel et al paper [?]. We may also use the gradient descent proposed by Wilson and Leimeister [?], showing better convergence on several tasks, similarly we can also use the lorentz model method Kiala et al [?] and then projecting in the poincaré model.

Considering  $f_\theta : x \rightarrow r$  the projection function and  $L(x, y)$  the loss function, the Nickel et al methods update the parameters  $\theta$  by :

$$\theta_{t+1} = \theta_t - \alpha \frac{(1 - \|\theta_t\|^2)^2}{4} \Delta_E$$

With  $\Delta_E$  the gradient of loss function with respect to  $\theta_t$

The second method proposed is based on optimization on the tangent space and then using exponential map to remap gradient on the hyperbolic space.

## 3.2 EM

## 3.3 Connecting both

To connect both embeddings must fit the prior, This is done by minimizing :

$$\begin{aligned}
O_{h,3} &= -\alpha_3 \sum_{v_i \in V} \log \left[ \pi_{ik} \sum_{k=0}^K \frac{1}{\zeta(i)} e^{-\frac{(d_h(x, \mu_k))^2}{2\sigma_k^2}} \right] \\
O'_{h,3} &= -\alpha_3 \sum_{v_i \in V} \sum_{k=0}^K \pi_{ik} \log \left[ \frac{1}{\zeta(i)} e^{-\frac{(d_h(x, \mu_k))^2}{2\sigma_k^2}} \right] \\
O_{h,3} &\leq O'_{h,3}
\end{aligned}$$

# 4 Evaluating EM Algorithm On large dataset

## 5 K-Means results

## 6 KMEANS 40 TEST AND min VARIANCE

In this section we report results of kmeans, with mean performances obtained with 20 run of kmeans and kmeans selected from following criterion:

$$\begin{aligned}
\circ A &= \min_{kmeans} \sum_{i=0}^K \sum_{\{x | \min_k d(x, c_k) = i\}} d(x, c_i)^2 \\
\circ B &= \min_{kmeans} \max_i \left[ \sum_{\{x | \min_k d(x, c_k) = i\}} d(x, c_i)^2 \right]
\end{aligned}$$

The criterion A is the default given in the sklearn python library for kmeans algorithm. We obtain the following results for the different datasets

Dimension	performance hyperbolic	performance euclidean
2	$95.0 \pm 0.1$	$95.2 \pm 0.2$
3	$94.9 \pm 0.1$	$94.9 \pm 0.1$
4	$95.1 \pm 0.0$	$95.1 \pm 0.1$
5	$95.0 \pm 0.1$	$95.2 \pm 0.2$
10	$94.9 \pm 0.2$	$95.1 \pm 0.0$

Table 3: KMeans polblogs 10 kmeans init

Dimension	performance hyperbolic	performance euclidean
2	$72.7 \pm 2.1$	$68.9 \pm 5.7$
3	$71.3 \pm 6.8$	$74.9 \pm 6.9$
4	$74.5 \pm 7.8$	$77.6 \pm 5.0$
5	$75.3 \pm 8.4$	$79.2 \pm 0.6$
10	$78.6 \pm 6.6$	$84.2 \pm 2.1$

Table 4: KMeans dblp 10 kmeans init

Dimension	H-Mean	H-A	H-B	E-Mean	E-A	E-B
KARATE						
2	$96.1 \pm 1.4$	94.1	97.0	$94.1 \pm 0.0$	94.1	94.1
3	$95.5 \pm 1.5$	94.1	97.0	$94.1 \pm 0.0$	94.1	94.1
4	$91.7 \pm 7.4$	94.1	94.1	$94.1 \pm 0.0$	94.1	94.1
5	$91.7 \pm 7.4$	94.1	94.1	$94.1 \pm 0.0$	94.1	94.1
10	$89.4 \pm 9.9$	94.1	94.1	$94.1 \pm 0.0$	94.1	94.1
FOOTBALL						
2	$72.8 \pm 4.3$	78.2	72.1	$65.3 \pm 4.6$	67.8	68.6
3	$81.8 \pm 4.6$	86.9	87.8	$82.8 \pm 3.5$	84.3	87.8
4	$86.1 \pm 3.8$	88.6	87.8	$85.3 \pm 3.4$	87.8	87.8
5	$88.0 \pm 1.9$	90.4	90.4	$87.9 \pm 4.8$	91.3	91.3
10	$88.7 \pm 2.2$	91.3	91.3	$87.4 \pm 4.1$	91.3	86.9
POLBLOGS						
2	$94.8 \pm 0.0$	94.8	94.8	$94.8 \pm 0.0$	94.8	94.8
3	$94.7 \pm 0.0$	94.7	94.8	$95.0 \pm 0.0$	95.0	95.0
4	$95.1 \pm 0.0$	95.1	95.1	$95.1 \pm 0.0$	95.1	95.1
5	$95.1 \pm 0.0$	95.1	95.1	$94.9 \pm 0.0$	94.9	94.9
10	$94.6 \pm 0.0$	94.6	94.6	$94.8 \pm 0.0$	94.8	94.8