Apprentissage automatique sur des données de type graphe utilisant le plongement de Poincaré et les algorithmes stochastiques riemanniens

Hatem Hajri, Hadi Zaatiti, Georges Hébrail, Patrice Aknin

Apprentissage automatique sur des données de type graphe utilisant le plongement de Poincaré et les algorithmes stochastiques riemanniens

Hatem Hajri Hadi Zaatiti Georges Hébrail Patrice Aknin
Institut de recherche technologique SystemX, 8 Avenue de la Vauve, 91120 Palaiseau, France.
Prénom.Nom@irt-systemx.fr

Résumé

Afin de mieux capter la complexité des relations qui existent entre les nœuds d'un graphe binaire, des travaux originaux ont montré l'intérêt de représenter ces données dans des espaces hyperboliques. Ces travaux sont issus d'une première communauté qui s'intéresse aux données graphiques et à leurs représentations et visualisations. D'autre part, une seconde communauté qui adresse plutôt des applications en traitement des données Radar et en vision par ordinateur, a développé ces dernières années des outils d'apprentissage sur certains espaces hyperboliques en exploitant leurs intéressantes propriétés géométriques. Dans cet article, nous présentons nos travaux récents [13, 27] qui visent à rapprocher ces deux approches. Plus précisément, nous combinons les plongements des graphes et les méthodes récentes de partitionnement sur les espaces hyperboliques dans le but de réaliser une classification par apprentissage sur les données initiales du graphe. Nous illustrons cette proposition par des applications en montrant le gain obtenu vis-à-vis de l'état de l'art.

Mots Clef

Plongement de Poincaré, Barycentre Riemannien, Algorithme des K-moyennes Riemannien, Algorithme espérance-maximisation Riemannien.

1 Introduction

L'idée de plonger des données dans un nouvel espace pour en faciliter le traitement a démontré son efficacité dans de multiples applications. Les techniques comme Word2vec [15], Glove [18], Node2vec [12], Graph2vec [16] et DeepWalk [19] sont devenues célèbres grâce à leurs capacité à représenter les données tout en réduisant la complexité de l'espace initial pour les traitements ultérieurs. Des travaux récents comme [17, 10, 23, 24] montrent que les représentations des graphes dans des espaces hyperboliques préservent plus fidèlement la structure sous-jacente ainsi que les relations latentes entre les noeuds d'un graphe que leurs homologues euclidiens.

Dans le but de partitionner les noeuds d'un graphe binaire donné, plusieurs équipes proposent de plonger d'abord le graphe dans un espace euclidien avec les techniques citées (Node2vec, Graph2vec, DeepWalk) et ensuite d'appliquer des méthodes classiques de partitionnement comme les algorithmes des K-moyennes et d'espérance-maximisation (EM) [28, 25, 26, 9, 28]. Nos articles récents [13, 27] proposent des versions hyperboliques de ces travaux, motivés en grande partie par les bons résultats obtenus en traitement de données Radar [3] et en vision par ordinateur [7, 21, 22]. En particulier [7] utilise le barycentre riemannien sur l'espace des matrices de covariances pour classifier des signaux EEG. [3] utilise la médiane riemannienne sur le disque de Poincaré pour détecter les données aberrantes parmi des signaux Radar. [21] introduit l'algorithme EM sur l'espace des matrices de covariance et l'applique au problème de classification des images. Ces applications se basent sur les propriétés intéressantes des espaces hyperboliques qui seront rappelées dans ce papier.

L'article est organisé comme suit. La section 2 présente un rappel sur la géométrie riemannienne du disque de Poincaré comme un modèle typique de géométrie hyperbolique et introduit la version riemannienne des algorithmes des K-moyennes et d'espérance-maximisation. La section 3 rappelle la méthode récente du plongement des données de type graphe dans le disque de Poincaré [17]. Nous complétons ensuite cette section par deux propositions pour l'apprentissage supervisé et non supervisé sur les graphes. La section 4 présente des expérimentations démontrant l'intérêt des approches proposées tout en les comparant avec l'état de l'art récent. Enfin la section 5 conclut l'article en esquissant les perspectives de ce travail.

2 Apprentissage statistique riemannien sur le disque de Poincaré

Dans cette section, nous introduisons les outils mathématiques nécessaires qui seront utilisés pour la partie apprentissage automatique. Nous commençons par rappeler la géométrie du disque de Poincaré. Ensuite, nous présentons deux extensions des algorithmes K-moyennes et espérance maximisation sur cet espace.

Le disque de Poincaré est l'espace noté $\mathbb{D} = \{z \in \mathbb{C} : |z| < 1\}$, muni de la distance dite de Poincaré $d(z_0, z_1)$ définie par :

$$\frac{1}{2}\log\left(\left(1+\left|\frac{z_1-z_0}{1-\bar{z_0}z_1}\right|\right)\left(1-\left|\frac{z_1-z_0}{1-\bar{z_0}z_1}\right|\right)^{-1}\right)$$

Par ailleurs, rappelons qu'une transformation de Möbius

est une application de la forme $f_{a,b} : z \in \mathbb{D} \mapsto \frac{az+b}{\overline{b}z+\overline{a}}$ où $a, b \in \mathbb{C}$ et $|a|^2 - |b|^2 = 1$. $\mathcal{M}$, l'ensemble de toutes les transformations de Möbius est un groupe pour la composition qui agit sur $\mathbb{D}$ transitivement (i) et isométriquement (ii) :

(i) pour tous $z_1, z_2 \in \mathbb{D}$, il existe $f_{a,b} \in \mathcal{M}$ t.q $f_{a,b}(z_1) = z_2$

(ii) pour tous $z_1, z_2 \in \mathbb{D}$, $d(f_{a,b}(z_1), f_{a,b}(z_2)) = d(z_1, z_2)$

Muni du groupe $\mathcal{M}$, $\mathbb{D}$ devient un espace riemannien symétrique homogène de courbure négative [14].

Une propriété importante satisfaite par ces espaces est l'existence et l'unicité du barycentre riemannien (aussi appelé moyenne riemannienne ou moyenne de Fréchet). Plus précisément, pour tout ensemble $\{z_i, 1 \leq i \leq n\}$ inclus dans $\mathbb{D}$, la solution du problème d'optimisation suivant (barycentre riemannien)

$$\hat{z}_n = \text{argmin}_{z \in \mathbb{D}} \left(\sum_{i=1}^{n} d^2(z, z_i) \right) \quad (1)$$

existe, est unique et appartient à $\mathbb{D}$ [2]. $\hat{z}_n$ peut être approché numériquement par une extension au cadre riemannien de la méthode de descente de gradient classique. Ceci a fait l'objet de plusieurs travaux comme [8, 4, 6, 5, 3]. Dans la suite, nous allons utiliser l'algorithme (1) proposé dans [3]. Les notations $\exp_z$ et $\log_z$ désignent respectivement les fonctions exponentielle et logarithme riemanniennes.

Algorithm 1 Calcul approché du barycentre riemannien sur $\mathbb{D}$ par descente de gradient

Entrée : $Z = \{z_i, 1 \leq i \leq n\}$: sous ensemble de $\mathbb{D}$, z_{init} : initialisation du barycentre, τ : pas de la descente de gradient, ε : précision d'arrêt

Sortie : $\hat{z}$: approximation du barycentre riemannien

$\hat{z} \leftarrow z_{init}$

$d = 1e^6$ ▷ un nombre assez large

while $d > \varepsilon$ **do**

$$\mu \leftarrow \frac{2}{n} \sum_{i=1}^{n} \log_{\hat{z}}(z_i), \hat{z} \leftarrow \exp_{\hat{z}}(\tau\mu), d \leftarrow \sqrt{\frac{|\mu|^2}{(1-|\hat{z}|^2)^2}}$$

end while **return** $\hat{z}$

Dans les deux paragraphes suivants, nous présentons les algorithmes des K-moyennes et espérance-maximisation riemanniens qui exploitent les propriétés intéressantes de l'espace de Poincaré.

2.1 Algorithme des K-moyennes riemannien

Par analogie avec le cas euclidien, nous pouvons étendre l'algorithme des K-moyennes au cadre riemannien. Cette extension est basée sur l'observation que pour un ensemble $\{x_1, \cdots, x_n\}$, la moyenne $\hat{x}_n = \frac{1}{n}\sum_i x_i$ correspond aussi au minimum global de la fonction $\sum_{i=1}^{n}(x - x_i)^2$. Dans la version riemannienne, la distance euclidienne sera remplacée par son homologue riemannien.

Algorithm 2 K-moyennes riemannien sur $\mathbb{D}$

Entrée : K : nombre de classes, Z : un ensemble de n complexes appartenant à $\mathbb{D}$

Sortie : N : ensemble de K barycentres, $labels$: n étiquettes des données d'entrée Z

Initialisons K barycentres, $N = \{\nu_1, \nu_2, ..., \nu_K\}$ aléatoirement dans $\mathbb{D}$

repeat

  for $z_i \in Z$ **do** $c_i \leftarrow argmin_{j \in \{1,...,K\}} d(z_i, \nu_j)$

  for $j \in \{1, ..., K\}$ **do**

    $\nu_j \leftarrow$ `Barycentre_Riemannien`$(\{z_i | c_i = j\})$

  end for

until les barycentres deviennent stables

return $N, labels = \{c_i\}$

2.2 Algorithme EM riemannien

En exploitant les propriétés de la distance de Poincaré, il est possible d'étendre l'algorithme EM classique à cet espace [22]. Cette extension s'appuie sur une version riemannienne de la loi gaussienne.

Étant donné deux paramètres $\bar{z} \in \mathbb{D}$ (une position) et $\sigma > 0$ (un écart type), la loi gaussienne $G(\bar{z}, \sigma)$ dépendant de $(\bar{z}, \sigma)$ est définie dans [22] par sa densité

$$p(z|\bar{z}, \sigma) = \frac{1}{\zeta(\sigma)} \exp\left[-\frac{d^2(z, \bar{z})}{2\sigma^2} \right]$$

relativement au volume riemannien $dv(z)$. Une propriété clé de cette distribution, découlant de la symétrie de $\mathbb{D}$, est que $\zeta(\sigma)$ ne dépend pas de $\bar{z}$ (tout comme la loi gaussienne classique). Ceci permet d'en déduire les estimateurs du maximum de vraisemblance (EMV) de $\bar{z}$ et σ étant donné un échantillon $z_1, \cdots, z_N$ de $G(\bar{z}, \sigma)$ comme suit.

EMV de $\bar{z}$. L'EMV de $\bar{z}$ noté $\hat{z}_N$ est le barycentre riemannien empirique de $z_1, \cdots, z_N$.

EMV de σ. L'EMV de σ est $\hat{\sigma}_N = \Phi(\frac{1}{N}\sum_{n=1}^{N} d^2(\hat{z}_N, z_n))$ où Φ est l'inverse de la fonction $\sigma \mapsto \sigma^3 \times \frac{d}{d\sigma} \log \zeta(\sigma)$

En pratique, comme $\hat{z}_N$, $\hat{\sigma}_N$ n'a pas d'expression explicite mais peut être approché numériquement par des méthodes de Newton. Considérons maintenant un modèle de mélange gaussien

$$p(z|(\varpi_\mu, \bar{z}_\mu, \sigma_\mu)_{1 \leq \mu \leq M}) = \sum_{\mu=1}^{M} \varpi_\mu \times p(z|\, \bar{z}_\mu, \sigma_\mu)$$

où les ϖ_μ sont positifs de somme 1. Les paramètres du modèle $\varpi_\mu, \bar{z}_\mu, \sigma_\mu$ peuvent être estimés par une extension de l'algorithme EM classique [22]. Pour ceci, introduisons pour tout $\vartheta = \{(\varpi_\mu, \bar{z}_\mu, \sigma_\mu)\}$,

$$\omega_\mu(z_n, \vartheta) = \frac{\varpi_\mu \times p(z_n|\, \bar{z}_\mu, \sigma_\mu)}{\sum_{s=1}^{M} \varpi_s \times p(z_n|\, \bar{z}_s, \sigma_s)}$$

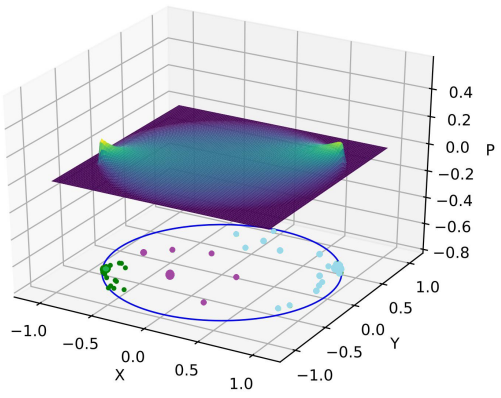


FIGURE 1 – Application de l'algorithm EM riemannien au dataset Polbooks à trois classes (présenté dans la section 4) avec une représentation 3D des densités correspondantes

et

$$N_\mu(\vartheta) = \sum_{n=1}^{N} \omega_\mu(z_n)$$

L'algorithme EM riemannien met à jour $\hat{\vartheta} = \{(\hat{\varpi}_\mu, \hat{Y}_\mu, \hat{\sigma}_\mu)\}$ de façon itérative pour générer l'EMV de $\vartheta = (\varpi_\mu, \bar{Y}_\mu, \sigma_\mu)$ de la manière suivante.

► Mise à jour de $\hat{\varpi}_\mu$: $\hat{\varpi}_\mu = N_\mu(\hat{\vartheta}) \Big/ N$.

► Mise à jour de $\hat{z}_\mu$ (en utilisant une descente de gradient riemannienne) :

$$\hat{z}_\mu = \text{argmin}_z \sum_{n=1}^{N} \omega_\mu(z_n, \hat{\vartheta})\, d^2(z, z_n)$$

► Mise à jour de $\hat{\sigma}_\mu$ (en utilisant une méthode de Newton) :

$$\hat{\sigma}_\mu = \Phi\big(\, N_\mu^{-1}(\hat{\vartheta}) \times \textstyle\sum_{n=1}^{N} \omega_\mu(z_n, \hat{\vartheta})\, d^2(\hat{z}_\mu, z_n)\big)$$

La figure 1 présente une visualisation d'un partitionnement par l'algorithme EM riemannien du dataset Polbooks à trois classes (présenté dans la section 4).

3 Apprentissage sur les graphes en utilisant le plongement de Poincaré

Dans cette section, nous présentons une approche récente [17] qui propose le plongement d'un graphe binaire dans un disque de Poincaré. Ensuite, nous proposons deux algorithmes de partitionnement des noeuds d'un graphe dans les cas supervisé et non supervisé.

Considérons un graphe $(\mathcal{V}, \mathcal{E})$ où $\mathcal{V}$ est l'ensemble des noeuds et $\mathcal{E}$ est l'ensemble des arrêtes. Pour plonger $\mathcal{V}$ dans $\mathbb{D}$, [17] propose d'apprendre une application $u \in \mathcal{V} \mapsto \theta_u \in \mathbb{D}$ minimisant la fonction

$$\mathcal{L}(\Theta) = \log(\sigma(-d(\theta_u, \theta_v))) + \sum_{i=1}^{M} \mathbb{E}_{P_\mathcal{N}}[\log(\sigma(d(\theta_{u_i}, \theta_v)))]$$

où $u_1, \cdots, u_M$ est un ensemble d'échantillons négatifs tirés suivant une distribution $P_\mathcal{N}$ et $\sigma(x) = (1 + e^{-x})^{-1}$ est la fonction softmax [15]. En minimisant cette fonction, θ_u et θ_v se rapprochent l'un de l'autre et les noeuds des échantillons négatifs s'éloignent de θ_v. En pratique $\mathcal{L}(\Theta)$ est optimisé en générant des noeuds sur le graphe à l'aide de l'algorithme DeepWalk [19] puis en tirant des échantillons négatifs suivant la distribution unigram à la puissance $3/4$ [15].

Étant donne un cluster $\mathcal{C} = \{z_i, 1 \leq i \leq n\}$ dans $\mathbb{D}$, ayant comme barycentre $\hat{z}$, on définit sa variance empirique par

$$\text{Var}(\mathcal{C}) = \frac{1}{n} \sum_{i=1}^{n} d^2(\hat{z}, z_i)$$

L'algorithme 3 ci-dessous présente notre schéma pour effectuer un partitionnement non supervisé sur un graphe. L'idée est de générer plusieurs plongements de Poincaré et de garder le plongement le plus concentré après déroulement d'un algorithme K-moyennes. Pour notre travail, nous allons considérer comme meilleur, le plongement ayant la variance minimale (d'autres métriques sont possibles). Dans ce qui suit, la fonction `Poincare_Plongement` plonge un graphe sur le disque de Poincaré en minimisant $\mathcal{L}(\Theta)$.

Cet algorithme peut être suivi aussi par un algorithme EM riemannien pour obtenir une classification plus robuste tenant compte de la dispersion des points au sein des clusters.

L'algorithme 4 ci-dessous présente le schéma proposé pour l'apprentissage supervisé. Bien que cet algorithme puisse être alimenté par l'algorithme 3, une version simple et indépendante de celui ci est présentée ici.

La seconde alternative proposée réalise le plongement de la vérité de terrain selon un mélange gaussien en utilisant l'algorithme EM dans un premier temps et associant par la suite une donnée de l'ensemble de test à une classe en utilisant la règle de Bayes (voir règle (69) dans [21]).

4 Expérimentations

Dans cette section, nous illustrons l'intérêt des approches proposées précédemment par quelques applications en apprentissage supervisé et non supervisé sur des graphes de référence. Nous montrons le gain apporté vis-à-vis de l'état de l'art en nous concentrant sur l'algorithme des K-moyennes. Les datatsets utilisés sont publics [20] et leurs caractéristiques sont données dans le tableau suivant (le nombre de nœuds, arrêtes et classes est désigné respectivement par Nœuds, Liens et K).

4.1 Apprentissage non supervisé

Dans une première série d'expériences, nous avons utilisé les 5 premiers datasets donnés dans le tableau précédent. Pour chaque dataset, nous avons généré 10 marches aléatoires sur le graphe en suivant DeepWalk [19]. Sur chaque

Algorithm 3 Apprentissage non supervisé

Entrée : $\mathcal{G}$: graphe binaire, K : nombre de clusters (entier), NE : nombre d'expérimentations

Sortie : $Meilleur\_Plongement$: Plongement des noeuds du graphe d'entrée ayant la variance totale minimale, N : barycentres des K clusters, $labels$: étiquette de chaque noeud

repeat

$$Plongement_i \leftarrow \texttt{Poincare\_Plongement}(\mathcal{G})$$

$$N_i, labels_i \leftarrow \texttt{K\_moyennes}(K, Plongement_i)$$

$$Clusters \leftarrow \{c_1, ..., c_K\}$$

tel que $c_j = \{z_q \in Plongement_i | labels_i[q] = j\}$

$$var_i = max_{j \in \{1,...,K\}}(Var(c_j))$$

until NE plongements sont effectuées

$$Meilleur\_Plongement \leftarrow Plongement_{meilleur}$$

tel que $meilleur = argmin_{i \in \{1,...,NE\}}(var_i)$

return

$Meilleur\_Plongement, N_{Meilleur}, labels_{Meilleur}$

Algorithm 4 Apprentissage supervisé

Entrée : $\mathcal{G}$: graphe binaire, VT : une vérité terrain pour chaque noeud de $\mathcal{G}$

Sortie : $Cluster$: cluster d'appartenance de chaque noeud des données d'apprentissage

$Plongement \leftarrow \texttt{Poincare\_Plongement}(\mathcal{G})$

$$DE, DT \leftarrow \texttt{Diviser}(Plongement)$$

▷ DE sont les données d'entraînements

▷ DT sont les données de tests

$$\{c_1, ..., c_K\} \leftarrow \texttt{Calculer\_Clusters}(DE, VT)$$

for $q \in \{1, ..., K\}$ **do**

  $\nu_q \leftarrow \texttt{Barycentre\_Riemannien}(c_q)$

end for

for $u \in DT$ **do**

$$Cluster(u) \leftarrow argmin_{p \in \{1,...,K\}}(d(\nu_p, u))$$

▷ d est la distance riemannienne

end for

return $Cluster$

marche aléatoire, nous générons un plongement de Poincaré et un plongement euclidien dans $\mathbb{R}^{10}$ (en utilisant la version euclidienne de l'algorithme (3)). Ensuite, nous appliquons à chaque plongement hyperbolique (resp. euclidien), un algorithme des K-moyennes hyperbolique (resp. euclidien). Dans les deux cas, nous sélectionnons comme meilleur plongement celui ayant une variance totale minimale au sens de l'algorithme (3). Enfin, nous évaluons la performance des meilleurs plongements dans les deux cas ainsi que la performance moyenne sur les 10 tests effectués. Les résultats sont donnés dans le tableau 2 en utilisant les abréviations suivantes :

• MPP : meilleur plongement de Poincaré
• MPE : meilleur plongement euclidien
• 10MP : moyenne sur les 10 plongements de Poincaré
• 10ME : moyenne sur les 10 plongements euclidiens

Le tableau 2 montre que suivre un plongement par un algorithme des K-moyennes est plus avantageux dans le cas hyperbolique que dans le cas euclidien. La figure 2 montre l'évolution croissante de la performance MPP avec le nombre d'expérimentation effectués NE pour le dataset Football et illustre ainsi l'intérêt de générer plusieurs plongements.

En ce qui concerne l'exploitation des plongements de Poin-

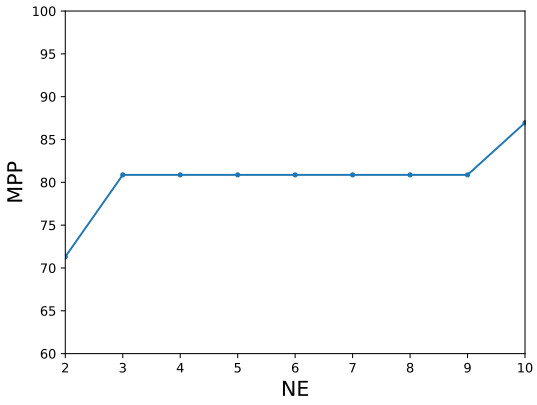


FIGURE 2 – Courbe représentant la variation du MPP en fonction de NE (nombre d'expérimentations effectuées)

Datasets	Noeuds	Liens	K
Karate	34	77	2
Polblogs	1224	16781	2
Polbooks	105	441	3
Football	115	613	12
Adjnoun	112	425	2
Mammifères	1179	6541	NA

TABLE 1 – Caractéristiques des graphes utilisés.

caré pour le partitionnement des noeuds, remarquons que le meilleur plongement de Poincaré (MPP) tel que défini précédemment, n'obtient pas nécessairement la meilleure performance. Pour un seul dataset, la performance moyenne (10MP) obtenue est légèrement supérieure à MPP. Ceci peut être expliqué par l'existence de données aberrantes (des personnes pour lesquelles il est difficile de prédire une classe). L'avantage du meilleur plongement hyperbolique est remarquable pour l'exemple Football (+17%) et pour l'exemple Polbooks (+4.8%). En comparaison avec l'état de l'art récent, nos résultats dépassent ceux obtenus dans [1] par un plongement sur une surface généralisée. Les datasets considérés dans [1] sont **Polbooks, Football, Adjnoun** avec des taux de réussite respectifs de 75%, 77% et 51% respectivement. Les améliorations pour ces exemples en utilisant notre méthode sont significatives : **+9.8%**, **+10%** et **+0.8**%.

La figure 3 montre des visualisations des clusters calculés en utilisant le meilleur plongement de Poincaré (MPP) pour chaque dataset. Chaque cluster est représenté par une couleur et le barycentre est symbolisé par un carré.

Comme deuxième application, nous avons considéré un exemple de données hiérarchiques (sous forme d'arbre) qui est le sous-arbre des mammifères extrait de Wordnet traité comme un graphe. La figure 4 montre les clusters obtenus quand $K = 6$. Les barycentres sont représentés par des carrés. Enfin, la figure 5 montre explicitement certaines étiquettes des noeuds choisies aléatoirement. Un focus est mis sur les noeuds proches des barycentres d'une part et sur les frontières entre les différents clusters d'autre part.

Remarquons que les clusters calculés permettent de discerner entre différents types de mammifères. Par exemple le cluster bleu contient en majorité la famille des canidés alors que le cluster orange contient des mammifères de taille plus importante (lion, tigre, éléphant,...).

4.2 Apprentissage supervisé

Dans cette partie, nous exploitons la notion du barycentre riemannien pour effectuer de l'apprentissage supervisé sur les graphes en suivant l'algorithme 4. Pour évaluer la méthode, nous avons subdivisé chaque dataset de la liste précédente (wordnet exclu) en 5 parties : $4/5$ pour l'entraînement et $1/5$ pour le test. Après plongement de tout le graphe, chaque élément de l'ensemble test est associé à un cluster suivant la règle du plus proche barycentre. Nous avons répété cette expérience 5 fois en permutant les ensembles tests par validation croisée. La même expérience (5 permutations de l'ensemble test) a été refaite 10 fois. Finalement, nous avons évalué la performance moyenne en utilisant la vérité terrain des données tests. Le tableau 2 présente les résultats obtenus. On utilisera l'abréviation suivante :

10VC : Performance moyenne par validation croisée sur les 10 expériences effectuées.

Nous allons maintenant comparer nos résultats avec ceux de [11] qui utilise une généralisation de la méthode SVM au disque de Poincaré. [11] considère les datasets **Karate, Polbooks, Football, Polblogs** et obtient des taux de réussite respectifs de $86\%, 73\%, 24\%, 93\%$ sur 5 expérimentations par validation croisée sur 5 plongements différents selon [10]. Nous avons obtenu des améliorations significatives pour les datasets Karate, Polbooks et Football de **+7.9%**, **+10.3%**, de **+53.9**% avec une légère baisse de performance de -0.6% pour le dataset Polblogs.

5 Conclusion

Dans cet article, nous avons présenté des algorithmes d'apprentissage sur les graphes basés sur les plongements de Poincaré et des extensions au cadre riemannien des algorithmes des K-moyennes et d'espérance-maximisation. Nous avons illustré l'intérêt des algorithmes des K-moyennes par des expérimentations en montrant le gain apporté vis-à-vis de l'état de l'art.

D'autre part, nous avons présenté les algorithmes EM et esquissé quelques exemples de leurs applications en lien avec les plongements de Poincaré. Notons qu'avec la notion de variance totale minimale introduite dans ce papier, il est toujours possible d'augmenter le nombre de plongements (fixé à 10 dans l'article) et aussi la dimension de l'espace de Poincaré (fixé à 2 dans l'article). Ce raisonnement est possible grâce aux bonnes propriétés des variétés hyperboliques et est complètement non supervisé dans le sens où il ne nécessite aucune vérité terrain. Par conséquent, des améliorations des résultats présentés dans ce papier sont toujours possibles. Les résultats obtenus avec le plongement euclidien utilisant DeepWalk suggèrent que pour obtenir de bonnes performances avec cette approche (ou autres variétés comme Graph2vec [16], Node2vec [12]), il faudrait prendre des espaces euclidiens avec des dimensions très grandes.

En perspective, nous envisageons d'étudier la complexité des algorithmes riemanniens présentés dans cet article. Nous comptons aussi utiliser ces travaux pour étendre les résultats de [9, 28] qui propose d'apprendre les communautés sur les graphes en utilisant les lois de mélange gaussien (euclidien). Notre proposition sera d'étendre cela à une version géométrique. Plus précisément au lieu de générer plusieurs plongements et en choisir un à la fin, il sera possible d'optimiser le plongement et d'entraîner le partitionnement par le mélange gaussien simultanément. Enfin, nous souhaitons étendre les travaux à l'apprentissage

	Apprentissage non-supervisé					Apprentissage supervisé	
Datasets	**MPP**	**MPE**	**10MP**	**10ME**	[1]	**10VC**	[11]
Karate	**91.2**%	70.6%	**91.4**%	65.8%	–	**93.9**%	86%
Polblogs	**92.8**%	51.9%	**92.5**%	53.5%	–	**92.4**	**93**%
Polbooks	**84.8**%	77.1%	**80**%	62%	75%	**83.3**%	73%
Football	**87%**	67.8%	**69.4**%	56.8%	77%	**77.9** %	24%
Adjnoun	51.8%	51.8 %	**52.5**%	51.6%	51%	**57.8** %	-

TABLE 2 – Tableau comparatif des performances obtenues pour les algorithmes d'apprentissage supervisé et non-supervisé. Les meilleurs résultats sont en gras.

en ligne et adresser le problème des tests statistiques basés sur les lois gaussiennes et leurs applications sur les données graphiques et les données texte.

Références

[1] M. Aalto and N. Verma. Metric learning on manifolds. *CoRR*, https ://arxiv.org/abs/1902.01738, 2018.

[2] B. Afsari. Riemannian L^p center of mass : existence, uniqueness and convexity. *Proc. Amer. Math. Soc.*, 139(2) :655–6673, 2011.

[3] M. Arnaudon, F. Barbaresco, and L. Yang. Riemannian medians and means with applications to radar signal processing. *J. Sel. Topics Signal Processing*, 7(4) :595–604, 2013.

[4] M. Arnaudon, C. Dombry, A. Phan, and L. Yang. Stochastic algorithms for computing means of probability measures. *Stoch. Proc. Appl.*, 58(9) :1473–1455, 2012.

[5] M. Arnaudon and L. Miclo. Means in complete manifolds : uniqueness and approximation. *ESAIM Probability and statistics*, 18 :185–206, 2014.

[6] 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.

[7] A. Barachant, S. Bonnet, M. Congedo, and C. Jutten. Multiclass brain-computer interface classification by Riemannian geometry. *IEEE Trans. Biomed. Engineering*, 59(4) :920–928, 2012.

[8] S. Bonnabel. Stochastic gradient descent on Riemannian manifolds. *IEEE Trans. Autom. Control.*, 122(4) :2217–2229, 2013.

[9] S. Cavallari, V. W. Zheng, H. Cai, K. 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 2017, Singapore, November 06 - 10, 2017*, pages 377–386, 2017.

[10] B. P. Chamberlain, J. Clough, and M. P. Deisenroth. Neural embeddings of graphs in hyperbolic space. *13th international workshop on mining and learning with graphs. Available at https ://arxiv.org/abs/1705.10359*, 2017.

[11] H. Cho, B. Demeo, J. Peng, and B. Berger. Large-margin classification in hyperbolic space. *CoRR*, abs/1806.00437, 2018.

[12] A. Grover and J. Leskovec. node2vec : Scalable feature learning for networks. *KDD : proceedings. International Conference on Knowledge Discovery & Data Mining*, 2016 :855–864, 2016.

[13] H. Hajri, H. Zaatiti, and G. Hébrail. Learning graph-structured data using poincaré embeddings and riemannian k-means algorithms. *CoRR*, abs/1907.01662, 2019.

[14] S. Helgason. *Differential geometry, Lie groups, and symmetric spaces*. American Mathematical Society, 2001.

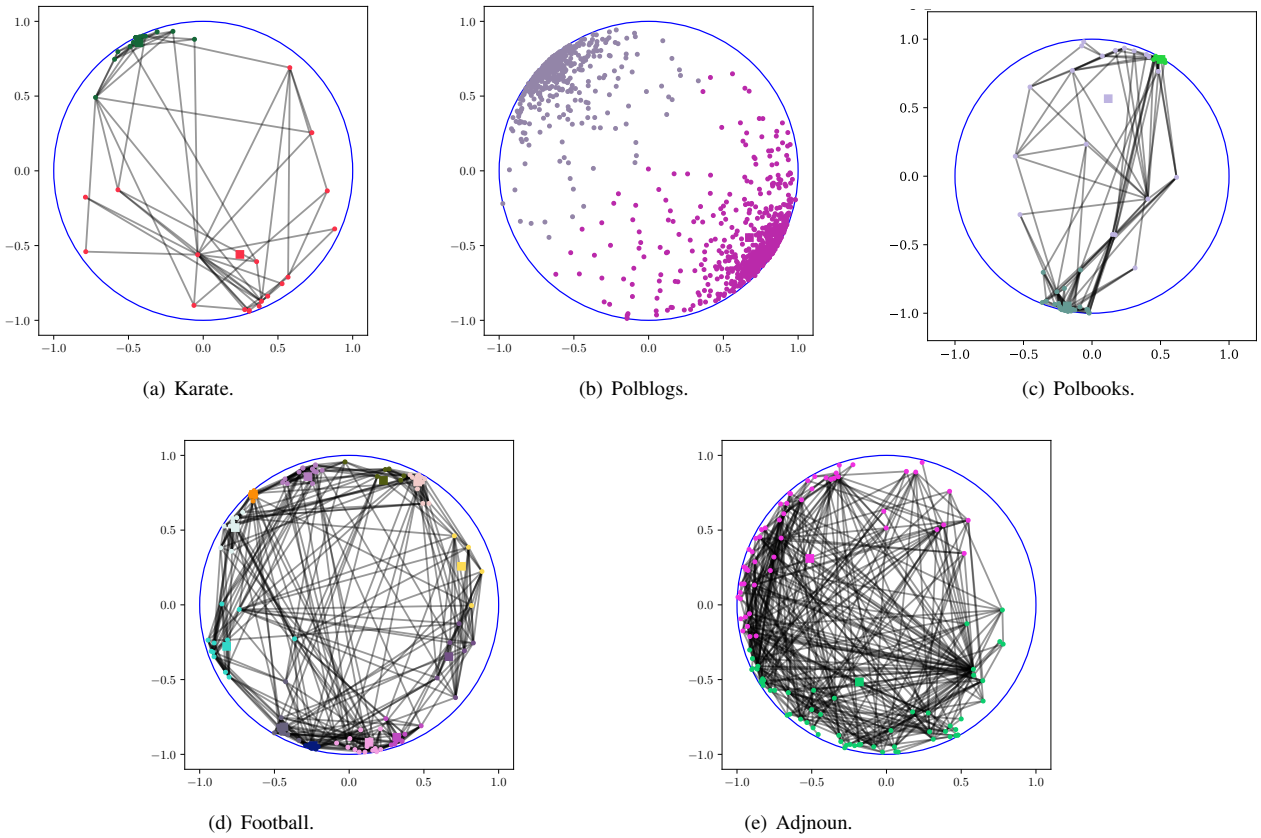
[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[16] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal. graph2vec : Learning distributed representations of graphs. *CoRR*, abs/1707.05005, 2017.

[17] M. Nickel and D. Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems 30*, pages 6338–6347. Curran Associates, Inc., 2017.

[18] J. Pennington, R. Socher, and C. D. Manning. Glove : Global vectors for word representation. In *In EMNLP*, 2014.

[19] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk : Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM.



(a) Karate. (b) Polblogs. (c) Polbooks.

(d) Football. (e) Adjnoun.

FIGURE 3 – Visualisation des clusters dans le meilleur plongement de Poincaré

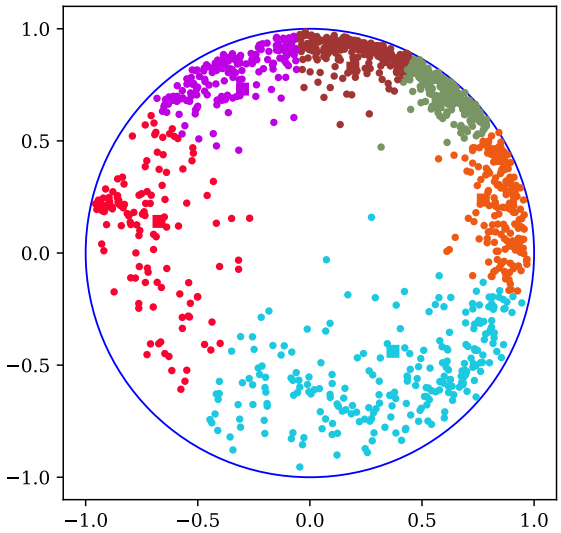


FIGURE 4 – Partitionnement du sous-arbre des mammifères en 6 clusters par l'algorithme des K-moyennes riemannien.

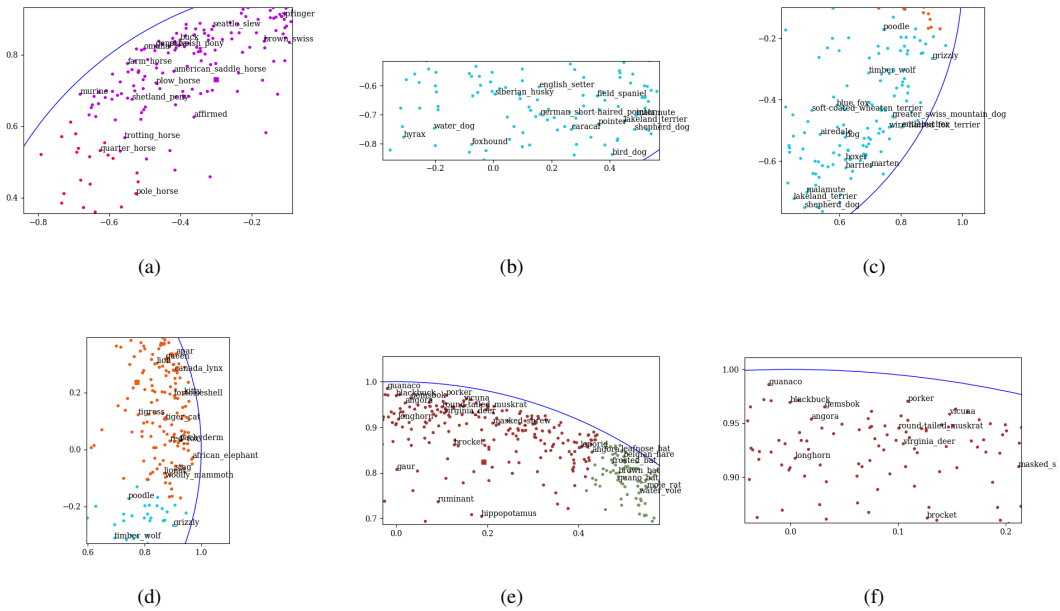


FIGURE 5 – Aperçu des étiquettes du sous-arbre des mammifères : sur les frontières de clusters distincts et à l'intérieur des clusters dans un voisinage du barycentre (ce dernier est représenté par un carré)

[20] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[21] 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.

[22] 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.

[23] 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 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 4457–4466, 2018.

[24] A. Tifrea, G. Bécigneul, and O.-E. Ganea. Poincaré glove : Hyperbolic word embeddings. *arXiv preprint arXiv :1810.06546*, 2018.

[25] C. Tu, H. Wang, X. Zeng, Z. Liu, and M. Sun. Community-enhanced network representation learning for network analysis. *CoRR*, abs/1611.06645, 2016.

[26] 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*, KDD '16, pages 1225–1234, New York, NY, USA, 2016. ACM.

[27] H. Zaatiti et al. Learning community embedding with Riemannian expectation maximisation algorithms. *Work in progress*, 2019.

[28] V. W. Zheng, S. Cavallari, H. Cai, K. C. Chang, and E. Cambria. From node embedding to community embedding. *CoRR*, abs/1610.09950, 2016.