HYDRA: A METHOD FOR STRAIN-MINIMIZING HYPERBOLIC EMBEDDING

MARTIN KELLER-RESEL AND STEPHANIE NARGANG

ABSTRACT. We introduce hydra (hyperbolic distance recovery and approximation), a new method for embedding network- or distance-based data into hyperbolic space. We show mathematically that hydra satisfies a certain optimality guarantee: It minimizes the ‘hyperbolic strain’ between original and embedded data points. Moreover, it recovers points exactly, when they are located on a hyperbolic submanifold of the feature space. Testing on real network data we show that hydra typically outperforms existing hyperbolic embedding methods in terms of embedding quality.

1. INTRODUCTION

Embeddings of networks and distance-based data into hyperbolic geometry have received substantial interest in recent years. Such embeddings have been used for visualization [Wal04], link prediction [PKS+12, PPK15] and community detection [PPK15, MTC+17]. They offer insight into the tradeoff between popularity and similarity effects in network growth [PKS+12] and have interesting implications for routing, network navigability [Kle07, BKC09] and efficient computation of shortest network paths [ZSZZ11, CK17]. Moreover, such embedding methods can be seen as alternatives to classic dimensionality reduction techniques based on Euclidean geometry, such as principal component analysis or multidimensional scaling. However, the hyperbolic embedding methods as yet proposed in the literature have either been based on specific assumptions about network growth (e.g. [PPK15, MTC+17]), or methods with strong theoretical properties, but requiring costly non-linear numerical optimization procedures (e.g. H-MDS of [Wal04], Rigel of [ZSZZ11] and HyPy of [CK17]). Here, we introduce hydra (hyperbolic distance recovery and approximation), a novel method for embedding network or distance-based data into hyperbolic space, which has strong mathematical foundations and does not depend on specific assumptions on network growth or structure. At the same time, the method is computationally efficient and based on reduced matrix Eigendecomposition. We show mathematically, that when presented with mutual distances of data points located on a low-dimensional hyperbolic submanifold of the feature space, hydra will recover these points exactly. For general data, the method satisfies a certain optimality property, similar to the strain-minimizing property of multidimensional scaling. Finally, we introduce hydra+, an extension where the result of hydra is used as inital condition for hyperbolic embedding methods based on optimization, such as Rigel/HyPy.
substantially improving their efficiency. When tested on real network data, hydra and its variants typically outperform existing hyperbolic embedding methods. All new methods introduced are available in the package hydra [KR19] for the statistical computing environment R [R C16].

2. Embeddings into Hyperbolic Space

2.1. Hyperbolic Space. We summarize the key features of the hyperboloid model of hyperbolic geometry (cf. [Rat06, CFKP97]) in dimension \( d \). This will provide the mathematical framework in which we formulate our embedding method. To start, we define for \( x, y \in \mathbb{R}^{d+1}_{1} \) the indefinite inner product

\[
\circ x \circ y := x_1 y_1 - (x_2 y_2 + \ldots + x_{d+1} y_{d+1}),
\]

also called Lorentz product. The real vector space \( \mathbb{R}^{d+1}_{1} \) equipped with this inner product is called Lorentz space and denoted by \( \mathbb{R}^{1,d}_{1} \). As nested subsets, it contains the positive Lorentz space \( \mathbb{R}^{1,d}_{1}^+ = \{ x \in \mathbb{R}^{1,d}_{1}: x_1 > 0 \} \) and the single-sheet hyperboloid

\[
H_d = \{ x \in \mathbb{R}^{1,d}_{1}: x \circ x = 1, x_1 > 0 \}.
\]

The hyperboloid model with curvature \(-\kappa\), \((\kappa > 0)\), consists of \( H_d \) endowed with the hyperbolic distance

\[
d_{\kappa}^H(x, y) = \frac{1}{\kappa} \arccosh (x \circ y), \quad x, y \in H_d.
\]

The hyperbolic distance \( d_{\kappa}^H \) is a distance on \( H_d \) in the usual mathematical sense; in particular it takes only positive values and satisfies the triangle inequality, cf. [Rat06, §3.2]. In fact, it can be shown that \( H_d \) endowed with the Riemannian metric tensor

\[
ds^2 = \frac{1}{\kappa} (dx \circ dx)
\]

is a Riemannian manifold and \( d_{\kappa}^H(x, y) \) is the corresponding Riemannian distance. The sectional curvature of this manifold is constant and equal to \(-\kappa\), which explains the role of \( \kappa \) as curvature parameter. Just as Euclidean space is the canonical model of geometry with zero curvature, hyperbolic space is the canonical model of geometry with negative curvature.

2.2. The Poincaré Ball Model. In addition to the hyperboloid model, we introduce the Poincaré ball model of hyperbolic geometry, which is more appealing for visualizations of hyperbolic space and hyperbolic embeddings. In \( \mathbb{R}^d \), consider the open unit ball

\[
B_d := \{ z \in \mathbb{R}^d : |z| < 1 \},
\]

where \(|z| = \sqrt{z_1^2 + \ldots + z_d^2}\) is the usual Euclidean norm. The hyperboloid \( H_d \) can be mapped bijectively onto \( B_d \) by the stereographic projection (cf. [Rat06 §4.2])

\[
\xi(x) = \left( \frac{x_2}{1 + x_1}, \ldots, \frac{x_{d+1}}{1 + x_1} \right).
\]

This projection transfers the hyperbolic distance from \( H_d \) to \( B_d \), by setting

\[
d_{\kappa}^H(\xi(z_1), \xi(z_2)) = d_{\kappa}^B(z_1, z_2), \quad z_1, z_2 \in B_d.
\]

Endowed with this distance, \((B_d, d_{\kappa}^B)\) is isometric to \((H_d, d_{\kappa}^H)\) and therefore an equivalent model of hyperbolic geometry.

\[1\]That is, \( d_{\kappa}^B(x, y) \) is the length of the shortest path from \( x \) to \( y \), where lengths are measured using the length element \( ds = \sqrt{ds^2} \).
It will be convenient to parameterize $B_d$ by the radial coordinate $r \in [0, 1)$ and the directional coordinate $u$ (a unit vector in $\mathbb{R}^d$), given by

$$r := \sqrt{z_1^2 + \cdots + z_d^2}, \quad u := \frac{z}{r}.$$ 

An easy calculation shows that the conversion from coordinates in $H_d$ is given by

$$r = \xi_r(x_1) := \sqrt{x_1 - 1} + 1 \quad \text{and} \quad u = \xi_u(x_2, \ldots, x_{d+1}) := \left(\frac{x_2}{\sqrt{x_2^2 + \cdots + x_{d+1}^2}}\right).$$

In dimension $d = 2$, the Poincaré ball becomes the Poincaré disc, and each of its points can be described by the radius $r$ and the unique angle $\theta \in [0, 2\pi)$ such that $z_1 = r \cos \theta, \quad z_2 = r \sin \theta$.

### 2.3. Embedding of Distances and Graphs

To formulate the embedding problem, let a symmetric matrix $D = [d_{ij}] \in \mathbb{R}^{n \times n}_{\geq 0}$ with zero diagonal be given, which represents the pairwise dissimilarities between some objects $o_1, \ldots, o_n$. The basic premise of hyperbolic embedding is that the matrix $D$ can be approximated by a hyperbolic distance matrix $H = \{d_H(x_i, x_j)\}$, i.e., that we can find points $x_1, \ldots, x_n$ in low-dimensional hyperbolic space $H_d$, such that

$$d_H^2(x_i, x_j) \approx d_{ij}.$$ 

The points $x_1, \ldots, x_n$ give a low-dimensional representation in hyperbolic space of the configuration of $o_1, \ldots, o_n$ induced by their dissimilarities. In Euclidean space, such approximations are well studied and can be calculated e.g. by multidimensional scaling (MDS), see also Section 5.2 and [BG05].

An important special case is the graph embedding problem, where a (unweighted, undirected) graph $G = (V, E)$ is given and $D = [d_{ij}]$ is the graph distance matrix of $G$, i.e., $d_{ij}$ is the length of the shortest path in $G$ from vertex $v_i$ to $v_j$. In particular for graphs with locally tree-like structure it can be expected that hyperbolic geometry gives a better representation than Euclidean geometry, see e.g. [Kle07]. Instead of the shortest-path distance, other dissimilarity measures based on the structure of $G$ can be used, such as the repulsion-attraction (RA) rule or edge-betweenness-centrality (EBC), cf. [MTC17].

### 2.4. Connection to prior work and innovations

Most existing methods for hyperbolic embedding can be placed into one of two classes: Stress-based methods or network-specific methods.

- **Stress-based methods** aim to solve the embedding problem (2.7) by minimizing the stress functional

$$\text{Stress}(x_1, \ldots, x_n)^2 := \sum_{i,j=1}^n (d_{ij} - d_H^2(x_i, x_j))^2$$

over all $x_1, \ldots, x_n \in H_d$. This minimization problem is a challenging high-dimensional non-convex optimization problem, and methods largely differ in their numerical approach to minimize (2.8). The H-MDS method proposed in [Wal04] is a gradient descent scheme for minimizing (2.8) based on explicit calculation of the gradient. [CCD17] propose a neural-network-based approach to minimizing (2.8), while [ZSZZ11] and [CK17] develop so-called ‘landmark-based’ minimization algorithms (Rigel and HyPy respectively) based on iterative quasi-Newton minimization. Due
to the ‘landmark’ heuristic, these methods are able to deal with large-scale instances of (2.8) and do not require full knowledge of $D$, see [CK17] for details.

• **Network-specific methods** focus on the graph embedding problem and rely on underlying assumptions on the generating mechanism of the graph $G$, see e.g. [PKS12] for a model of ‘hyperbolic network growth’. In HyperMap of [PPK15] and the coalescent embedding of [MTC17], the radial coordinate $r_i$ of the embedded points in the Poincaré ball model is determined directly from the degree of the vertices $v_i$, using the assumption of a power-law relationship. The directional component $u_i$ of the embedding is then determined by maximizing likelihood in an underlying probabilistic model (cf. [PPK15]) or by applying existing nonlinear dimensionality reduction methods (such as Laplacian Eigenmapping or ISOMAP) to the underlying data (cf. [MTC17]).

Here, our main innovation is to replace the stress functional (2.8) by the strain functional

$$\text{Strain}(x_1, \ldots, x_n)^2 := \sum_{i,j=1}^{n} (\cosh(\sqrt{\kappa} d_{ij}) - x_i \circ x_j)^2,$$

which results from (2.8) when all distances are transformed by hyperbolic cosine. Furthermore, we introduce a highly efficient method for the minimization of hyperbolic strain, called hydra (hyperbolic distance recovery and approximation). Contrary to stress-minimization, hydra is based on matrix Eigendecomposition, similar to principal component analysis or classic multidimensional scaling.\footnote{In fact, the relation between hyperbolic strain- and stress-minimization is similar to the relation between ‘classic’ and ‘metric’ multidimensional scaling in the Euclidean case, cf. [BG05].}

In Theorems 3.1 and 3.2 we show that hydra satisfies important theoretical optimality properties, in particular, it returns a guaranteed global minimum of (2.9). For many instances based on real data, the embedding results of hydra are better than those based on pure stress-minimization, even when embedding quality is measured by the stress functional (2.8); see Section 4 below. This shows, that even when minimization of stress is the final goal, the strain functional (2.9) is a valuable and useful proxy for stress. Indeed, the best results in terms of stress are obtained when strain- and stress-minimization are combined. This is the basis of the hydra+ method, introduced in Section 3.2, where the embedding result of hydra is used as initial condition for a stress-minimization run.

### 3. A NEW HYPERBOLIC EMBEDDING METHOD

#### 3.1. The hydra algorithm

We introduce the hydra algorithm, displayed as Algorithm 3.1 which calculates an embedding into the Poincaré ball model of hyperbolic space by efficiently solving the strain-minimization problem

$$\min_{x_i \in \mathbb{R}^{1,d}} \sum_{i,j} (\cosh(\sqrt{\kappa} d_{ij}) - (x_i \circ x_j))^2.$$

The algorithm proceeds as follows:

- In steps A1 and A2 the strain-minimization problem is solved by means of a matrix Eigendecomposition. These steps return a coordinate matrix $X = [x_{ij}]$, whose rows $x_1, \ldots, x_n$ are elements of positive Lorentz space $\mathbb{R}^{1,d}_+$ and the optimizers of (3.1). The optimality of $x_1, \ldots, x_n$ is the subject of Theorem 3.2 below.
In steps B1 and B2, the points \( x_1, \ldots, x_n \) are projected onto the Poincaré ball \( B_d \) using the stereographic projection (2.4) and converted to radial/directional coordinates \( (u_i) \) using (2.5) and (2.6). No adjustment is necessary for the directional coordinates, which are computed in step B1.

Due to (2.5), the radial coordinates \( (r_i) \) depend only on the first column \((x_{11}, \ldots, x_{n1})\) of \( X \) and can be obtained by applying \( \xi_r \) elementwise. But \( \xi_r(x_{1i}) \) may be undefined for elements with \( x_{1i} \in (0,1) \). Therefore, \( r_i \) is calculated in step B2 as

\[
r_i = \xi_r\left(\frac{\alpha x_{1i}}{x_{\min}}\right),
\]

that is, after rescaling the first column of \( X \) by \( \alpha/x_{\min} \), where \( x_{\min} > 0 \) is its smallest element and \( \alpha \geq 1 \) a user-supplied parameter defaulting to 1.

---

**Algorithm 1 hydra(D,d,κ)**

| Step A1: Set |
|-------------|
| \( A = [a_{ij}] := [\cosh(\sqrt{\kappa}d_{ij})] \) |
| and compute the Eigendecomposition |
| \( A = Q\Lambda Q^T, \) |
| where \( A \) is the diagonal matrix of the Eigenvalues \( \lambda_1 \geq \cdots \geq \lambda_n \) and the columns of \( Q \) are the Eigenvectors \( q_1, \ldots, q_n \). |

| Step A2: Allocate the \( n \times (d+1) \)-matrix |
| \( X := \begin{bmatrix} \sqrt{\lambda_1} q_1 & \sqrt{(-\lambda_{n-d+1})^+} q_{n-d+1} \cdots \sqrt{(-\lambda_n)^+} q_n \end{bmatrix}, \) |
| where \( x^+ \) denotes the positive part \( x^+ = \max(x,0) \). |

| Step B1: ‘Directional projection’ For \( i \in 1, \ldots, n \) set |
| \( u_i := \frac{(x_{i2}, \ldots, x_{i(d+1)})}{\sqrt{x_{i2}^2 + \cdots + x_{i(d+1)}^2}}, \) |
| with \( x_{ij} \) the elements of \( X \). |

| Step B2: ‘Radial projection’ For \( i \in 1, \ldots, n \) set |
| \( x_{\min} := \min(1, x_{11}, \ldots, x_{n1}) \) |
| and |
| \( r_i := \sqrt{\frac{\alpha x_{1i} - x_{\min}}{\alpha x_{1i} + x_{\min}}} \) |

| Return: Matrix \( X \) and embedding \( (r_i, u_i)_{i=1,\ldots,n} \) as radial and directional coordinates in the Poincaré ball \( B_d \). |

---

\( ^3 \)By Theorem 3.2, steps A1 and A2 guarantee that all \( x_{1i} \) are positive, but not that they are larger than one.
The key theoretical properties of the hydra algorithm are summarized in the following theorems, whose proofs are given in Section 5.1. The first theorem shows that hydra recovers any configuration of points in d-dimensional hyperbolic space up to isometry:

**Theorem 3.1 (Exact Recovery).** Let \( a_1, \ldots, a_n \) be points in hyperbolic d-space \( H_d \), and let \( D = [d_{ij}] = [d_H(a_i, a_j)] \) be the matrix of their hyperbolic distances with curvature \( -\kappa \). Then hydra\((D,d,\kappa)\) recovers the points \( a_1, \ldots, a_n \) up to isometry. In particular, the rows \( x_1, \ldots, x_n \) of the matrix \( X \) and the points \((r_1, u_1), \ldots, (r_n, u_n)\) returned by hydra\((D,d,\kappa)\) satisfy

\[
d^\kappa_B((r_i, u_i), (r_j, u_j)) = d_H(x_i, x_j) = d_{ij}, \quad i, j = 1, \ldots, n.
\]

For applications to real data, exact recovery is an atypical situation. However, hydra enjoys an optimality guarantee for strain minimization, expressed in the following theorem:

**Theorem 3.2 (Optimal Approximation).** The rows \( x_1, \ldots, x_n \) of the matrix \( X \) returned by hydra\((D,d,\kappa)\) are the globally optimal solutions of the strain minimization problem (3.1). Moreover, the first column of \( X \) is strictly positive; equivalently, all \( x_i \) are elements of positive Lorentz space \( \mathbb{R}^{1,d}_+ \).

3.2. **Practical guidelines and extensions.** While the result of hydra satisfies the theoretical optimality guarantees in Theorem 3.1 and 3.2, it can still be advantageous to make adjustments to the result. Reasons for such adjustments are: To improve the attractiveness of visualization or the embedding quality in terms of stress (2.8) (as opposed to strain, which is globally minimal). We discuss two simple, but heuristic adjustments and the hydra+ method, which is based on explicit stress-minimization:

**Alpha-adjustment:** Setting the parameter \( \alpha \) to values slightly larger than one (recommendation: \( \alpha = 1.1 \)) typically improves both visual appeal and stress value of the embedding; see also Figure 2. Increasing \( \alpha \) corresponds to vertical stretching of the hyperboloid \( H_d \) and – after projecting to \( B_d \) – to pushing points towards the boundary of the Poincaré ball.

**Equiangular adjustment:** Equiangular adjustment can be applied to two-dimensional hyperbolic embeddings and was introduced in [MTC+17]. Let \( \lambda \in [0, 1] \) be the adjustment parameter and define \( \text{ark}(\theta_i) \) as the angular rank of \( x_i \), i.e. when the embedded points are ordered by increasing angular coordinate \( \theta_i \), then \( \text{ark}(\theta_i) \) is defined as the rank (from 1 to \( n \)) of \( x_i \) in this list. The adjusted angular coordinate is then set to

\[
\theta'_i := \lambda \theta_i + (1 - \lambda)(\text{ark}(\theta_i) - 1) \frac{2\pi}{n}, \quad i = 1, \ldots, n.
\]

If \( \lambda = 0 \), no adjustment takes place. If \( \lambda = 1 \) then the angles \( \theta'_i \) are regularly spaced (‘equiangular’) and only the ordering given by \( \theta_i \) is retained. Values of \( \lambda \in (0, 1) \) interpolate between these two extremes. Values of \( \lambda \) around 1/2 typically lead to improvements in both visual appeal and stress value of the embedding; see also Figure 2.

**hydra+:** If minimization of stress is the ultimate objective and strain is used only as a proxy, the result of hydra can be used as an initial condition for a direct minimization of the stress functional (2.8). This can be seen as a chaining of hydra and HyPy/Rigel [ZSZZ11, CK17], where hydra substitutes the random initial condition of HyPy/Rigel. For the minimization of stress, efficient quasi-Newton routines, such
as LBFGS [ZBLN97] can be used and supplied with the explicit gradient of stress, given in [CK17, Eqs. (3.1),(3.2)].

In terms of efficiency, the following simple improvement can be made to hydra:

**Reduced Eigendecomposition:** The numerically dominating part of hydra is the Eigendecomposition in (3.3). Note however, that in (3.4) only the single first and the last $d$ Eigenvalues and Eigenvectors of the matrix $A$ are needed. There are efficient numerical routines (see e.g. [LSY98]) to perform such a reduced Eigendecomposition without computing the full Eigendecomposition of $A$. These routines can be used to substantially improve efficiency if $n \gg d$.

### 3.3. Remarks on strain-minimizing graph embeddings.

In the seminal paper [PKS+12] it has been argued that the inherent negative curvature in hyperbolic geometry resolves the trade-off between the conflicting attractive forces of popularity and similarity in network growth models. For this reason [PKS+12] have proposed to interpret the radial coordinate $r$ in the Poincaré disc as dimension of ‘popularity’ and the angular coordinate $\theta$ as dimension of ‘similarity’. Interestingly, the strain minimization problem (3.1) and its solution by hydra gives additional mathematical support for this interpretation. More precisely, revisiting Algorithm 3.1 in the graph embedding context, we observe that:

- **The radial coordinates** $r_i$ are determined only from the Perron-Frobenius Eigenvector $q_1$ of the matrix $A$. This provides a remarkable connection to the Eigenvector centralities (corresponding to the popularity dimension) of the nodes $v_i$, which are determined from the Perron-Frobenius Eigenvector of their adjacency matrix.
- **The directional coordinates** $u_i$ are determined only by the Eigenvectors $q_{n-d+1}, \ldots, q_n$ (and corresponding Eigenvalues) at the low end of the spectrum of $A$. This provides a remarkable connection to Cheeger’s inequality (cf. [CL06, Ch. 9]), which shows that the low end of the spectrum of the graph Laplacian matrix encodes the separability of the graph into sparsely connected ‘communities’ (corresponding to the similarity dimension).

The second point above also gives additional theoretical support for the approach of [MTC+17], whose best-performing methods use dimension-reduction techniques based on spectral decomposition, such as Laplacian Eigenmapping, for the inference of the angular coordinate in their *coalescent embeddings*. Finally, we remark that while the connections described above are a first step towards a mathematization of the popularity-similarity paradigm in hyperbolic network geometry, the matrix $A = [\cosh(\sqrt{\kappa} d_{ij})]$ is in general neither identical to the adjacency nor to the Laplacian matrix of a given graph, and thus further research into the rigorous mathematical underpinning of these connections is warranted.

### 4. Numerical Results

In our numerical experiments, we evaluate different variants of hydra and compare them to existing hyperbolic embedding methods, using stress as performance criterion. We focus on embedding quality (measured in terms of stress) and intend to study improvements of numerical efficiency and scaling to large network sizes in future research. Our evaluations are based on four small and one medium sized real-world

---

4The Perron-Frobenius Eigenvector is the Eigenvector associated to the largest Eigenvalue of a positive matrix (i.e. a matrix consisting only of positive entries) and is itself a positive vector, cf. [Hog06, Ch. 10].
| Network | Description                                                                 | Source       | # Nodes |
|---------|------------------------------------------------------------------------------|--------------|---------|
| karate  | Social interaction network (‘Zachary’s karate club network’) from [Zac77]    | igraphdata   | 34      |
| macaque | Graph model of the visuotactile brain areas and connections of the macaque monkey from [NNKB06] | igraphdata   | 45      |
| rfid    | Contacts among patients and health care workers in the geriatric unit of a hospital in Lyon, France; from [VBC +13] | igraphdata   | 75      |
| UKfaculty | Personal friendship network of a UK university faculty from [NPNB08]       | igraphdata   | 81      |
| opsalh  | One-node projection of message Exchange Network from [Ope13]; two isolated nodes have been removed | toreopsahl.com | 897     |

Table 1. Networks used for numerical experiments

networks, which are described in Table 4. Edge weights (when available) were discarded, i.e., all networks were treated as unweighted undirected graphs. This network data was used as input for the following methods:

**hydra**: The ‘pure’ hydra method (with $\alpha = 1.0$ and $\lambda = 0$) as described in Algorithm 3.1

**hydra-adj**: The hydra method with alpha-adjustment $\alpha = 1.1$ and equiangular adjustment $\lambda = 0.5$, as described in Section 3.2

**hydra+**: The hydra+ method as described in Section 3.2 and using the result of hydra-adj as initial condition.

**CE-LE**: The coalescent embedding (CE) using Laplacian Eigenmapping (LE) as dimension-reduction method, full equiangular adjustment and repulsion-attraction (RA) pre-weighting; see [MTC+17] for details. Among the methods developed in [MTC+17], this was the best performing method to invert the PSO generating mechanism of [PPK15] for hyperbolic networks.

**HyPy/Rigel**: The HyPy algorithm from [CK17], which is based on Rigel from [ZSZZ11]. Both methods are based on direct minimization of the stress functional (2.8). Landmark selection, as proposed in [CK17] was not implemented, since it serves to reduce runtime and memory use for large networks, but is not expected to improve embedding results. As in [CK17], the initial condition for minimization was chosen at random and we repeated the embedding 100 times.

For the methods hydra and hydra-adj we use our own implementations in R, which are available in the R-package hydra, [KR19]. For CE-LE we use the MATLAB implementation of the methods of [MTC+17] available from github. For HyPy/Rigel and the optimization part of hydra+ we use our own implementation in MATLAB. Here, we make use of the analytic form of the gradient of the stress functional (2.8) from [CK17]. Note that all methods except CE-LE use the shortest-path matrix as input dissimilarities; CE-LE uses repulsion-attraction (RA) weights as input dissimilarities, see [MTC+17]. For all methods hyperbolic curvature was fixed to $-\kappa = -1$ and we embed into dimension $d = 2$.

Results for all networks are shown in Figure 4. Note that stress values are normalized by dividing them by the results of hydra to facilitate comparison between

---

https://github.com/biomedical-cybernetics/coalescent_embedding
Figure 1. **Embedding performance on real network data.**

Embedding quality (measured by stress (2.8), relative to hydra) of different hyperbolic embedding methods applied to the five networks listed in Table 4. For HyPy/Rigel a 5%-95% error bar is shown, corresponding to 100 runs with randomized initial condition.

- In terms of embedding quality measured in stress, hydra outperforms CE-LE and the average result of HyPy/Rigel in all five networks considered. In three of them also the 5%-quantile of HyPy/Rigel is outperformed.
- The heuristic adjustments described in Section 3.2 yield further improvements in performance over plain hydra in all cases.
- The hydra+ method yields another substantial improvement over hydra-adj and is the best-performing method in terms of stress for all networks.

**Alpha-adjustment and equiangular adjustment.** The effects of alpha-adjustment and equiangular adjustment for different values of $\alpha$ and $\lambda$ are shown in Figure 2 for the karate network. The results of plain hydra (upper left) look somewhat crowded with several nodes located very close to each other. Both visual appeal and stress value are improved upon increasing $\alpha$ to 1.1 and $\lambda$ to 0.5 (upper right). The effect of alpha-adjustment can be described as ‘pushing towards the boundary’ of the nodes, while the effect of equiangular adjustment leads to ‘fanning out’ of the links. Increasing $\alpha$ to 1.5 and $\lambda$ to 1.0 (full equiangular adjustment) leads to a visually even more harmonious result (lower left), however, the stress value increases substantially. A contour plot of stress over $\alpha$ and $\lambda$ is shown in the lower right panel, with a global minimum at...
Figure 2. Effect of alpha- and equiangular adjustment.
The first three panels (upper row, lower left) show embeddings of the karate network using the hydra method with different alpha-adjustment $\alpha$ and equiangular adjustment $\lambda$. The two different communities of the karate network are indicated by red triangles/black circles. Filled nodes indicate the ‘community leaders’ Mr. Hi and John A., cf. [Zac77, Csa15] for details. Links are drawn as hyperbolic geodesics. The lower right panel illustrates the dependence of stress on $\alpha$ and $\lambda$ for the same network.

approx. $\alpha = 1.05$ and $\lambda = 0.5$. For other networks, results are qualitatively similar, with slightly larger optimal values of $\alpha$. This leads to our recommendation of $\alpha = 1.1$ and $\lambda = 0.5$ for a generic network embedding setting.

5. Theoretical Results

To prove the theoretical properties of the hydra method, it is convenient to re-formulate the strain minimization problem (3.1) in matrix form. To this end, let $D = [d_{ij}]$ be the given dissimilarity matrix, set $A = [\cosh(\sqrt{\kappa} d_{ij})]$ and write

$$X = (x_1, \ldots, x_n)^\top \in \mathbb{R}^{n \times (d+1)}$$

for the coordinate matrix of some points $x_1, \ldots, x_n$ in $\mathbb{R}^{d+1}$. Finally, let $J$ be the $(d + 1) \times (d + 1)$ diagonal matrix

$$J = \text{diag}(1, -1, \ldots, -1),$$

$$
\begin{align*}
\alpha = 1, \lambda = 0, \text{ stress } & = 20.83 \\
\alpha = 1.1, \lambda = 0.5, \text{ stress } & = 18.2 \\
\alpha = 1.5, \lambda = 1, \text{ stress } & = 30.72
\end{align*}
$$
The strain minimization problem can now be written in compact form as
\[
\min_{X \in \mathbb{R}^{n \times (d+1)}} \| A - X^T J X \|_F^2,
\]
where \(\| \cdot \|_F\) denotes the Frobenius norm. Imposing the constraint that all \(x_i\) are elements of the hyperboloid \(H_d\) is equivalent to requiring that
\[
\text{diag}(X^T J X) = (1, \ldots, 1) \quad \text{and} \quad X e_1 > 0,
\]
where \(e_1\) is the first standard unit vector. In particular, the first condition guarantees \(x_i \circ x_i = 1\), and the second one selects the upper sheet of the two-sheet hyperboloid thus described.

5.1. Hyperbolic strain minimization and exact recovery. For a real symmetric matrix \(A\), denote by \(n_+(A)\) and \(n_-(A)\) the number of positive and negative Eigenvalues of \(A\). The following Lemma characterizes matrices that can be written as inner product matrices (‘Gram matrices’) with respect to the Lorentz product (2.1):

**Lemma 5.1.** Let \(G = [g_{ij}] \in \mathbb{R}_{\geq 0}^{n \times n}\) be positive and symmetric, and let \(d \leq n - 1\). The following are equivalent
\begin{enumerate}[(a)]
\item \(G\) satisfies \(n_+(G) = 1\) and \(n_-(G) \leq d\).
\item \(G\) is a ‘Lorentzian Gram matrix’, i.e., there exist \(x_1, \ldots, x_n\) in \(\mathbb{R}^{1,d}\), such that
\[
g_{ij} = x_i \circ x_j, \quad \forall i, j \in 1, \ldots, n.
\]
\item There exists \(X \in \mathbb{R}^{n \times (d+1)}\), such that
\[
G = X J X^T,
\]
where \(J\) is given by (5.1).
\end{enumerate}

In addition,
\begin{itemize}
\item The first column of \(X\) is positive if and only if \(x_1, \ldots, x_n\) are in the positive Lorentz space \(\mathbb{R}_{+}^{1,d}\);
\item The points \(x_1, \ldots, x_n\) are in \(H_d\) if and only if \(\text{diag}(G) = (1, \ldots, 1)\) and the first column of \(X\) is positive.
\end{itemize}

**Proof.** The equivalence of (b) and (c) follows directly from the definition of the Lorentz product in (2.1). Next, we show that (c) implies (a): From [Lax07, Ch. 10.3] it follows from (5.3) that \(n_+(G) \leq n_+(J) = 1\) and \(n_-(G) \leq n_-(J) = d\). But \(G\) is a positive matrix and Perron’s theorem (cf. [Lax07, Ch. 16]) guarantees that its leading Eigenvalue is positive, i.e., \(n_+(G) \geq 1\), and we conclude (a). To show that (a) implies (c), assume first that \(n_-(G) = d\). By Sylvester’s law of inertia, there exists a decomposition
\[
G = \hat{X} \hat{J} \hat{X}^T, \quad \text{where} \quad \hat{J} = \text{diag}\left( +1,0,\ldots,0,-1,\ldots,-1 \right).
\]
This decomposition can be reduced to (5.3), by simply dropping all rows and columns containing only zeroes from \(J\) and by also dropping the corresponding columns from \(X\). If \(n_-(G) = d' < d\), the same procedure yields a decomposition with \(X\) of dimension \(n \times (d'+1)\) and \(J\) of dimension \((d'+1) \times (d'+1)\). Padding \(X\) with zero columns and \(J\)’s diagonal with \(-1\’s, (5.3) also follows in this case.

The additional statements follow directly from the following observations: The first column of \(X\) contains exactly the first coordinate of all points \(x_1, \ldots, x_n\). If the first
coordinate of a point \( \mathbf{x} \) is positive, it is an element of positive Lorentz space and vice versa. The diagonal of \( G \) contains the values \( x_i \circ x_i, i = 1, \ldots, n \). If \( x_i \circ x_i = 1 \) and \( x_i \in \mathbb{R}^{1,d} \) then \( x_i \) is an element of the hyperboloid \( \mathcal{H}_d \) and vice versa.

Proof of Theorem 3.2. Let \( A = [a_{ij}] = [\cosh(\sqrt{\kappa} d_{ij})] \) and let \( B = [b_{ij}] \) be another symmetric matrix in \( \mathbb{R}^{n \times n} \). Let \((\lambda_i(A))_{i=1,\ldots,n}\) and \((\lambda_i(B))_{i=1,\ldots,n}\) be their Eigenvalues in descending order, and denote by \( \| \cdot \|_F \) the Frobenius norm. By a result of Wielandt-Hoffmann, cf. [Lax07, Ch. 10, Thm. 18],

\[
\sum_{i,j} (a_{ij} - b_{ij})^2 = \| A - B \|_F^2 \geq \sum_i (\lambda_i(A) - \lambda_i(B))^2.
\]

Assume now that \( B \) is a ‘Lorentzian Gram matrix’ with elements given by \( b_{ij} = \mathbf{b}_i \circ \mathbf{b}_j, \quad i, j = 1, \ldots, n \) for some \( \mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^{1,d} \). By Lemma 5.1 this implies that \( n_+(B) = 1 \) and \( n_-(B) \leq d \). Hence all Eigenvalues of \( B \) with index \( 2, \ldots, n - d \) are zero, and we obtain

\[
\sum_{i,j} (a_{ij} - b_{ij})^2 = \| A - B \|_F^2 \geq (\lambda_1(A) - \lambda_1(B))^2 + \sum_{i=2}^{n-d} \lambda_i(A)^2 + \sum_{i=n-d+1}^n (\lambda_i(A) - \lambda_i(B))^2.
\]

For the first summand on the right hand side we have the trivial lower bound 0. In the last sum, all \( \lambda_i(B) \) are negative or zero, and hence, for any \( i = (n-d+1), \ldots, n \), we can estimate

\[
(\lambda_i(A) - \lambda_i(B))^2 \geq \begin{cases} 0 & \text{if } \lambda_i(A) \leq 0, \\ \lambda_i(A)^2 & \text{if } \lambda_i(A) > 0, \end{cases}
\]

which is the same as \( (\lambda_i(A)^+)^2 \). Together, we obtain that

\[
\sum_{i,j} (a_{ij} - b_{ij})^2 \geq \sum_{i=2}^{n-d} \lambda_i(A)^2 + \sum_{i=n-d+1}^n (\lambda_i(A)^+)^2.
\]

Denote by \( A = Q \Lambda_A Q^\top \) the Eigendecomposition of \( A \) with \( \Lambda_A = \text{diag}(\lambda_1(A), \ldots, \lambda_n(A)) \). Let \( X \) be the matrix returned by \textsc{hydra}(\( D, d, \kappa \) and \( \mathbf{x}_1, \ldots, \mathbf{x}_n \) the rows of \( X \). By Theorem 3.3 the associated Lorentzian Gram matrix \( G = X J X^\top \) has the Eigendecomposition

\[
G = Q \Lambda_G Q^\top
\]

with

\[
\Lambda_G = \text{diag}(\lambda_1(A), 0, \ldots, 0, (-\lambda_{n-d+1}(A))^+, \ldots, (-\lambda_n(A))^+).
\]

Using the unitary invariance of the Frobenius norm and the trivial identity \( x - (-x)^+ = x^+ \), we obtain

\[
\sum_{i,j} (a_{ij} - x_i \circ x_j)^2 = \| Q \Lambda_A Q^\top - Q \Lambda_G Q^\top \|_F^2 = \| \Lambda_A - \Lambda_G \|_F^2 = \sum_{i=2}^{n-d} \lambda_i(A)^2 + \sum_{i=n-d+1}^n (\lambda_i(A)^+)^2.
\]

This shows that setting \( \mathbf{b}_i := \mathbf{x}_i \) for all \( i = 1, \ldots, n \) achieves equality in (5.5) and hence that the points \( \mathbf{x}_i \) minimize (5.2). \( \square \)
Proof of Theorem 3.1. Let \( D = [d_{ij}] \) be the hyperbolic distance matrix of \( a_1, \ldots, a_n \) in \( \mathcal{H}^d \). Then \( A = [a_{ij}] = [\cosh(\sqrt{\kappa}d_{ij})] \) is the associated Lorentzian Gram matrix with elements

\[
a_{ij} = a_i \circ a_j.
\]

By Lemma 5.1, \( A \) satisfies \( n_+(A) = 1 \) and \( n_-(A) \leq d \), i.e. the Eigenvalues of \( A \) satisfy \( \lambda_i(A) = 0 \) for \( i = 2, \ldots, n - d \) and \( \lambda_i(A) \leq 0 \) for \( i = n - d + 1, \ldots, n \). Hence, it follows from (5.9) that \( \sum_{i,j} (a_{ij} - x_i \circ x_j)^2 = 0 \) or, equivalently, that

\[
x_i \circ x_j = a_i \circ a_j
\]

for all \( i, j \in 1, \ldots, n \). Applying \( \cosh(\sqrt{\kappa} \cdot \cdot \cdot) \) to both sides, we see that

\[
d_{H}(x_i, x_j) = d_{H}(a_i, a_j)
\]

and hence that \( (x_i) \) and \( (a_i) \) are isometric. \( \square \)

5.2. Comparison to classic multidimensional scaling. In several aspects, the hydra method can be seen as the ‘hyperbolic analogue’ of classic multidimensional scaling (MDS), cf. [BG05], which is based on Euclidean geometry. Below, we summarize the classical MDS method and point out parallels to (and differences from) hydra. Classical MDS also takes a matrix \( D = [d_{ij}] \in \mathbb{R}^{n \times n} \geq 0 \) with zero diagonal as input. Using the centering matrix

\[
C = I - \frac{1}{n} 1 1^\top \in \mathbb{R}^{n \times n},
\]

the ‘doubly centered’ matrix \( A = -\frac{1}{2} C^\top DC \) [compare (3.2)] is derived from \( D \), and its Eigendecomposition

\[
A = QA\Gamma^\top \quad [\text{compare (3.3)}]
\]

computed. Again, \( \Lambda \) is the diagonal matrix of the Eigenvalues \( \lambda_1 \geq \cdots \geq \lambda_n \) and the columns of \( Q \) are the Eigenvectors \( q_1, \ldots, q_n \). MDS then returns the (Euclidean) coordinate matrix

\[
X = \begin{bmatrix} \sqrt{\lambda_1} q_1 & \sqrt{\lambda_2} q_2 & \cdots & \sqrt{\lambda_d} q_d \end{bmatrix}, \quad [\text{compare (3.4)}]
\]

whose rows \( x_i \) are interpreted as points in Euclidean space \( \mathbb{R}^d \). This coordinate matrix \( X \) solves the strain minimization problem

\[
\min_{X \in \mathbb{R}^{n \times d}} \| A - X^\top X \|_F, \quad [\text{compare (5.2)}]
\]

cf. [BG05] Ch. 12. Moreover, if the input matrix \( D \) is a matrix of squared Euclidean distances, i.e., \( d_{ij} = |x_i - x_j|^2 \) then MDS recovers the points \( x_i \) exactly (up to Euclidean isometry). Note that \( X^\top X \) appearing above is the Gram matrix of the points \( x_1, \ldots, x_n \), i.e. the matrix of their scalar products \( x_i^\top x_j \), whereas the matrix \( X^\top JX \) in (5.2) is the ‘Lorentzian Gram matrix’ of the Lorentz products \( x_i \circ x_j \).

REFERENCES

[BG05] Ingwer Borg and Patrick JF Groenen. Modern multidimensional scaling: Theory and applications. Springer Science & Business Media, 2005.
[BKCO09] Marian Boguna, Dmitri Krioukov, and Kimberly C Claffy. Navigability of complex networks. Nature Physics, 5(1):74, 2009.
[CCD17] Benjamin Paul Chamberlain, James Clough, and Marc Peter Deisenroth. Neural embeddings of graphs in hyperbolic space. arXiv:1705.10359, 2017.
[CFKP97] W. James Cannon, William J. Floyd, Richard Kenyon, and Walter R. Parry. Hyperbolic geometry. In Silvio Levy, editor, Flavors of Geometry, pages 59–115. 31 edition, 1997.

\*Here, \( 1 \) denotes a matrix of ones of matching dimension.
