Knowledge and Information Systems (2023) 65:5601–5634
https://doi.org/10.1007/s10115-023-01934-2

REGULAR PAPER

Dimensionality selection for hyperbolic embeddings using decomposed normalized maximum likelihood code-length

Ryo Yuki[1] · **Yuichi Ike**[2] · **Kenji Yamanishi**[1]

Received: 4 February 2023 / Revised: 14 June 2023 / Accepted: 10 July 2023 /
Published online: 9 August 2023

Abstract
Graph embedding methods are effective techniques for representing nodes and their relations in a continuous space. Specifically, the hyperbolic space is more effective than the Euclidean space for embedding graphs with tree-like structures. Thus, it is critical how to select the best dimensionality for the hyperbolic space in which a graph is embedded. This is because we cannot distinguish nodes well with dimensionality that is considerably low, whereas the embedded relations are affected by irregularities in data with excessively high dimensionality. We consider this problem from the viewpoint of statistical model selection for latent variable models. Thereafter, we propose a novel methodology for dimensionality selection based on the minimum description length principle. We aim to introduce a latent variable modeling of hyperbolic embeddings and apply the decomposed normalized maximum likelihood code-length to latent variable model selection. We empirically demonstrated the effectiveness of our method using both synthetic and real-world datasets.

Keywords Hyperbolic graph embeddings · Minimum description length principle · Decomposed normalized maximum likelihood code-length · Statistical model selection · Latent variable models

✉ Ryo Yuki
jie-cheng-ling@g.ecc.u-tokyo.ac.jp

Yuichi Ike
ike@imi.kyushu-u.ac.jp

Kenji Yamanishi
yamanishi@g.ecc.u-tokyo.ac.jp

[1] Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, Tokyo 113-8654, Japan

[2] Institute of Mathematics for Industry, Kyushu University, 744 Motooka, Nishi-ku, Fukuoka-shi, Fukuoka 819-0395, Japan

1 Introduction

1.1 Motivation

Graphs are convenient tools for knowledge representation and can be used to represent various types of real-world data. Consequently, graph analysis has garnered significant attention in various fields, such as biology (e.g., protein–protein interaction networks) [1], social sciences (e.g., friendship networks) [2], and linguistics (e.g., word co-occurrence networks) [3], in recent years. Generally, tasks in graph analysis are classified into the following four categories: (1) node classification, (2) link prediction, (3) node clustering, and (4) graph visualization [4].

Graph embeddings, which convert discrete representations into continuous ones, such as vectors in Euclidean space, have become popular tools in graph analysis [5–8]. They provide effective solutions for the aforementioned tasks, as continuous representations can be used as the input in tasks of types (1), (2), and (3), whereas two-dimensional continuous representations are used directly in tasks of type (4).

Dimensionality is one of the most important hyperparameters in graph embeddings. First, the node classification, link prediction, and node clustering performance depend on it. Intuitively, we cannot distinguish nodes well with considerably low dimensionality, while the embedded relations are significantly affected by irregularities of data with considerably high dimensionality. Second, the training time and computational expenses directly depend on it. Therefore, the issue of dimensionality selection for graph and word embedding has garnered significant attention [9–13]. However, most of existing studies have focused on Euclidean space, although hyperbolic space is a viable alternative embedding space.

The hyperbolic space is a Riemannian manifold with negative constant curvature. In network science, a hyperbolic space is suitable for modeling hierarchical structures [14, 15]. In a tree at level h, the number of leaves and nodes is exponential in h. The analogies of the two aforementioned concepts in hyperbolic and Euclidean space are the circumference and area of a circle, respectively. In the two-dimensional hyperbolic space with constant curvature $K = -1$, the circumference of a circle is provided by $2\pi \sinh r$ and its area is $2\pi(\cosh r - 1)$ with hyperbolic radius r, both increasing exponentially with r. This analogy demonstrates that the hyperbolic space has an affinity for the hierarchical structure. However, in the two-dimensional Euclidean space, $\mathbb{R}^2$, the circumference of a circle is provided by $2\pi r$ and its area is given by πr^2, both increasing polynomially with r. Thus, increasing the dimensionality is essential for embedding a hierarchical structure in the Euclidean space. Owing to these properties, hyperbolic embeddings have been extensively studied in recent years [16–18]. However, to the best of our knowledge, there has been no previous research except [19] on dimensionality selection in the hyperbolic space.

In this study, we propose a novel methodology for dimensionality selection of hyperbolic graph embeddings. We address this issue from the viewpoint of statistical model selection. First, we demonstrate that there is a non-identifiability problem in the conventional probabilistic model of hyperbolic embeddings; that is, there is no one-to-one correspondence between the parameter and the probability distribution. This problem invalidates the use of the conventional model selection criteria, such as Akaike's information criterion (AIC) [20] and the Bayesian information criterion (BIC) [21]. To overcome this difficulty, we employ two latent variable models of hyperbolic embeddings following pseudo-uniform distributions (PUDs) [14, 15] and wrapped normal distributions (WNDs) in a hyperbolic space [22]. We

thereby introduce a criterion for dimensionality selection based on the minimum description length (MDL) principle [23].

The MDL principle asserts that the best model minimizes the total code-length required for encoding the particular data. It exhibits several advantages, such as consistency [24] and rapid convergence in the framework of probably approximately correct (PAC) learning [25]. Although the MDL-based dimensionality selection was developed for Word2Vec-type word embeddings into the Euclidean space by Hung and Yamanishi [12], their techniques cannot straightforwardly be applied to hyperbolic graph embeddings.

The DNML criterion [26] is a model selection criterion for latent variable models based on the MDL principle, where the non-identifiability problem is resolved by jointly encoding the observed and latent variables. The shorter the DNML criterion, the better the dimensionality. Herein, we propose to apply DNML into the problem of dimensionality selection for hyperbolic embeddings. The DNML criteria obtained by applying to PUD and WND are called *decomposed normalized maximum likelihood code-length for pseudo-uniform distributions* (DNML-PUD) and *DNML code-length for wrapped normal distributions* (DNML-WND), respectively.

The novelty and significance of this study are summarized as follows.

- *Proposal of a novel methodology of dimensionality selection for hyperbolic embeddings* We propose DNML-PUD and DNML-WND for selecting the best dimensionality of hyperbolic graph embeddings. We aim to introduce latent variable models of hyperbolic embeddings with PUDs and WNDs and then apply the DNML criterion to its dimensionality selection, based on the MDL principle. One of our significant contributions is to derive explicit formulas of DNML for specific cases of PUDs and WNDs.
- *Empirical demonstration of the effectiveness of our methodology* We evaluated the proposed method using both synthetic and real-world datasets. For synthetic datasets, firstly, graphs with their true dimensionality were generated. We then performed the identification of the true dimensionality. For real-world datasets, we examine a relationship between the selected dimensionality and performance of link prediction. Furthermore, we quantified to what extent the hierarchical structure of a graph was preserved using WordNet (WN) [27] dataset. Overall, our experimental results confirmed the effectiveness of our method.

The preliminary version of this paper appeared in [28]. The major updates of this paper are summarized below:

- We introduced a new latent variable model called wrapped normal distributions in hyperbolic space [22] and derived the upper bound on its DNML criterion, which we call DNML-WND.
- Besides, the evaluation of DNML-WND was added in the experimental results.
- We added the new metric called *conciseness* in the evaluation of the link prediction task in Sect. 4.3.1.

1.2 Related work

Conventionally, the dimensionality is determined heuristically based on domain knowledge. However, in recent years, several studies have proposed more principled approaches for this purpose.

Yin and Shen [9] proposed a pairwise inner product (PIP) loss, which quantifies the performance of embeddings based on the bias-variance trade-off. PIP loss is applicable to

embeddings that can be formulated as low-rank matrix approximations, and its theoretical aspects have been investigated extensively. However, it is not known if hyperbolic embeddings satisfy this condition; thus, PIP loss cannot be directly used for hyperbolic embeddings. Gu et al. [10] extended PIP loss to normalized embedding loss. It is applicable to hyperbolic embeddings after defining their normalized embedding loss of hyperbolic embeddings. However, empirical observations (for example, normalized embedding loss following Eq. (2) in [10]) are limited to the Euclidean space, and it is still unknown whether such observations are also valid or not for hyperbolic embeddings.

Luo et al. [11] proposed minimum graph entropy (MinGE) to select a dimensionality that minimizes *graph entropy*, which is a weighted sum of *feature entropy* and *structure entropy*. However, feature entropy depends on a certain probability distribution in the Euclidean space, and its extension to hyperbolic space is not straightforward. Moreover, although it was demonstrated to exhibit excellent experimental performance, there was no particular rationale with respect to the selected dimensionality. Wang [13] proposed a method that first learns embeddings in a sufficiently high-dimensional Euclidean space (e.g., the 1000-dimensional Euclidean space) and then applies principal component analysis (PCA) to the embeddings and selects the dimensionality that minimizes the predefined score function. Recently, several hyperbolic dimensionality reduction methods have also been proposed [29–31], which indicates the possibility of extending the method to the hyperbolic space. To extend the method to the hyperbolic case, the following two points should be discussed: (1) how to define the score function and (2) which dimensionality reduction method should be used.

Recently, the graph neural architecture search method (GraphNAS) has been proposed in [32]. GraphNAS selects the best architecture of graph neural networks, including their dimensionality, using reinforcement learning. The most important difference between the proposed method and GraphNAS is that GraphNAS determines the architecture in a task-dependent manner (e.g., the accuracy in node classification), while the proposed method is task-independent and estimates universal dimensionalities based on the MDL principle. Another difference is that the proposed method targets hyperbolic embeddings, while GraphNAS targets Euclidean embeddings. Thus, it is potentially possible to extend GraphNAS to hyperbolic neural networks and compare their performance with the proposed method. However, to the best of our knowledge, there are no papers that addressed this extension, and the extension is not straightforward.

Almagro and Boguna [19] proposed a dimensionality selection method for hyperbolic embeddings. In [19], dimensionality was inferred using predictive models, such as the k-nearest neighbors algorithm or deep learning, where the input is the triplet of the mean densities of chordless cycles, squares, and pentagons of a given graph.

Hung and Yamanishi [12] proposed a dimensionality selection method for Word2Vec. They applied the MDL principle to select the optimal dimensionality. However, contrary to our method, they did not employ latent variable models for embeddings and used sequentially normalized maximum code-length rather than DNML code-length.

The remainder of this paper is organized as follows. Section 2 introduces hyperbolic geometry, the non-identifiability problem, and latent variable models of hyperbolic graph embeddings. Section 3 explains the DNML criteria and algorithms used for optimization. Section 4 presents the results obtained using artificial and real-world datasets. Section 5 presents the conclusions and future work. "Appendix" section provides the derivation of the DNML code-lengths and experimental details.

2 Preliminaries

In this section, we first introduce the hyperbolic geometry following [18]. Subsequently, the non-identifiability problem of hyperbolic embeddings is discussed. Finally, we introduce two latent variable models for hyperbolic graph embeddings.

2.1 Hyperbolic geometry

2.1.1 Definition of hyperbolic space

There are several models for representing hyperbolic space[1] (e.g., the Poincaré disk model, the Beltrami–Klein model, and the Poincaré half-plane model) [33]. In this study, a hyperboloid model was used. Since all the models introduced above are isometric to each other, the discussion of the distance structure is the same for the other models. Let $\mathcal{H}^D = (\mathbb{H}^D, g_D)$ be the D-dimensional hyperbolic space, where

$$g_D = \begin{pmatrix} -1 & 0 & \cdots & 0 \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix} \in \mathbb{R}^{(D+1)\times(D+1)},$$

$\mathbb{H}^D := \{\boldsymbol{x} = (x_0, x_1, \ldots, x_D)^\top \mid \boldsymbol{x} \in \mathbb{R}^{D+1}, \langle x, x\rangle_{\mathcal{L}} = -1, x_0 > 0\}$ and $\langle \boldsymbol{u}, \boldsymbol{v}\rangle_{\mathcal{L}} = \boldsymbol{u}^\top g_D \boldsymbol{v}$. The associated distance between $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{H}^D$ is provided by $d_{\boldsymbol{uv}} = \operatorname{arcosh}\big(-\langle \boldsymbol{u}, \boldsymbol{v}\rangle_{\mathcal{L}}\big)$, where $\operatorname{arcosh}(x) := \log(x + \sqrt{x^2 - 1})$. Note that x_0 is determined by

$$x_0 = \sqrt{1 + x_1^2 + \cdots + x_D^2}. \tag{1}$$

Thus, only D variables are independent.

2.1.2 Coordinate system of hyperbolic space

Next, we explain the coordinate system of the hyperbolic space. The Cartesian coordinate system of the ambient Euclidean space was used as an element of the hyperbolic space (i.e., $\boldsymbol{x} = (x_0, x_1, \ldots, x_D)^\top \in \mathbb{H}^D$). Alternatively, for a maximum hyperbolic radius $R > 0$, the polar coordinate system $(r, \theta_1, \ldots, \theta_{D-1})^\top$ introduced in [34] was used, where $r \in [0, R]$, $\theta_1, \theta_2, \ldots, \theta_{D-2} \in [0, \pi)$, and $\theta_{D-1} \in [0, 2\pi)$. The coordinate transformation is expressed as follows:

$$\begin{cases} x_0 = \cosh r, \\ x_1 = \sinh r \cos\theta_1, \\ x_2 = \sinh r \sin\theta_1 \cos\theta_2, \\ \quad \vdots \\ x_{D-1} = \sinh r \sin\theta_1 \sin\theta_2 \ldots \sin\theta_{D-2} \cos\theta_{D-1}, \\ x_D = \sinh r \sin\theta_1 \sin\theta_2 \ldots \sin\theta_{D-2} \sin\theta_{D-1}. \end{cases} \tag{2}$$

[1] Throughout this paper, the value of the curvature K in hyperbolic space is assumed to be $K = -1$ because, as described in [15], the effect of changing the curvature in hyperbolic space can be expressed by scaling the other parameters.

In this study, we specify the coordinate system we use when we introduce the notation of elements in the hyperbolic space.

2.1.3 Tangent space and exponential map

When we introduce wrapped normal distributions and the optimization algorithm, the concepts of tangent space $\mathcal{T}_{\boldsymbol{x}}\mathbb{H}^D$ and exponential map $\mathrm{Exp}_{\boldsymbol{x}}(\cdot)$ are necessary.

For $\boldsymbol{x} \in \mathbb{H}^D$, the tangent space $\mathcal{T}_{\boldsymbol{x}}\mathbb{H}^D$ is defined as the set of vectors orthogonal to $\boldsymbol{x}$ with respect to the inner product $\langle \cdot, \cdot \rangle_{\mathcal{L}}$. Hence,

$$\mathcal{T}_{\boldsymbol{x}}\mathbb{H}^D := \{\boldsymbol{v} \mid \boldsymbol{v} \in \mathbb{R}^{D+1}, \langle \boldsymbol{x}, \boldsymbol{v} \rangle_{\mathcal{L}} = 0\}.$$

Thereafter, the exponential map $\mathrm{Exp}_{\boldsymbol{x}}(\cdot) \colon \mathcal{T}_{\boldsymbol{x}}\mathbb{H}^D \to \mathbb{H}^D$ maps a tangent vector $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathbb{H}^D$ onto $\mathbb{H}^D$ along the geodesic, where the geodesics are the generalizations of straight lines to Riemannian manifolds. The explicit forms of the exponential map and its inverse are well known (e.g., [22]) and are defined as follows:

$$\begin{aligned} \mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{v}) &:= \cosh\left(\sqrt{\langle \boldsymbol{v}, \boldsymbol{v} \rangle_{\mathcal{L}}}\right) \boldsymbol{x} + \sinh\left(\sqrt{\langle \boldsymbol{v}, \boldsymbol{v} \rangle_{\mathcal{L}}}\right) \frac{\boldsymbol{v}}{\sqrt{\langle \boldsymbol{v}, \boldsymbol{v} \rangle_{\mathcal{L}}}}, \\ \mathrm{Exp}_{\boldsymbol{x}}^{-1}(\boldsymbol{y}) &= \frac{\mathrm{arcosh}\left(-\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}}\right)}{\sqrt{\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}}^2 - 1}} \left(\boldsymbol{y} + \langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}} \boldsymbol{x}\right). \end{aligned} \tag{3}$$

2.2 Non-identifiability problem of hyperbolic embeddings

In a non-identifiable model, as pointed out in [26], the central limit theorem (CLT) does not hold for the maximum likelihood estimator uniformly over the parameter space. Thus, under these circumstances, neither AIC nor BIC can be applied to latent variable models because they are derived under the CLT assumption uniformly over the parameter space.

For notational simplicity, we omit D from the probability distribution, unless noted otherwise. We focus on undirected, unweighted, and simple graphs. Let $n \in \mathbb{Z}_{\geq 2}$ be the number of nodes. For $k \in \mathbb{Z}_{\geq 2}$, $[k]$ and Λ_k are defined as follows: $[k] := \{1, 2, \ldots, k\}$ and $\Lambda_k := \{(i, j) \mid i, j \in [k], i < j\}$. For $(i, j) \in \Lambda_n$, let $y_{ij} = y_{ji} \in \{0, 1\}$ be a random variable that assumes the value 1 if the i-th node is connected to the j-th node and 0 otherwise.

For $D = 2$ and $i \in [n]$, let $\phi_i := (r_i, \theta_i)^\top \in \mathbb{H}^D$ be the polar coordinates of the i-th node, where $r_i \in [0, R]$ and $\theta_i \in [0, 2\pi)$. In this model, $\boldsymbol{y} := \{y_{ij}\}_{(i,j)\in\Lambda_n}$ is an observable variable, whereas $\boldsymbol{\phi} := \{\phi_i\}_{i\in[n]}$ is a probability distribution parameter. For $\beta_{\max} > \beta_{\min} > 0$, $\gamma_{\max} > \gamma_{\min} > 0$, $\beta \in [\beta_{\min}, \beta_{\max}]$, and $\gamma \in [\gamma_{\min}, \gamma_{\max}]$, we assume that a random variable $\boldsymbol{y}$ is drawn from the following distribution:

$$\begin{aligned} p(\boldsymbol{y}; \boldsymbol{\phi}, \beta, \gamma) &:= \prod_{(i,j)\in\Lambda_n} p(y_{ij}; \phi_i, \phi_j, \beta, \gamma), \\ p(y_{ij}; \phi_i, \phi_j, \beta, \gamma) &:= \begin{cases} \frac{1}{1+\exp(\beta d_{\phi_i \phi_j} - \gamma)} & (y_{ij} = 1), \\ 1 - \frac{1}{1+\exp(\beta d_{\phi_i \phi_j} - \gamma)} & (y_{ij} = 0). \end{cases} \end{aligned} \tag{4}$$

Then, the following lemma holds.

Lemma 1 *We assume that $r_j \neq 0$ for some $j \in [n]$. For $\alpha \in (0, 2\pi)$, we define $\phi_i' := (r_i, \theta_i + \alpha)^\top$ for all $i \in [n]$. Then, $\boldsymbol{\phi} \neq \boldsymbol{\phi}' := \{\phi_i'\}_{i\in[n]}$, and the following equation holds:*

$$p(\boldsymbol{y}; \boldsymbol{\phi}, \beta, \gamma) = p(\boldsymbol{y}; \boldsymbol{\phi}', \beta, \gamma).$$

Therefore, the probability distribution of hyperbolic embeddings is non-identifiable.

Proof Since $\phi_j \neq \phi'_j$ holds for some $j \in [n]$ such that $r_j \neq 0$, we have $\boldsymbol{\phi} \neq \boldsymbol{\phi}'$. For all $(i, j) \in \Lambda_n$, $d_{\phi_i \phi_j} = d_{\phi'_i \phi'_j}$ because a simple calculation yields $\langle \phi_i, \phi_j \rangle_{\mathcal{L}} := \langle \phi'_i, \phi'_j \rangle_{\mathcal{L}}$. Thus, the result follows from Eq. (4). □

For $D \geq 3$, non-identifiability can be proved by a similar transformation to the $(D-1)$-th angular coordinate.

2.3 Latent variable models of hyperbolic embeddings with PUDs and WNDs

To resolve the non-identifiability problem, we introduce two latent variable models, following work on PUDs [14, 15, 34] and WNDs [22]. In PUDs and WNDs, an embedding is regarded as a set of latent variables and edges as observed variables. Among several latent variable models, we chose two for the following reasons.

- For PUDs, in [14, 15, 34], it has been demonstrated that the graphs generated with PUDs have two properties: the power law for the degree of a node and high clustering coefficient. These properties are common in real-world graphs [35].
- For WNDs, it has been demonstrated that the experimental performance of various downstreaming tasks has been improved.

In these models, we consider $\boldsymbol{\phi} = \{\phi_i\}_{i \in [n]}$ as latent random variables rather than parameters. Thus, we rewrite it as $\boldsymbol{z} := \{z_i\}_{i \in [n]}$, where $z_i := (z_{i,0}, z_{i,1}, \ldots, z_{i,D})^\top$ denotes its Cartesian coordinates.

2.3.1 Latent variable model with PUDs

The generation process of $\boldsymbol{y}, \boldsymbol{z}$ with PUDs can be summarized as follows:

1. For vertex $i \in [n]$:
   (a) Generate $u_i := (r_i, \theta_{i,1}, \ldots, \theta_{i,D-1})^\top \sim p(u_i; \sigma, R)$.
   (b) Transform u_i to z_i through Eq. (2).
2. For vertices $(i, j) \in \Lambda_n$:
   (a) Generate an observable variable $y_{ij} \sim p(y_{ij} \mid z_i, z_j; \beta, \gamma)$ using Eq. (4).

Below we provide an explicit form of the probability distribution of $\boldsymbol{z}$ for PUDs.

For $\sigma_{\max} > \sigma_{\min} \geq 0$, the random variable $\boldsymbol{u} := \{u_i\}_{i \in [n]}$ is drawn according to the following distribution with the parameter $\sigma \in [\sigma_{\min}, \sigma_{\max}]$:

$$
\begin{aligned}
p(\boldsymbol{u}; \sigma, R) &:= \prod_{i \in [n]} p(u_i; \sigma, R), \\
p(u_i; \sigma, R) &:= p(r_i; \sigma, R) \prod_{j=1}^{D-1} p(\theta_{i,j}), \\
p(\theta_{i,j}) &:= \begin{cases} \frac{\sin^{D-1-j} \theta_{i,j}}{I_{D,j}} & (j \neq D-1), \\ \frac{1}{2\pi} & (j = D-1), \end{cases} \\
p(r_i; \sigma, R) &:= \frac{\sinh^{D-1}(\sigma r_i)}{C_D(\sigma)},
\end{aligned}
$$

where $I_{D,j} := \int_0^\pi \sin^{D-1-j}\theta \mathrm{d}\theta$ and $C_D(\sigma) := \int_0^R \sinh^{D-1}(\sigma r)\mathrm{d}r$ denote the normalization constants. For $p(\boldsymbol{z}; \sigma, R)$, because $z_{i,0}$ is determined by the other D variables in Eq. (1), we have

$$p(\boldsymbol{z}; \sigma, R) := \prod_{i\in[n]} p(z_i; \sigma, R),$$
$$p(z_i; \sigma, R) := \frac{1}{J(z_{i,1:D} : u_i)} p(u_i, \sigma, R),$$

where $z_{i,1:D} := (z_{i,1}, \ldots, z_{i,D})^\top$ and $J(z_{i,1:D} : u_i)$ is the Jacobian of the transformation from u_i to $z_{i,1:D}$, which is given as

$$J(z_{i,1:D} : u_i) = \cosh(r_i)\sinh^{D-1}(r_i) \prod_{j=1}^{D-2} \sin^{D-j-1}\theta_{i,j}.$$

The derivation is provided in "Appendix A." The probability distribution $p(\boldsymbol{z}; \sigma, R)$ is called the *pseudo-uniform distribution* because it is reduced to the uniform distribution in hyperbolic space when $\sigma = 1$.

In the following discussion, the value of R is assumed to be constant and satisfies $R = O(\log n)$ where n is the number of nodes, and it is omitted from the description of the probability distribution. This is because the maximum average degree satisfies $k_{\max} = O(n)$, and the minimum average degree satisfies $k_{\min} = O(1)$ under certain conditions, which is a common property of real-world complex networks [34].

In the aforementioned distribution, σ, β, and γ are parameters, and D denotes the model of the probability distribution.

2.3.2 Probability distribution of WNDs

WNDs are a generalization of Euclidean Gaussian distributions to the hyperbolic space. Thus, WNDs have two parameters: a mean in hyperbolic space $\boldsymbol{\mu} \in \mathbb{H}^D$ and a positive-definite covariance matrix $\Sigma \in \mathbb{R}^{D\times D}$. In our model, we set $\boldsymbol{\mu}$ to $\boldsymbol{\mu}_0$, where $\boldsymbol{\mu}_0 := (1, 0, \ldots, 0)^\top$ denote the *origin* of $\mathbb{H}^D$. We assume this because a tree-like graph is considered to be radially distributed around the origin $\boldsymbol{\mu}_0$.

The generation process of $\boldsymbol{y}$ and $\boldsymbol{z}$ with the WNDs is summarized as follows:

1. For a vertex $i \in [n]$:
   (a) Generate $v_i := (v_{i,1}, \ldots, v_{i,D})^\top \sim p(v_i; \Sigma)$.
   (b) Transform v_i to $\tilde{v}_i := [0, v_i^\top]^\top$, which is a tangent vector at $\boldsymbol{\mu}_0$.
   (c) Transform $\tilde{v}_i$ to z_i through the exponential map $\mathrm{Exp}_{\boldsymbol{\mu}_0}(\tilde{v}_i)$.
2. For a pair of vertices $(i, j) \in \Lambda_n$:
   (a) Generate an observable variable $y_{ij} \sim p(y_{ij} \mid z_i, z_j; \beta, \gamma)$ using Eq. (4).

Note that the second step is the same as that of the model with the PUDs. Below we provide an explicit form of the probability distribution of $\boldsymbol{z}$ with the WNDs.

A random variable $\boldsymbol{v} := \{v_i\}_{i\in[n]}$ is drawn according to the following distribution:

$$p(\boldsymbol{v}; \Sigma) := \prod_{i\in[n]} p(v_i; \Sigma),$$

$$p(v_i;\Sigma) := \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}v_i^\top\Sigma^{-1}v_i\right).$$

For $p(\boldsymbol{z};\Sigma)$, we have that

$$p(\boldsymbol{z};\Sigma) := \prod_{i\in[n]} p(z_i;\Sigma),$$
$$p(z_i;\Sigma) := \frac{1}{J(z_{i,1:D} : v_i)}p(v_i,\Sigma),$$

where $J(z_{i,1:D} : v_i)$ is the Jacobian of the transformation from v_i to $z_{i,1:D}$, which is provided by

$$J(z_{i,1:D} : v_i) = \left\{\frac{\sinh\|v_i\|_{\mathcal{L}}}{\|v_i\|_{\mathcal{L}}}\right\}^{D-1}.$$

The derivation of the Jacobian is obtained from [22].

3 Dimensionality selection using DNML code-lengths

In this section, we present the calculation of the DNML code-lengths for two latent variable models. Thereafter, we present the optimization algorithm.

3.1 DNML code-lengths with PUDs and WNDs

According to the MDL principle [24], the probabilistic model that minimizes the total code-length required to encode the given data is selected. Data may be encoded using multiple methods. Although the NML code-length [36] is one of the most common encoding methods, its calculation is quite difficult for complex probability distributions such as PUDs and WNDs. Therefore, we employ the DNML code-length [26], whose calculation for latent variable models is relatively easier.

Let $\mathcal{D} := \{D_1, D_2, \ldots D_N\}$ denote a finite set of candidates of dimensionalities ($D_i \in \mathbb{Z}_{\geq 2}$). We estimate optimal dimensionality $\hat{D} \in \mathcal{D}$ and the optimal embedding $\hat{\boldsymbol{z}}$ that minimizes the following criterion, which we call DNML-PUD:

$$\begin{aligned}
L_{\text{DNML-PUD}}(\boldsymbol{y},\boldsymbol{z}) &:= L_{\text{NML}}(\boldsymbol{y}\mid\boldsymbol{z}) + L_{\text{NML}}(\boldsymbol{z}),\\
L_{\text{NML}}(\boldsymbol{y}\mid\boldsymbol{z}) &:= -\log p(\boldsymbol{y}\mid\boldsymbol{z};\hat{\beta}(\boldsymbol{y},\boldsymbol{z}),\hat{\gamma}(\boldsymbol{y},\boldsymbol{z}))\\
&\quad + \log\sum_{\bar{\boldsymbol{y}}} p(\bar{\boldsymbol{y}}\mid\boldsymbol{z};\hat{\beta}(\bar{\boldsymbol{y}},\boldsymbol{z}),\hat{\gamma}(\bar{\boldsymbol{y}},\boldsymbol{z})),\\
L_{\text{NML}}(\boldsymbol{z}) &:= -\log p(\boldsymbol{z};\hat{\sigma}(\boldsymbol{z})) + \log\int p(\bar{\boldsymbol{z}};\hat{\sigma}(\bar{\boldsymbol{z}}))\mathrm{d}\bar{\boldsymbol{z}}_{1:D},
\end{aligned} \tag{5}$$

where $\hat{\beta}(\cdot,\cdot)$, $\hat{\gamma}(\cdot,\cdot)$, and $\hat{\sigma}(\cdot)$ denote the maximum likelihood estimators, $\bar{\boldsymbol{z}}_{1:D} := \{\bar{z}_{i,1:D}\}_{i\in[n]}$, and the sum and integral are obtained over all possible data in the predefined data domain. The second term in each NML code-length is called *parametric complexity*. As the exact value of the integral is analytically intractable, we approximate $L_{\text{NML}}(\boldsymbol{y}\mid\boldsymbol{z})$ and $L_{\text{NML}}(\boldsymbol{z})$ as follows:

$$L_{\text{NML}}(\boldsymbol{y}\mid\boldsymbol{z}) \approx -\log p(\boldsymbol{y}\mid\boldsymbol{z};\hat{\beta}(\boldsymbol{y},\boldsymbol{z}),\hat{\gamma}(\boldsymbol{y},\boldsymbol{z}))$$

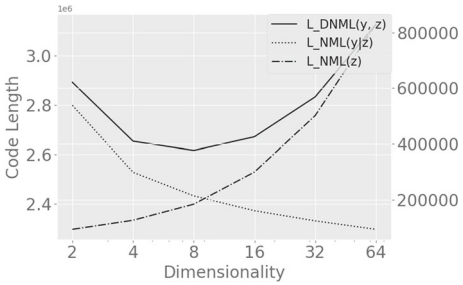

Fig. 1 Example of DNML-PUD. The selected dimensionality and true dimensionality are $D = 8$. The graph was generated with parameters $n = 6400$, $\beta = 0.6$, $\gamma = \beta \log n$, and $\sigma = 1.0$. The scores $L_{\mathrm{DNML}}(\boldsymbol{y}, \boldsymbol{z})$ and $L_{\mathrm{NML}}(\boldsymbol{y} \mid \boldsymbol{z})$ follow the left scale, whereas $L_{\mathrm{NML}}(\boldsymbol{z})$ follows the right scale

$$+ \log \frac{n(n-1)}{4\pi} + \log \int_{\gamma_{\min}}^{\gamma_{\max}} \int_{\beta_{\min}}^{\beta_{\max}} \sqrt{|I_n(\beta, \gamma)|} \mathrm{d}\beta \mathrm{d}\gamma,$$

$$L_{\mathrm{NML}}(\boldsymbol{z}) \approx - \log p(\boldsymbol{z}; \hat{\sigma}(\boldsymbol{z})) + \frac{1}{2} \log \frac{n}{2\pi} + \log \int_{\sigma_{\min}}^{\sigma_{\max}} \sqrt{|I(\sigma)|} \mathrm{d}\sigma,$$

where $I_n(\beta, \gamma)$ and $I(\sigma)$ denote Fisher information, which is computed as

$$I_n(\beta, \gamma) = \frac{2}{n(n-1)} \begin{pmatrix} \sum_{(i,j) \in \Lambda_n} \frac{d_{z_i z_j}^2}{4 \cosh^2\left(\frac{\beta d_{z_i z_j} - \gamma}{2}\right)} & \sum_{(i,j) \in \Lambda_n} \frac{-d_{z_i z_j}}{4 \cosh^2\left(\frac{\beta d_{z_i z_j} - \gamma}{2}\right)} \\ \sum_{(i,j) \in \Lambda_n} \frac{-d_{z_i z_j}}{4 \cosh^2\left(\frac{\beta d_{z_i z_j} - \gamma}{2}\right)} & \sum_{(i,j) \in \Lambda_n} \frac{1}{4 \cosh^2\left(\frac{\beta d_{z_i z_j} - \gamma}{2}\right)} \end{pmatrix},$$

$$I(\sigma) = (D-1)^2 \frac{\int_0^R r^2 \cosh^2(\sigma r) \sinh^{D-3}(\sigma r) \mathrm{d}r}{C_D(\sigma)} - \left\{ \frac{\int_0^R (D-1) r \cosh(\sigma r) \sinh^{D-2}(\sigma r) \mathrm{d}r}{C_D(\sigma)} \right\}^2.$$

The derivation is presented in "Appendix B." Practically, $I_n(\beta, \gamma)$ and $I(\sigma)$ are calculated numerically because the analytic solution of the integral terms is not trivial.

For WNDs, the DNML criterion is defined as follows:

$$L_{\text{DNML-WND}}(\boldsymbol{y}, \boldsymbol{z}) := L_{\mathrm{NML}}(\boldsymbol{y} \mid \boldsymbol{z}) + L_{\mathrm{NML}}(\boldsymbol{z}),$$

$$L_{\mathrm{NML}}(\boldsymbol{z}) := - \log p(\boldsymbol{z}; \hat{\Sigma}(\boldsymbol{z})) + \log \int p(\bar{\boldsymbol{z}}; \hat{\Sigma}(\bar{\boldsymbol{z}})) \mathrm{d}\bar{\boldsymbol{z}}_{1:D}, \tag{6}$$

where $\hat{\Sigma}$ denotes the maximum likelihood estimator of Σ. Since the exact value of $\int p(\bar{\boldsymbol{z}}; \hat{\Sigma}(\bar{\boldsymbol{z}})) \mathrm{d}\bar{\boldsymbol{z}}_{1:D}$ is analytically intractable, we employ the following upper bound.

$$\int p(\bar{\boldsymbol{z}}; \hat{\Sigma}(\bar{\boldsymbol{z}})) \mathrm{d}\bar{\boldsymbol{z}}_{1:D} \leq \frac{\pi^{\frac{D^2}{2}} \prod_{i \in [D-1]} \epsilon_{2i}^{D-i} \prod_{j \in [D]} \epsilon_{1j}^{\frac{1-D}{2}}}{\Gamma_D\left(\frac{D}{2}\right) \Gamma_D\left(\frac{n}{2}\right)} \left(\frac{n}{2e}\right)^{\frac{nD}{2}} \left(\frac{2}{D-1}\right)^D.$$

The derivation is presented in "Appendix B."

We provide a more detailed explanation of DNML-PUD and DNML-WND. Figures 1 and 2 show $L_{\mathrm{NML}}(\boldsymbol{y} \mid \boldsymbol{z})$, $L_{\mathrm{NML}}(\boldsymbol{z})$ and $L_{\mathrm{DNML}}(\boldsymbol{y}, \boldsymbol{z})$ of an artificially generated graph with the true dimensionality $D_{\mathrm{true}} = 8$ and $n = 6400$. The value of $L_{\mathrm{NML}}(\boldsymbol{y} \mid \boldsymbol{z})$ decreases as dimensionality increases. This implies that as the dimensionality increases, the graph can be reconstructed more accurately. However, the value of $L_{\mathrm{NML}}(\boldsymbol{z})$ increases as dimensionality increases. This is because more code-length is required to encode the extra dimension of the embedding; that is, the model becomes more complex, and $L_{\mathrm{NML}}(\boldsymbol{z})$ acts as a penalty

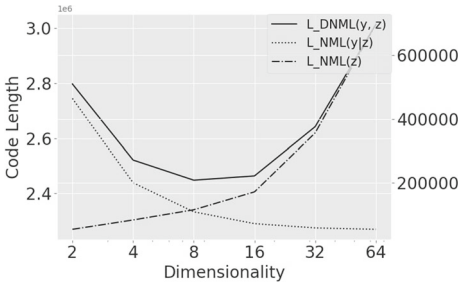

Fig. 2 Example of DNML-WND. The selected dimensionality and true dimensionality are $D = 8$. The graph was generated with parameters $n = 6400$, $\beta = 1.2$, $\gamma = \beta \log n$, and $\Sigma = (0.35 \log n)^2 I$, where $I \in \mathbb{R}^{D \times D}$ is the identity matrix. The scores $L_{\mathrm{DNML}}(\boldsymbol{y}, \boldsymbol{z})$ and $L_{\mathrm{NML}}(\boldsymbol{y} \mid \boldsymbol{z})$ follow the left scale, whereas $L_{\mathrm{NML}}(\boldsymbol{z})$ follows the right scale

term. Hence, minimizing $L_{\mathrm{DNML}}(\boldsymbol{y}, \boldsymbol{z})$ implies that dimensionality is chosen by considering both the accuracy of the reconstruction and the complexity of the model. Therefore, the DNML-PUD and DNML-WND select the true dimensionality $D_{\mathrm{true}} = 8$.

3.2 Optimization

To derive the optimal dimensionality, Eqs. (5) and (6) should be optimized with respect to $\boldsymbol{z}, \beta, \gamma$ and σ for each dimensionality $D \in \mathcal{D}$. However, direct optimization is difficult because of the analytical intractability of the parametric complexity of $L_{\mathrm{NML}}(\boldsymbol{y} \mid \boldsymbol{z})$. Instead, we optimize $L(\boldsymbol{z}, \beta, \gamma, \sigma) := -\log p(\boldsymbol{y}, \boldsymbol{z}; \beta, \gamma, \sigma)$ and $L(\boldsymbol{z}, \beta, \gamma, \Sigma) := -\log p(\boldsymbol{y}, \boldsymbol{z}; \beta, \gamma, \Sigma)$, which are lower bounds of Eqs. (5) and (6), respectively.

First, we explain how to optimize $L(\boldsymbol{z}, \beta, \gamma, \sigma)$. We rewrite it as

$$\begin{aligned} L(\boldsymbol{z}, \beta, \gamma, \sigma) &= -\sum_{(i,j) \in \Lambda_n} \log p(y_{ij} \mid z_i, z_j; \beta, \gamma) - \sum_{i \in [n]} \log p(z_i; \sigma) \\ &= \sum_{(i,j) \in \Lambda_n} \left\{ -\log p(y_{ij} \mid z_i, z_j; \beta, \gamma) - \frac{1}{n-1} \log p(z_i; \sigma) \right. \\ &\quad \left. - \frac{1}{n-1} \log p(z_j; \sigma) \right\}. \end{aligned}$$

We applied the stochastic update rule at iteration t using the following equation:

$$\begin{aligned} L^{(t)}(\boldsymbol{z}, \beta, \gamma, \sigma) := \frac{1}{|\mathcal{B}^{(t)}|} \sum_{(i,j) \in \mathcal{B}^{(t)}} \left\{ -\log p(y_{ij} \mid z_i, z_j; \beta, \gamma) \right. \\ \left. - \frac{1}{n-1} \log p(z_i; \sigma) - \frac{1}{n-1} \log p(z_j; \sigma) \right\}, \end{aligned}$$

where $\mathcal{B}^{(t)} \subset \Lambda_n$ is the mini-batch for each iteration and $|\cdot|$ denotes the number of elements in a set.

For z_i, we used the geodesic update in the hyperboloid model [18]. The update rule for $\boldsymbol{z}$ is given as follows:

$$z_i^{(t+1)} \leftarrow \mathrm{proj}_{\mathbb{H}_R^D} \left(\mathrm{Exp}_{z_i^{(t)}} \left(-\eta_z^{(t)} \pi_{z_i^{(t)}} \left(g_D^{-1} \frac{\partial L^{(t)}}{\partial z_i^{(t)}} \right) \right) \right), \tag{7}$$

where $\eta_z^{(t)}$ denotes the learning rate, $\pi_z(\cdot)$ denotes the projection from the Euclidean gradient to the Riemannian gradient, $\mathrm{proj}_{\mathbb{H}_R^D}(\cdot)$ denotes the projection from $\mathbb{H}^D$ to $\mathbb{H}_R^D := \{\boldsymbol{x} \mid \boldsymbol{x} \in \mathbb{H}^D, d_{\boldsymbol{\mu}_0 \boldsymbol{x}} \leq R\}$, and $\mathrm{Exp}_z(\cdot)$ is given by Eq. (3). The functions $\pi_z(\cdot)$ and $\mathrm{proj}_{\mathbb{H}_R^D}(\cdot)$ are defined as follows:

$$\pi_z(\boldsymbol{u}) := \boldsymbol{u} - \frac{\langle z, \boldsymbol{u} \rangle_{\mathcal{L}}}{\langle z, z \rangle_{\mathcal{L}}} z = \boldsymbol{u} + \langle z, \boldsymbol{u} \rangle_{\mathcal{L}} z,$$

$$\mathrm{proj}_{\mathbb{H}_R^D}(z) := \begin{cases} z & (d_{\boldsymbol{\mu}_0 z} \leq R), \\ \left(\cosh R, \frac{\sinh R}{\|z_{1:D}\|} z_1, \cdots, \frac{\sinh R}{\|z_{1:D}\|} z_D\right)^\top & (d_{\boldsymbol{\mu}_0 z} > R), \end{cases}$$

where $\|\cdot\|$ denotes the Euclidean norm.

With the learning rates $\eta_\beta^{(t)}$ and $\eta_\sigma^{(t)}$, the update rules of β and γ are provided as

$$\beta^{(t+1)} \leftarrow \mathrm{proj}_{[\beta_{\min}, \beta_{\max}]}\left(\beta^{(t)} - \eta_\beta^{(t)} \frac{\partial L^{(t)}}{\partial \beta^{(t)}}\right), \tag{8}$$

$$\gamma^{(t+1)} \leftarrow \mathrm{proj}_{[\gamma_{\min}, \gamma_{\max}]}\left(\gamma^{(t)} - \eta_\gamma^{(t)} \frac{\partial L^{(t)}}{\partial \gamma^{(t)}}\right), \tag{9}$$

$$\mathrm{proj}_{[a,b]}(x) = \begin{cases} b & (x \geq b), \\ x & (a \leq x \leq b), \\ a & (x < a). \end{cases}$$

Through a preliminary experiment using synthetic datasets, we confirmed that σ rarely converges to the true value when using the gradient descent method. Thus, for each epoch, we numerically calculated $\hat{\sigma}(z)$ as

$$\hat{\sigma}(z) = \arg\min_{\sigma \in S} \{-\log p(z; \sigma)\}, \tag{10}$$

where $S = \{\sigma_{\min}, \sigma_{\min} + \frac{1}{C}(\sigma_{\max} - \sigma_{\min}), \ldots, \sigma_{\min} + \frac{C-1}{C}(\sigma_{\max} - \sigma_{\min}), \sigma_{\max}\}$, and $C + 1$ denote the number of candidates.

For the optimization of $L(z, \beta, \gamma, \Sigma)$, we define $L^{(t)}(z, \beta, \gamma, \Sigma)$ in a similar manner. The update rules for z_i, β, γ are provided by Eqs. (7), (8), and (9), respectively. For each epoch, we optimized Σ using the following equation:

$$\begin{aligned} \hat{\Sigma}(z) &= \arg\min_{\Sigma} \{-\log p(z; \Sigma)\} \\ &= \frac{1}{n} \sum_{i \in [n]} v_i v_i^\top, \end{aligned} \tag{11}$$

where $v_i := \mathrm{Exp}_{\boldsymbol{\mu}_0}^{-1}(z_i)$ for all $i \in [n]$. The optimization procedure is summarized in Algorithm 1.

Subsequently, we analyze the time efficiency of the proposed algorithms. For sufficiently large n, the update of $z^{(t)} := \{z_i^{(t)}\}_{i \in [n]}$ is dominant. We assume that $|\mathcal{B}^{(t)}|$ takes a constant value B for all iteration. Since at most $2B$ nodes are used in each iteration to compute $L^{(t)}$, we only need to update $O(B)$ nodes. Thus, $z^{(t)}$ will be updated $O(ETB)$ times in total.

Algorithm 1 Riemannian Gradient Descent for DNML criteria

Given: n: number of nodes, D: dimensionality, $\beta_{\min}$ and $\beta_{\max}$: hyperparameters for β, $\gamma_{\min}$ and $\gamma_{\max}$: hyperparameters for γ, $\sigma_{\min}$ and $\sigma_{\max}$: hyperparameters for σ, R: maximum hyperbolic radius, $\eta_z^{(t)}, \eta_\beta^{(t)}, \eta_\gamma^{(t)} > 0$: learning rates, C: number of candidate points for σ, E: number of epochs, T: number of iterations.
Initialize $\{z_i^{(t)}\}_{i\in[n]}$, $\beta^{(t)}$, and $\gamma^{(t)}$.
Initialize $\sigma^{(t)}$ if we calculate DNML-PUD; initialize $\Sigma^{(t)}$ if DNML-WND.
for $e = 1$ to E **do**
  for $t = 1$ to T **do**
    for $i = 1$ to n **do**
      if $z_i^{(t)}$ is used for calculating $L^{(t)}(z, \beta, \gamma, \sigma)$ or $L^{(t)}(z, \beta, \gamma, \Sigma)$ **then**
        Update $z_i^{(t)}$ using Eq. (7).
      end if
    end for
    Update $\beta^{(t)}$ using Eq. (8).
    Update $\gamma^{(t)}$ using Eq. (9).
  end for
  Update $\sigma^{(t)}$ using Eq.(10) if we calculate DNML-PUD, and update $\Sigma^{(t)}$ using Eq.(11) if DNML-WND.
end for
return $\{z_i^{(T)}\}_{i\in[n]}, \beta^{(T)}, \gamma^{(T)}, \sigma^{(T)}, \Sigma^{(t)}$.

4 Experimental results

This section presents a comparison between the proposed criteria and conventional methods using artificial and real-world datasets. The details of code, data, and training details are presented in "Appendix C."

4.1 Methods for comparison

We used three criteria—AIC, BIC, and MinGE—for a comparative analysis of the performance of the proposed method. Here, the AIC and BIC with respect to the non-identifiable model, that is, β, γ, and z, are interpreted as parameters and are defined as follows:

$$\mathrm{AIC}(\boldsymbol{y}; D) := -\log p(\boldsymbol{y} \mid z; \hat{\beta}(\boldsymbol{y}, z), \hat{\gamma}(\boldsymbol{y}, z)) + (nD + 2),$$
$$\mathrm{BIC}(\boldsymbol{y}; D) := -\log p(\boldsymbol{y} \mid z; \hat{\beta}(\boldsymbol{y}, z), \hat{\gamma}(\boldsymbol{y}, z)) + \frac{nD+2}{2} \log \frac{n(n-1)}{2}.$$

Note that these criteria are not guaranteed to work for this model because of the non-identifiability. MinGE [11] is a dimensionality selection criterion for Euclidean graph embeddings. We set the weighting factor $\lambda = 1$ and selected the dimensionality, where MinGE was closest to 0.

Furthermore, we did not consider the cross-validation (CV) for comparison. This is because CV requires considerable computation time, particularly, when learning graph embeddings.

It should be noted that we optimized $-\log p(\boldsymbol{y}, z; \beta, \gamma, \sigma)$ for DNML-PUD, $-\log p(\boldsymbol{y}, z; \beta, \gamma, \Sigma)$ for DNML-WND, and $-\log p(\boldsymbol{y} \mid z; \beta, \gamma)$ for AIC and BIC. Thus, three embeddings were generated for each graph.

Table 1 Average MAPs on PUD-8. The bold indicates either the maximum MAP or MAP within a 10% decrease from the maximum one (average estimated dimensionality in the parentheses)

# of nodes	DNML-PUD	DNML-WND	AIC	BIC	MinGE
800	0.500 (4.0)	0.500 (4.0)	**0.666** (5.3)	0.333 (4.0)	0.250 (64.0)
1600	0.500 (4.0)	0.500 (4.0)	**1.000** (8.0)	0.486 (4.0)	0.250 (64.0)
3200	0.500 (4.0)	**1.000** (8.0)	**1.000** (8.0)	0.500 (4.0)	0.250 (64.0)
6400	**0.958** (8.7)	0.569 (14.7)	0.486 (16.0)	0.500 (4.0)	0.250 (64.0)

Table 2 Average MAPs on PUD-16. The bold indicates either the maximum MAP or MAP within a 10% decrease from the maximum one (average estimated dimensionality in the parentheses)

# of nodes	DNML-PUD	DNML-WND	AIC	BIC	MinGE
800	0.250 (4.0)	0.271 (4.0)	**0.333** (4.7)	0.250 (4.0)	**0.333 (64.0)**
1600	**0.326 (4.0)**	**0.333** (4.0)	**0.333** (8.0)	0.250 (4.0)	**0.333** (64.0)
3200	0.333 (4.0)	0.333 (8.0)	**0.500** (8.0)	0.250 (4.0)	0.333 (64.0)
6400	0.541 (10.0)	**1.000** (16.0)	**1.000** (16.0)	0.250 (4.0)	0.333 (64.0)

4.2 Artificial dataset

In this experiment, we verified whether the proposed DNML criteria could estimate the true dimensionality.

4.2.1 Dataset detail

We considered the case of $D_{\text{true}} = 8$, where D_{true} is the true dimensionality of the PUDs. We generated a graph for each combination of parameters from the following candidates: $n \in \{800, 1600, 3200, 6400\}$, $\beta \in \{0.5, 0.6, 0.7, 0.8\}$, and $\sigma \in \{0.5, 1.0, 2.0\}$. Furthermore, we set $R = \log n$ and $\gamma = \beta \log n$. Consequently, we obtained 48 graphs in total, which we call PUD-8. Similarly, we generated PUD-16 with the true dimensionality which was 16. The other parameters are the same as those of PUD-8.

Subsequently, we made WND-8. For the true dimensionality $D_{\text{true}} = 8$, we generated a graph for each combination of parameters from the following candidates: $n \in \{800, 1600, 3200, 6400\}$, $\beta \in \{0.5, 0.6, 0.7, 0.8\}$, and $\Sigma \in \{(0.35 \log n)^2 I, (0.375 \log n)^2 I, (0.40 \log n)^2 I\}$, where $I \in \mathbb{R}^{D \times D}$ is the identity matrix. Furthermore, we also set $R = \log n$ and $\gamma = \beta \log n$. Similarly, we generated WND-16 with $\Sigma \in \{(0.225 \log n)^2 I, (0.25 \log n)^2 I, (0.275 \log n)^2 I\}$. The other parameters are the same as those of WND-8. The set of candidates for the dimensionalities was $\mathcal{D} = \{2, 4, 8, 16, 32, 64\}$.

In the above generation process, the parameters were set such that the generated graphs were sparse; that is, the average degree is low with respect to the number of nodes.

4.2.2 Results

To provide an illustrative example for each criterion, we first compared the selected dimensionality of PUD-8 and WND-8 with $n = 800, 6400$. Figures 3 and 4 show the normalized values for each criterion.

Table 3 Average MAPs on WND-8. The bold indicates either the maximum MAP or MAP within a 10% decrease from the maximum one (average estimated dimensionality in the parentheses)

# of nodes	DNML-PUD	DNML-WND	AIC	BIC	MinGE
800	0.500 (4.0)	0.500 (4.0)	**0.708** (5.7)	0.347 (4.0)	0.250 (64.0)
1600	0.500 (4.0)	0.500 (4.0)	**1.000** (8.0)	0.500 (4.0)	0.250 (64.0)
3200	0.542 (4.3)	**1.000** (8.0)	**1.000** (8.0)	0.500 (4.0)	0.250 (64.0)
6400	**1.000** (8.0)	**0.917** (9.3)	0.667 (13.3)	0.500 (4.0)	0.250 (64.0)

Table 4 Average MAPs on WND-16. The bold indicates either the maximum MAP or MAP within a 10% decrease from the maximum one (average estimated dimensionality in the parentheses)

# of nodes	DNML-PUD	DNML-WND	AIC	BIC	MinGE
800	0.264 (4.0)	**0.306 (4.0)**	**0.333** (5.7)	0.250 (4.0)	**0.333** (64.0)
1600	**0.326 (4.0)**	**0.333** (4.3)	**0.347** (8.0)	0.250 (4.0)	**0.333** (64.0)
3200	0.333 (4.3)	0.333 (8.0)	**0.500** (8.0)	0.250 (4.0)	0.333 (64.0)
6400	0.389 (8.0)	**0.917** (14.7)	**0.958** (15.3)	0.250 (4.0)	0.333 (64.0)

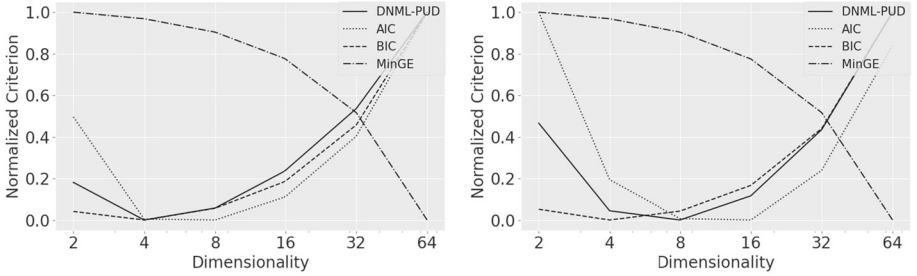

Fig. 3 Results on PUD-8. Left: $n = 800$. Right: $n = 6400$

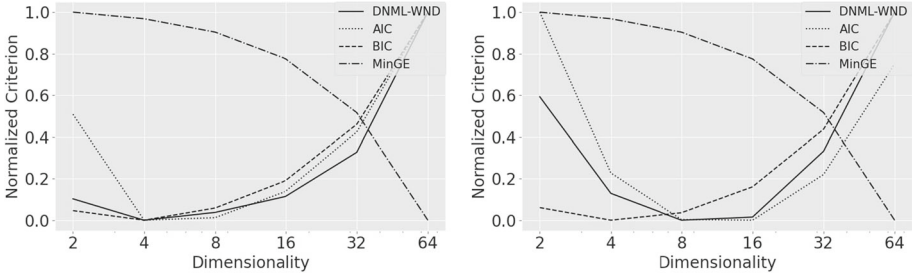

Fig. 4 Results on WND-8. Left: $n = 800$. Right: $n = 6400$

For $n = 800$, AIC, BIC, and DNML selected $D = 4$. Intuitively, a graph with a few nodes is expected to be embedded in low dimensionality, even if its true dimensionality is high. For $n = 6400$, DNML selected the correct dimensionality $D = 8$, whereas AIC and BIC selected incorrect dimensionalities. This implies that the DNML criteria can select the true dimensionality with a sufficient amount of data. For MinGE, it selected the maximum dimensionality $D = 64$ for all cases. This is possibly because MinGE was designed for

Euclidean embeddings, which require larger dimensionality than hyperbolic embeddings for hierarchical structures, as discussed in Sect. 1.1.

Next, we provide a quantitative comparison in terms of mean average precision (MAP) [37]. A MAP is calculated using the ranking of the dimensionalities, which was created in ascending order for each criterion. Furthermore, we applied DNML-PUD to WND dataset and DNML-WND to PUD dataset.

Tables 1, 2, 3, and 4 present the results for PUD-8, PUD-16, WND-8, and WND-16, respectively. Firstly, the MAPs of BIC and MinGE were not so high, and they always selected $D = 4$ and $D = 64$, respectively. Since the selected dimensionalities were constant, BIC and MinGE are less reliable.

For AIC, we observed good performance in many cases; however, it tends to overestimate the true dimensionality for PUD-8 and WND-8 with $n = 6400$. This is because the penalty term of AIC is smaller than those of other criteria. For DNML criteria, in general, when the sample size is sufficiently large, DNML-PUD identifies the true dimensionality of the PUD dataset and the same tendency holds for DNML-WND and the WND dataset. Thus, we concluded that the proposed DNML criteria are more effective than AIC when the true dimensionality of the given graph is low.

Note that, in general, the performance of DNML-PUD in the WND dataset varied, sometimes being better and sometimes worse compared to DNML-WND. Similarly, in the PUD dataset, the performance of DNML-WND also varied, sometimes being better and sometimes worse compared to DNML-PUD. This is because the theoretical properties of the MDL principle are not valid when the generation process of the given data and the assumed generation process for calculating DNML code-length are different from each other. Therefore, this observation is an expected result of the mismatch of the generation process.

4.3 Real-world datasets

We used scientific collaboration networks from [38–40], flight network from https://openflights.org, protein–protein interaction network from [41], and the WN datasets from [27] for our study because they were employed in [16, 42, 43], which are representative studies in the field of hyperbolic embeddings. The experimental results in [16, 42, 43] demonstrated that hyperbolic embeddings outperformed Euclidean embeddings in several graph mining tasks performed on the networks. Therefore, we concluded that they are suitable for comparing our proposed method with others.

4.3.1 Link prediction

The DNML-PUD, DNML-WND, and other model selection criteria were applied to eight real-world graphs. In real-world graphs, the true dimensionality is unknown. Therefore, in this experiment, the link prediction performance for the selected dimensions was evaluated.

Dataset detail We listed the details of eight graphs below.

- *Scientific collaboration networks*. We used `AstroPh`, `CondMat`, `GrQc`, and `HepPh` from [40], `Cora` from [38], and `PubMed` from [39]. These graphs are networks that represent the co-authorship of papers, where an edge exists between two people if they are co-authors.
- *Flight networks*. We used `Airport` from https://openflights.org/. In this graph, nodes represent airports, and edges represent airline routes.

Table 5 Statistics of scientific collaboration networks

Network	# Nodes	# Edges
AstroPh	18,772	198,080
CondMat	23,133	93,468
GrQc	5242	14,490
HepPh	12,008	118,505
Cora	2708	5278
PubMed	19,717	44,327
Airport	3188	18,630
PPI	1870	2240

Table 6 Selected dimensionalities of each method

Network	DNML-PUD	DNML-WND	AIC	BIC	MinGE
AstroPh	9	14	16	7	64
CondMat	11	14	16	7	64
GrQc	7	13	10	4	64
HepPh	9	12	12	5	64
Cora	4	4	6	3	64
PubMed	8	13	14	6	64
Airport	5	5	8	4	64
PPI	4	4	5	3	64

- *Protein–protein interaction (PPI) networks*. We further used PPI from [41]. This graph represents the protein interactions in yeast bacteria.

Furthermore, Table 5 summarizes the statistics of these graphs.

Then, each graph was split into a training set $\boldsymbol{y}_{\text{train}} \subset \boldsymbol{y}$ and test set $\boldsymbol{y}_{\text{test}} \subset \boldsymbol{y}$. The test set $\boldsymbol{y}_{\text{test}}$ comprises the positive and negative samples. First, we sampled 10% of the positive samples from a graph. Subsequently, to generate negative samples, we sampled an equal number of node pairs with no edge connection. Finally, we obtained the training set $\boldsymbol{y}_{\text{train}} := \boldsymbol{y} \backslash \boldsymbol{y}_{\text{test}}$.

For $\boldsymbol{y}_{\text{test}}$, we calculated the *area under the curve* (AUC), which we define as follows. We first calculated the distance of each samples in $\boldsymbol{y}_{\text{test}}$. Subsequently, we calculated the true positive rate and false positive rate with fixed threshold of the distance. Finally, we obtained the receiver operating characteristic (ROC) curve by varying the threshold, and the AUC is defined as the area under the ROC curve.

The AUC was used to quantify the link prediction performance. The candidates of dimensionalities were $\mathcal{D} = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 32, 64\}$.

Results

Figure 5 shows the AUCs of the optimal embeddings associated with $-\log p(\boldsymbol{y}, \boldsymbol{z}; \beta, \gamma, \sigma)$, $-\log p(\boldsymbol{y}, \boldsymbol{z}; \beta, \gamma, \Sigma)$, and $-\log p(\boldsymbol{y} \mid \boldsymbol{z}; \beta, \gamma)$. Figure 6 shows the normalized values of each criterion, and Table 6 shows the selected dimensionalities. The performance at the selected dimensionalities by DNML-PUD and DNML-WND was not the best, and higher dimensionalities tended to yield higher AUCs.

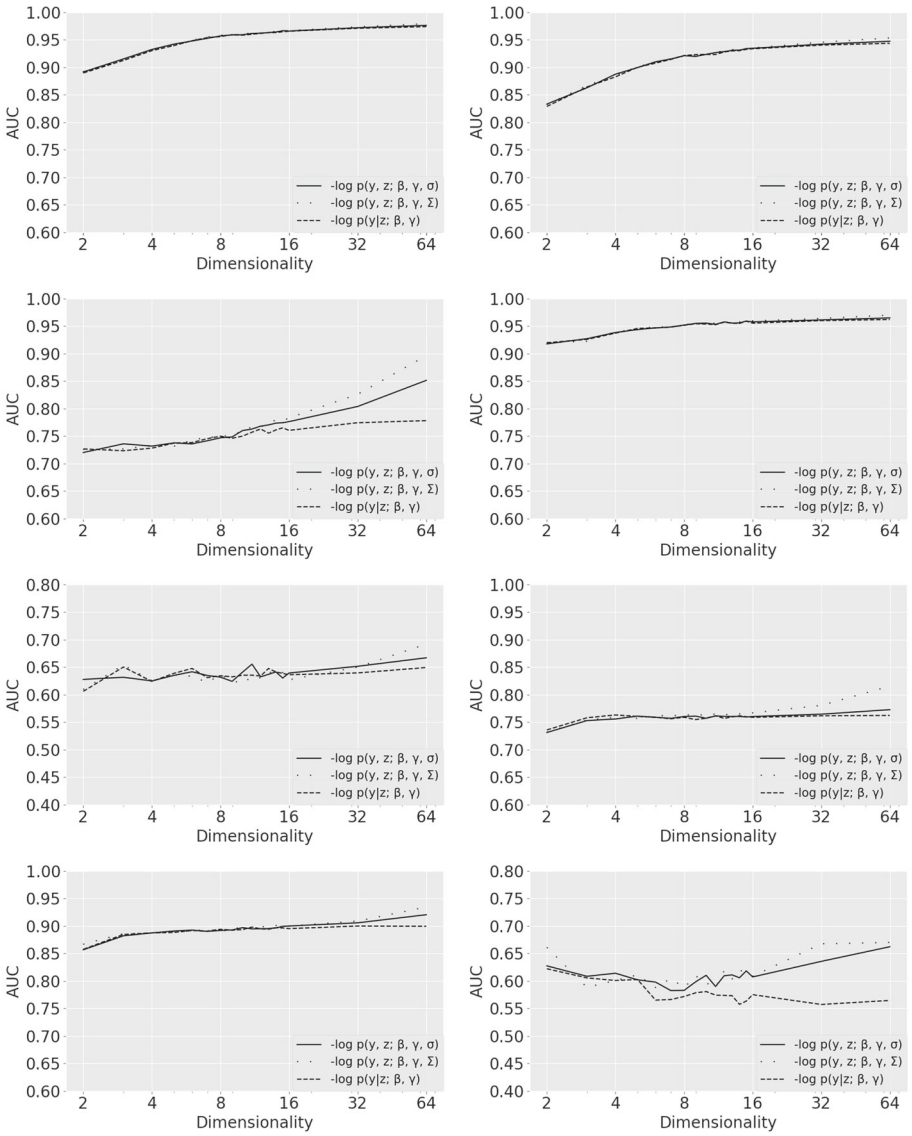

Fig. 5 AUCs on link prediction. First row: `AstroPh` and `CondMat`. Second row: `GrQc` and `HepPh`. Third row: `Cora` and `PubMed`. Fourth row: `Airport` and `PPI`

According to [24], the consistency of the MDL model selection is theoretically guaranteed; that is, the model with the shortest code-length would converge to the true one if it exists. Therefore, the dimensionalities selected by DNML were considered to be close to the dimensionalities of the true probabilistic models that generated the data. However, our results suggest that such dimensionality of the true probabilistic model is not necessarily the best one for link prediction.

In this section, we provide another perspective on the experimental results. As discussed in Sect. 1.1, dimensionality also controls the computation time and memory. Therefore, it

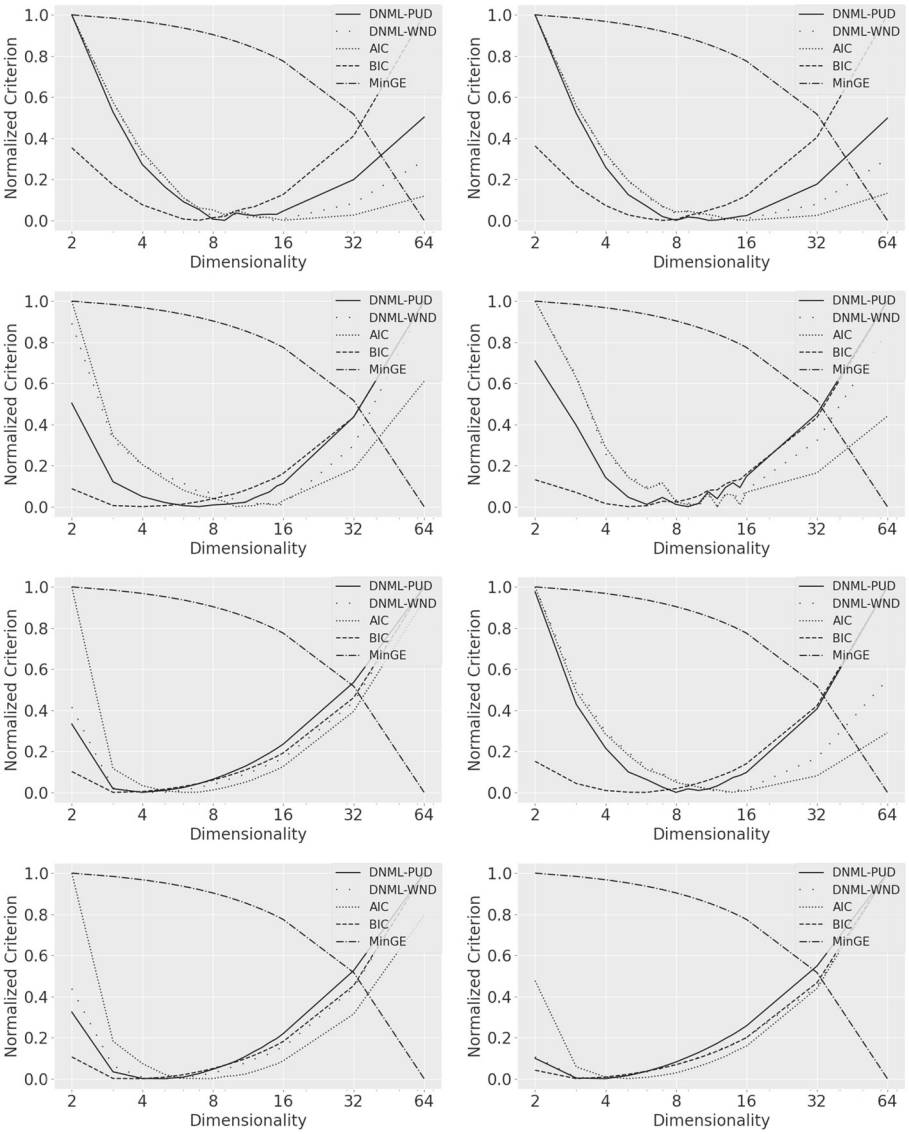

Fig. 6 Results of each criterion. First row: `AstroPh` and `CondMat`. Second row: `GrQc` and `HepPh`. Third row: `Cora` and `PubMed`. Fourth row: `Airport` and `PPI`

is important to select a dimensionality at which a relatively high performance is achieved while maintaining low computational resources (e.g., using embeddings in edge devices). To quantify this, we introduce *conciseness* defined as follows: Let $\mathcal{D} := \{D_1, \ldots, D_N\}$ be the candidates of dimensionalities, AUC_{D_i} be the AUC at dimensionality D_i, $\overline{\mathrm{AUC}}$ be the maximum AUC, and $\epsilon_{\max}$ be a maximum tolerance gap relative to $\overline{\mathrm{AUC}}$. For the selected

Table 7 Average conciseness of each method with $\epsilon_{\max} = 0.050$ and 0.100 (the bold text indicates either the maximum conciseness or conciseness within a 10% decrease from the maximum one)

Network	$\epsilon_{\max}$	DNML-PUD	DNML-WND	AIC	BIC	MinGE
AstroPh	0.050	**0.494**	**0.504**	0.469	0.453	0.000
	0.100	**0.556**	0.512	0.475	**0.560**	0.000
CondMat	0.050	0.391	**0.468**	0.439	0.213	0.000
	0.100	**0.518**	**0.511**	0.475	**0.514**	0.000
GrQc	0.050	0.000	0.000	0.000	0.000	0.000
	0.100	0.000	0.000	0.000	0.000	0.000
HepPh	0.050	**0.539**	**0.524**	0.511	0.477	0.000
	0.100	0.554	0.503	0.497	**0.606**	0.000
Cora	0.050	0.000	0.000	0.087	**0.145**	0.000
	0.100	0.262	0.284	0.417	**0.556**	0.000
PubMed	0.050	0.000	**0.060**	0.000	0.000	0.000
	0.100	**0.300**	0.272	0.219	**0.318**	0.000
Airport	0.050	0.122	0.106	**0.165**	0.059	0.000
	0.100	**0.458**	**0.438**	0.404	**0.458**	0.000
PPI	0.050	0.000	0.000	0.000	0.000	0.000
	0.100	**0.351**	0.247	0.053	0.204	0.000

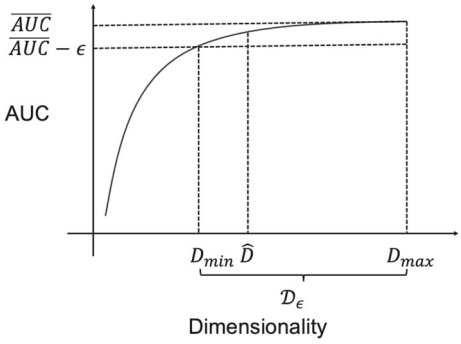

Fig. 7 Typical example of $c(\hat{D}, \epsilon)$

dimensionality $\hat{D}$ of each criterion, the conciseness is provided by:

$$\text{conciseness}(\hat{D}, \epsilon_{\max}) := \frac{1}{\epsilon_{\max} P} \sum_{i=0,1,\ldots,P} c\left(\hat{D}, \frac{i}{P}\epsilon_{\max}\right),$$

$$c(\hat{D}, \epsilon) := \begin{cases} 1 - \frac{\log_2 \hat{D} - \log_2 D_{\min}}{\log_2 D_{\max} - \log_2 D_{\min}} & (\hat{D} \in \mathcal{D}_\epsilon), \\ 0 & (\hat{D} \notin \mathcal{D}_\epsilon), \end{cases}$$

where $\mathcal{D}_\epsilon := \{D_i \in \mathcal{D} \mid \overline{\text{AUC}} - \text{AUC}_{D_i} < \epsilon\}$, $D_{\min} := \min_{D_i \in \mathcal{D}_\epsilon} D_i$, $D_{\max} := \max_{D_i \in \mathcal{D}_\epsilon} D_i$, and $P \in \mathbb{Z}_{\geq 1}$ is the number of candidate points to calculate the conciseness.

Figure 7 shows the typical example of $c(\hat{D}, \epsilon)$. Firstly, the proposed conciseness measure assumes a situation where the AUC improves by increasing the dimensionality, while the extent of improvement diminishes, and with limited computational resource, we want to

Table 8 Statistics of WN datasets

Network	# Nodes	# is-a relationships
WN-mammal	1180	6540
WN-solid	1232	5696
WN-tree	1015	2681
WN-worker	1116	4344
WN-adult	633	2154
WN-instrument	657	2156
WN-leader	788	2505
WN-implement	727	2690

select lower dimensionality while achieving the maximum tolerate AUC. Based on this motivation, the conciseness measure is designed to take high values when $\hat{D} \in \mathcal{D}$ is close to $D_{\min}$.

Since the conciseness significantly depends on $\epsilon_{\max}$, we computed it for $\epsilon_{\max} = 0.050$ and 0.100. Table 7 shows the average conciseness of the selected dimensionalities. To calculate the conciseness, we used the embeddings associated with $-\log p(\boldsymbol{y}, \boldsymbol{z}; \beta, \gamma, \sigma)$ for DNML-PUD, $-\log p(\boldsymbol{y}, \boldsymbol{z}; \beta, \gamma, \Sigma)$ for DNML-WND, and $-\log p(\boldsymbol{y} \mid \boldsymbol{z}; \beta, \gamma)$ for AIC, BIC, and MinGE. Furthermore, we set $\overline{\text{AUC}}$ as the maximum AUC associated with three embeddings.

The best or second best performance was achieved by either DNML-PUD or DNML-WND in many cases. For AIC, the performance was relatively high, but not the best in many cases. For BIC, the performance was relatively high, specifically for $\epsilon_{\max} = 0.100$. This indicates that BIC is effective when the maximum tolerate gap is high. For MinGE, the performance was close to 0 because the selected dimensionality was considerably high. Overall, the proposed method works well in that it identifies dimensionality with a relatively high AUC while maintaining low computational resources.

Note that all the performances were 0 for `GrQc`. This is because higher dimensions achieve considerably higher AUC in `GrQc`, unlike most other networks where increasing dimensions do not significantly improve AUC. In such scenarios, the conciseness measure does not take positive values unless $\epsilon_{\max}$ is set to a considerably high value; however, setting an excessively high tolerate gap (e.g., $\epsilon_{\max} = 0.300$) lacks practical meaning, and it is sufficient to select the maximum dimensionality within the given computational resources.

4.3.2 Preservation of hierarchy

To investigate the extent to which the hierarchical structure was preserved, we used a subset of WordNet [27] closely following the setting in [16, 18].

Dataset detail We first considered the transitive closure of the is-a relationship for all the nouns. Subsequently, we took the subset of the nouns that have "mammal" as a hypernym and selected relations that have "mammal" as a hyponym or hypernym. Finally, we connected the two nouns if they have an is-a relationship. We refer to this dataset as WN-mammal. Similarly, we generated WN-mammal, WN-solid, WN-tree, WN-worker, WN-adult, WN-instrument, WN-leader, and WN-implement. Table 8 summarizes the statistics for these datasets.

Each graph is expected to have a hierarchical structure because a hypernym is often related to many hyponyms. We embedded eight graphs with various dimensionalities and calculated each criterion.

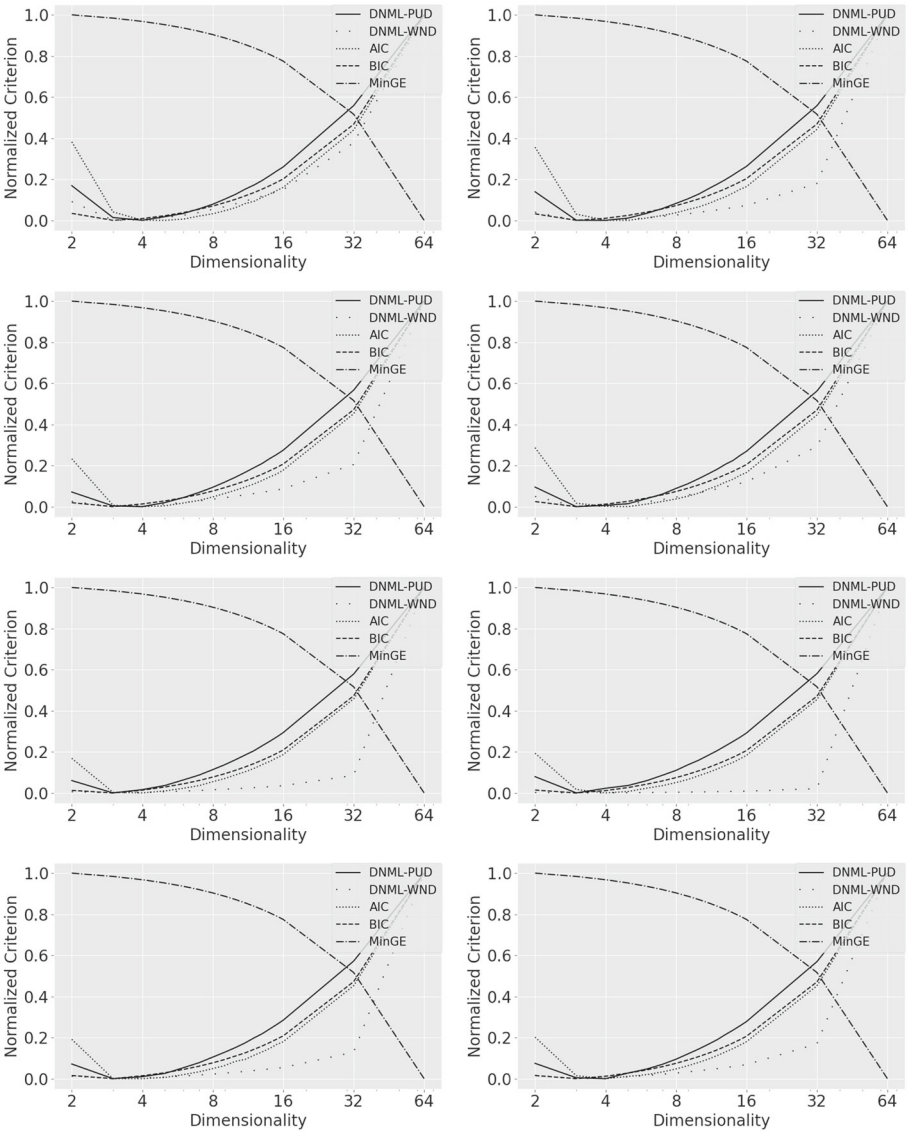

Fig. 8 Results on WN datasets. First row: WN-mammal, WN-solid. Second row: WN-tree and WN-worker. Third row: WN-adult and WN-instrument. Fourth row: WN-leader and WN-implement

Result Figure 8 shows the normalized criteria. The candidates of dimensionalities were $\mathcal{D} = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 32, 64\}$.

Subsequently, we quantified the extent to which the obtained embeddings reflected the true hierarchy of the is-a relation on the data. Following [16], we used the following score function:

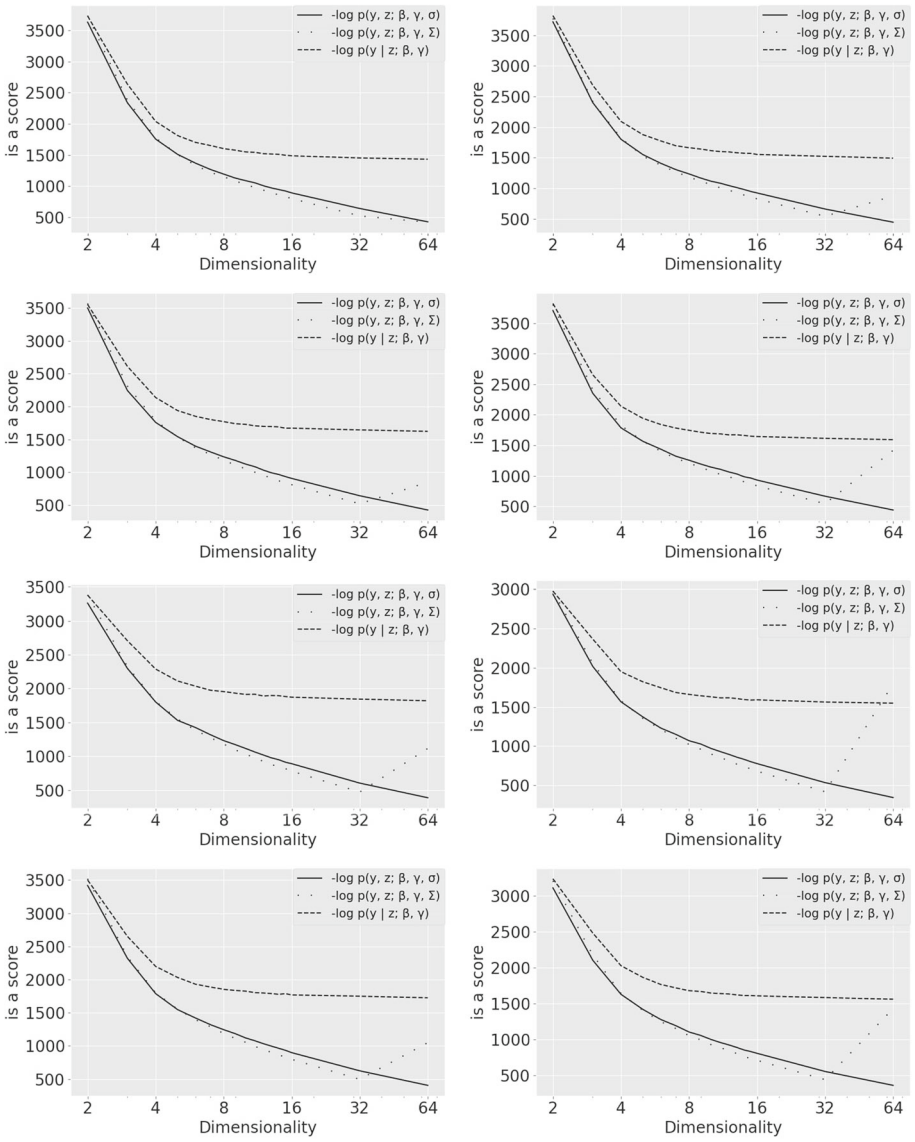

Fig. 9 Is-a scores on WN dataset. First row: WN-mammal, WN-solid. Second row: WN-tree and WN-worker. Third row: WN-adult and WN-instrument. Fourth row: WN-leader and WN-implement

$$\text{is-a-score}(\boldsymbol{u}, \boldsymbol{v}) := (\alpha(r_u - r_v) - 1)d_{\boldsymbol{uv}},$$

where r_u and r_v are the radius coordinates of $\boldsymbol{u}$ and $\boldsymbol{v}$, respectively, and $\alpha > 0$ is a constant. In general, it can be assumed that a hypernym has a lower radial coordinate than its hyponyms. Thus, $\alpha(r_u - r_v)$ acts as a penalty when v, which is a hypernym of u, is lower in the embedding hierarchy. Therefore, the score is expected to be high when the embedding reflects the true hierarchy of data. Figure 9 shows the average score over all is-a relation pairs for each dimensionality with $\alpha = 100$. Overall, the lower dimensionality achieved higher is-a scores.

Table 9 Average benefits on WN datasets (the bold text indicates the maximum one)

DNML-PUD	DNML-WND	AIC	BIC	MinGE
0.604	**0.708**	0.440	**0.708**	0.000

Next, we provide a quantitative comparison of the *benefits*. With an estimated dimensionality $\hat{D}$ and a maximum tolerance gap T_{gap}, the benefit is defined as follows:

$$b(\hat{D}, D_{\text{best}}) := \max\left\{0, 1 - \frac{|\log_2 \hat{D} - \log_2 D_{\text{best}}|}{T_{\text{gap}}}\right\},$$

where D_{best} is the dimensionality at which the best is-a score is achieved. It has a high value when the estimated dimensionality is close to D_{best}.

Table 9 shows the average benefits over the WN datasets. Note that D_{best} was 2 for all datasets, and we set $T_{\text{gap}} = 2$ in the experiments. The DNML-WND and BIC achieved the best results. This result implies that DNML-WND and BIC selected the dimensionality that reflects the true hierarchical structure of the particular graph.

4.4 Discussion

We summarize the experimental results. Firstly, as a general trend, the AIC usually selects higher dimensionality, the BIC lower dimensionality, and the DNML criteria middle dimensionality. This is due to the relative magnitude of the penalty terms. MinGE always selected the largest dimensionality in all the experiments. This is possibly because MinGE was designed for Euclidean embeddings, which require higher dimensionality than hyperbolic embeddings for hierarchical structures, as discussed in Sect. 1.1.

The performance of the AIC was relatively well on the first and second tasks, but not on the third task. In contrast, the BIC showed high performance in the third task, but low performance in the first and second tasks. The DNML criterion does not necessarily give the best performance, but it often gives the best performance or the next-best performance for all tasks. Therefore, it can be concluded that the performance of the proposed DNML criteria was good on average for all tasks.

5 Conclusion and future work

In this study, we proposed a dimensionality selection method for hyperbolic embeddings based on the MDL principle. We demonstrated that there is a non-identifiability problem for hyperbolic embeddings. Therefore, we employed the latent variable model of hyperbolic embeddings using PUDs and WNDs to formulate dimensionality selection as the statistical model selection for latent variable models. Within this formulation, we proposed the DNML code-length criterion for dimensionality selection based on the MDL principle. For artificial datasets, we experimentally demonstrated that our method is effective when the true dimensionality is low. For real-world datasets, we used the scientific collaboration networks and WN datasets. For the scientific collaboration networks, we demonstrated that the proposed method selected simple probabilistic models while maintaining AUCs. For the WN datasets, we confirmed that the proposed method selects the dimensionality which preserves the true hierarchy of graphs. Overall, the proposed method performed well in average.

Note that we cannot directly use the dimensionality selected by the proposed method in hyperbolic neural networks methods, such as [43, 44]. Because the probabilistic models used in the proposed method differ from those used in [43, 44], the optimal dimensionalities within them are inherently different. This implies that using the dimensionality selected by the proposed method in [43, 44] rationales the rationales of the DNML code length and MDL principle. However, we would like to emphasize that even in hyperbolic neural networks, it is possible to consider latent variable models and derive DNML code-lengths following a similar procedure to our study. Specifically, we can treat the parameters and outputs of hidden layers of hyperbolic neural networks as latent variables. In this sense, the proposed method can be generalized. However, as indicated in "Appendix B," the derivation of the DNML code-length requires significant effort and is not straightforward. Therefore, we consider the extension of our proposed method to hyperbolic neural networks as future work.

The latent variable model approach adopted in this study is promising and is not limited to hyperbolic space. In the future, we plan to build a methodology for dimensionality selection in Euclidean and spherical embeddings by introducing latent variable for their corresponding space [45, 46].

Acknowledgements This work was partially supported by JST KAKENHI JP19H0111.

Author Contributions R.Y. contributed to conceptualization, investigation, data curation, writing—original draft preparation, visualization, and project administration and provided software; R.Y., Y.I., and K. Y. were involved in methodology, validation, formal analysis, and writing—review and editing; and K.Y. contributed to supervision and funding acquisition.

Funding Open access funding provided by The University of Tokyo.

Declarations

Conflict of interest The authors declare no conflicts of interest.

Appendix A. Derivation of $J(z_{1:D} : u_i)$

For notational simplicity, we omit i from $z_{i,1:D}$ and u_i in this section. To drive $J(z_{1:D} : u)$, we introduce intermediate variable $u' := (r', \theta'_1, \ldots, \theta'_{D-1})^\top$. The coordinate transformations are as follows:

$$\begin{cases} z_1 = r' \cos\theta'_1, \\ z_2 = r' \sin\theta'_1 \cos\theta'_2, \\ \quad \vdots \\ z_{D-1} = r' \sin\theta'_1 \sin\theta'_2 \cdots \sin\theta'_{D-2} \cos\theta'_{D-1}, \\ z_D = r' \sin\theta'_1 \sin\theta'_2 \cdots \sin\theta'_{D-2} \sin\theta'_{D-1}. \end{cases}$$

$$\begin{cases} r' = \sinh r, \\ \theta_1' = \theta_1, \\ \theta_2' = \theta_2, \\ \quad \vdots \\ \theta_{D-1}' = \theta_{D-1}. \end{cases}$$

Subsequently, $J(z_{1:D} : u) = J(z_{1:D} : u')J(u' : u)$, where $J(z_{1:D} : u')$ and $J(u' : u)$ are the corresponding Jacobians. Jacobian $J(z_{1:D} : u')$ is well known (e.g., [47]):

$$J(z_{1:D} : u') = {r'}^{D-1} \prod_{i=1}^{D-2} \sin^{D-i-1} \theta_i'.$$

Jacobian $J(u' : u)$ is calculated by definition as

$$J(u' : u) = \cosh r.$$

Thus, we obtain the following:

$$J(z_{1:D} : u) = \sinh^{D-1} r \cosh r \prod_{i=1}^{D-2} \sin^{D-i-1} \theta_i.$$

Appendix B. Derivation of DNML code-length

B.1 Derivation of $L_{\mathrm{NML}}(\boldsymbol{y} \mid \boldsymbol{z})$

For $L_{\mathrm{NML}}(\boldsymbol{y} \mid \boldsymbol{z})$, following Chap. 5 in [48], we approximate parametric complexity as follows:

$$\begin{aligned} &\log \sum_{\bar{\boldsymbol{y}}} p(\bar{\boldsymbol{y}} \mid \boldsymbol{z}; \hat{\beta}(\bar{\boldsymbol{y}}, \boldsymbol{z}), \hat{\gamma}(\bar{\boldsymbol{y}}, \boldsymbol{z})) \\ &\approx \log \frac{n(n-1)}{4\pi} + \log \int_{\gamma_{\min}}^{\gamma_{\max}} \int_{\beta_{\min}}^{\beta_{\max}} \sqrt{|I_n(\beta, \gamma)|} \mathrm{d}\beta \mathrm{d}\gamma, \end{aligned}$$

where

$$I_n(\beta, \gamma) = E_{\beta\gamma} \left[\frac{2}{n(n-1)} \begin{pmatrix} \frac{\partial^2}{\partial \beta^2} L(\beta, \gamma) & \frac{\partial^2}{\partial \beta \partial \gamma} L(\beta, \gamma) \\ \frac{\partial^2}{\partial \beta \partial \gamma} L(\beta, \gamma) & \frac{\partial^2}{\partial \gamma^2} L(\beta, \gamma) \end{pmatrix} \right],$$

where $L(\beta, \gamma) := \log p(\boldsymbol{y} \mid \boldsymbol{z}; \beta, \gamma)$ and $E_{\beta\gamma}[\cdot]$ is the expectation with respect to $p(\boldsymbol{y} \mid \boldsymbol{z}; \beta, \gamma)$.

In the following, we calculate $I_n(\beta, \gamma)$: We rewrite $p(y_{ij} \mid z_i, z_j; \beta, \gamma)$ as

$$\begin{aligned} p(y_{ij} \mid z_i, z_j; \beta, \gamma) &= p_{ij}^{y_{ij}} (1 - p_{ij})^{(1-y_{ij})} \\ &= (1 - p_{ij}) \left(\frac{p_{ij}}{1 - p_{ij}} \right)^{y_{ij}}, \end{aligned}$$

where $p_{ij} = 1/(1 + \exp(\beta d_{z_i z_j} - \gamma))$. This form is a logistic function with the constraints $\beta \in [\beta_{\min}, \beta_{\max}]$ and $\gamma \in [\gamma_{\min}, \gamma_{\max}]$. Then, $L(\beta, \gamma)$ is rewritten as

$$\begin{aligned} L(\beta,\gamma) &= \sum_{(i,j)\in\Lambda_n} \left\{ -y_{ij} \log(1-p_{ij}) - \log \frac{p_{ij}}{1-p_{ij}} \right\} \\ &= \sum_{(i,j)\in\Lambda_n} \left\{ \log(1+\exp(-\beta d_{z_i z_j} + \gamma)) + y_{ij}(\beta d_{z_i z_j} - \gamma) \right\}. \end{aligned}$$

Hence, we obtain

$$\begin{aligned} \frac{\partial L(\beta,\gamma)}{\partial \beta} &= \sum_{(i,j)\in\Lambda_n} \left\{ \frac{-d_{z_i z_j} \exp(-\beta d_{z_i z_j} + \gamma)}{1+\exp(-\beta d_{z_i z_j} + \gamma)} + y_{ij} d_{z_i z_j} \right\} \\ &= \sum_{(i,j)\in\Lambda_n} \left\{ \frac{-d_{z_i z_j}}{1+\exp(\beta d_{z_i z_j} - \gamma)} + y_{ij} d_{z_i z_j} \right\}, \\ \frac{\partial^2 L(\beta,\gamma)}{\partial \beta^2} &= \sum_{(i,j)\in\Lambda_n} \frac{d_{z_i z_j}^2 \exp(\beta d_{z_i z_j} - \gamma)}{\left(1+\exp(\beta d_{z_i z_j} - \gamma)\right)^2} \\ &= \sum_{(i,j)\in\Lambda_n} \frac{d_{z_i z_j}^2}{4\cosh^2\left(\frac{\beta d_{z_i z_j} - \gamma}{2}\right)}, \\ \frac{\partial^2 L(\beta,\gamma)}{\partial \beta \partial \gamma} &= \sum_{(i,j)\in\Lambda_n} \frac{-d_{z_i z_j} \exp(\beta d_{z_i z_j} - \gamma)}{1+\exp(\beta d_{z_i z_j} - \gamma)} \\ &= \sum_{(i,j)\in\Lambda_n} \frac{-d_{z_i z_j}}{4\cosh^2\left(\frac{\beta d_{z_i z_j} - \gamma}{2}\right)}, \\ \frac{\partial L(\beta,\gamma)}{\partial \gamma} &= \sum_{(i,j)\in\Lambda_n} \left\{ \frac{\exp(-\beta d_{z_i z_j} + \gamma)}{1+\exp(-\beta d_{z_i z_j} + \gamma)} - y_{ij} \right\} \\ &= \sum_{(i,j)\in\Lambda_n} \left\{ \frac{1}{1+\exp(\beta d_{z_i z_j} - \gamma)} - y_{ij} \right\}, \\ \frac{\partial^2 L(\beta,\gamma)}{\partial \gamma^2} &= \sum_{(i,j)\in\Lambda_n} \frac{\exp(\beta d_{z_i z_j} - \gamma)}{\left(1+\exp(\beta d_{z_i z_j} - \gamma)\right)^2} \\ &= \sum_{(i,j)\in\Lambda_n} \frac{1}{4\cosh^2\left(\frac{\beta d_{z_i z_j} - \gamma}{2}\right)}. \end{aligned}$$

Since all terms are independent of y_{ij}, we obtain

$$I_n(\beta,\gamma) = \frac{2}{n(n-1)} \begin{pmatrix} \sum_{(i,j)\in\Lambda_n} \frac{d_{z_i z_j}^2}{4\cosh^2\left(\frac{\beta d_{z_i z_j} - \gamma}{2}\right)} & \sum_{(i,j)\in\Lambda_n} \frac{-d_{z_i z_j}}{4\cosh^2\left(\frac{\beta d_{z_i z_j} - \gamma}{2}\right)} \\ \sum_{(i,j)\in\Lambda_n} \frac{-d_{z_i z_j}}{4\cosh^2\left(\frac{\beta d_{z_i z_j} - \gamma}{2}\right)} & \sum_{(i,j)\in\Lambda_n} \frac{1}{4\cosh^2\left(\frac{\beta d_{z_i z_j} - \gamma}{2}\right)} \end{pmatrix}.$$

B.2 Derivation of $L_{\mathrm{NML}}(z)$ of PUDs

For $L_{\mathrm{NML}}(z)$, we first show that $\log \int p(\bar{z};\hat{\sigma}(\bar{z}))\mathrm{d}\bar{z}_{1:D} := \log \int p(\bar{\boldsymbol{u}};\hat{\sigma}(\bar{\boldsymbol{u}}))\mathrm{d}\bar{\boldsymbol{u}}$, where $\bar{\boldsymbol{u}} = \{\bar{u}_i\}_{i\in[n]}$, and $\bar{u}_i := (\bar{r}_i, \bar{\theta}_{i,1}, \ldots, \bar{\theta}_{i,D-1})^\top$ is the polar coordinates of each datum. Since $J(\bar{z}_{i,1:D} : \bar{u}_i)$ does not depend on σ, the following equation holds.

$$\begin{aligned}
\hat{\sigma}(\bar{\boldsymbol{u}}) &= \underset{\sigma\in[\sigma_{\min},\sigma_{\max}]}{\arg\max}\ p(\bar{\boldsymbol{u}};\sigma) \\
&= \underset{\sigma\in[\sigma_{\min},\sigma_{\max}]}{\arg\max}\ p(\bar{\boldsymbol{z}};\sigma)\prod_{i=1}^{n} J(\bar{z}_{i,1:D} : \bar{u}_i) \\
&= \underset{\sigma\in[\sigma_{\min},\sigma_{\max}]}{\arg\max}\ p(\bar{\boldsymbol{z}};\sigma) \\
&= \hat{\sigma}(\bar{\boldsymbol{z}}).
\end{aligned}$$

Thus,

$$\begin{aligned}
\log \int p(\bar{\boldsymbol{u}};\hat{\sigma}(\bar{\boldsymbol{u}}))\mathrm{d}\bar{\boldsymbol{u}} &= \log \int p(\bar{\boldsymbol{z}};\hat{\sigma}(\bar{\boldsymbol{u}}))\prod_{i=1}^{n} J(\bar{z}_{i,1:D} : u_i)\mathrm{d}\bar{\boldsymbol{u}} \\
&= \log \int p(\bar{\boldsymbol{z}};\hat{\sigma}(\bar{\boldsymbol{z}}))\prod_{i=1}^{n} J(\bar{z}_{i,1:D} : u_i)\mathrm{d}\bar{\boldsymbol{u}} \\
&= \log \int p(\bar{\boldsymbol{z}};\hat{\sigma}(\bar{\boldsymbol{z}}))\mathrm{d}\bar{\boldsymbol{z}}_{1:D}.
\end{aligned}$$

Thereafter, we use the asymptotic approximation of parametric complexity according to Rissanen [49]:

$$\log \int p(\bar{\boldsymbol{u}};\hat{\sigma}(\bar{\boldsymbol{u}}))\mathrm{d}\bar{\boldsymbol{u}} \approx \frac{1}{2}\log\frac{n}{2\pi} + \log\int_{\sigma_{\min}}^{\sigma_{\max}} \sqrt{|I(\sigma)|}\mathrm{d}\sigma,$$

where $I(\sigma)$ denotes Fisher information

$$I(\sigma) := \lim_{n\to\infty}\frac{1}{n}E_\sigma\left[-\frac{\partial^2 \log p(\boldsymbol{u};\sigma)}{\partial\sigma^2}\right] = E_\sigma\left[-\frac{\partial^2 \log p(u_i;\sigma)}{\partial\sigma^2}\right].$$

In the following, we calculate $I(\sigma)$.

The negative logarithm of the likelihood for $u_i = (r, \theta_1, \ldots, \theta_{D-1})^\top$ is

$$L(\sigma) := -\log\frac{\sinh^{D-1}(\sigma r)}{C_D(\sigma)} - \sum_{j=1}^{D-2}\log\frac{\sin^{D-1-j}\theta_j}{I_{D,j}} - \log\frac{1}{2\pi}.$$

By interchanging the derivative and integral, we obtain

$$\frac{\partial L(\sigma)}{\partial\sigma} = -(D-1)\frac{r\cosh\sigma r}{\sinh\sigma r} + \frac{\int_0^R (D-1)\bar{r}\cosh(\sigma\bar{r})\sinh^{D-2}(\sigma\bar{r})\mathrm{d}\bar{r}}{C_D(\sigma)}.$$

Similarly, we get

$$\frac{\partial^2 L(\sigma)}{\partial\sigma^2} = (D-1)\frac{r^2}{\sinh^2(\sigma r)}$$

$$+\frac{(D-1)}{C_D(\sigma)}\int_0^R\left(\bar{r}^2\sinh^{D-1}(\sigma\bar{r})+(D-2)\bar{r}^2\cosh^2(\sigma\bar{r})\sinh^{D-3}(\sigma\bar{r})\right)\mathrm{d}\bar{r}$$
$$-\left\{\frac{\int_0^R(D-1)\bar{r}\cosh(\sigma\bar{r})\sinh^{D-2}(\sigma\bar{r})\mathrm{d}\bar{r}}{C_D(\sigma)}\right\}^2.$$

The second and third terms are independent of r. The expectation of the first term with respect to r is calculated as follows.

$$(D-1)\int_0^R\frac{\sinh^{D-1}(\sigma\bar{r})}{C_D(\sigma)}\cdot\frac{\bar{r}^2}{\sinh^2(\sigma\bar{r})}\mathrm{d}\bar{r}=\frac{D-1}{C_D(\sigma)}\int_0^R\bar{r}^2\sinh^{D-3}(\sigma\bar{r})\mathrm{d}\bar{r}.$$

Finally, we derive the following:

$$I(\sigma)=(D-1)^2\frac{\int_0^R r^2\cosh^2(\sigma r)\sinh^{D-3}(\sigma r)\mathrm{d}r}{C_D(\sigma)}$$
$$-\left\{\frac{\int_0^R(D-1)r\cosh(\sigma r)\sinh^{D-2}(\sigma r)\mathrm{d}r}{C_D(\sigma)}\right\}^2.$$

B.3 Derivation of $L_{\mathrm{NML}}(\boldsymbol{z})$ of WNDs

Similar to the discussion presented in Sect. B.2, we obtain $\log\int p(\bar{\boldsymbol{z}};\hat{\Sigma}(\bar{\boldsymbol{z}}))\mathrm{d}\bar{\boldsymbol{z}}_{1:D}:=\log\int p(\bar{\boldsymbol{v}};\hat{\Sigma}(\bar{\boldsymbol{v}}))\mathrm{d}\bar{\boldsymbol{v}}$, where $\bar{\boldsymbol{v}}:=\{\bar{v}_i\}_{i\in[n]}$, and $\bar{v}_i:=(\bar{v}_{i,1},\dots\bar{v}_{i,D})^\top$. The following derivation of $\log\int p(\bar{\boldsymbol{v}};\hat{\Sigma}(\bar{\boldsymbol{v}}))\mathrm{d}\bar{\boldsymbol{v}}$ depends heavily on [50, 51]: Let $\hat{\Sigma}:=\frac{1}{n}\sum_{i\in[n]}v_iv_i^\top$ be the maximum likelihood estimator for Σ. The probability density of $\hat{\Sigma}$ is well known (e.g., Example 2.7 in [47]) and is given by

$$g(\hat{\Sigma};\Sigma)=\frac{|\hat{\Sigma}|^{\frac{n-D-1}{2}}}{2^{\frac{nD}{2}}|\frac{1}{n}\Sigma|^{\frac{n}{2}}\Gamma_D\left(\frac{n}{2}\right)}\exp\left(-\frac{1}{2}\mathrm{tr}(n\Sigma^{-1}\hat{\Sigma})\right),$$

where $\Gamma_D(x):=\pi^{\frac{D(D-1)}{4}}\prod_{j\in[D]}\Gamma\left(x+\frac{1-j}{2}\right)$, and $\Gamma(\cdot)$ is the gamma function. Using Rissanen's g-function [24], the parametric complexity of $L_{\mathrm{NML}}(\boldsymbol{z})$ can be rewritten as:

$$\int p(\bar{\boldsymbol{v}};\hat{\Sigma}(\bar{\boldsymbol{v}}))\mathrm{d}\bar{\boldsymbol{v}}=\int g(\hat{\Sigma};\hat{\Sigma})\mathrm{d}\hat{\Sigma},$$

where $d\hat{\Sigma}=\prod_{i=1}^D\prod_{j=i}^D d\Sigma_{ij}$.

We also rewrite $g(\hat{\Sigma};\hat{\Sigma})$ as:

$$g(\hat{\boldsymbol{\lambda}}):=\frac{n^{\frac{nD}{2}}}{2^{\frac{nD}{2}}\Gamma_D\left(\frac{n}{2}\right)e^{\frac{nD}{2}}}\prod_{j\in[D]}\hat{\lambda}_j^{\frac{-D-1}{2}},$$

where $\hat{\boldsymbol{\lambda}}:=\{\hat{\lambda}_i\}_{i\in[D]}$ is the set of the eigenvalues of $\hat{\Sigma}$ with $0\le\hat{\lambda}_1\le\cdots\le\hat{\lambda}_D$.

To avoid the divergence of parametric complexity, we restrict the data domain of $\boldsymbol{v}$ and $\boldsymbol{z}$ as follows:

$$\bar{\mathcal{Z}}^D(\boldsymbol{\epsilon}_1,\boldsymbol{\epsilon}_2):=\{\boldsymbol{v}\mid\forall i\in[n],v_i\in\mathbb{R}^D,\forall j\in[D],\epsilon_{1j}\le\hat{\lambda}_j(\boldsymbol{v})\le\epsilon_{2j}\},$$
$$\mathcal{Z}^D(\boldsymbol{\epsilon}_1,\boldsymbol{\epsilon}_2):=\{\boldsymbol{z}\mid\forall i\in[n],\exists v_i\in\bar{\mathcal{Z}}^D(\boldsymbol{\epsilon}_1,\boldsymbol{\epsilon}_2),z_i=\mathrm{Exp}_{\boldsymbol{\mu}_0}([0,v_i^\top]^\top)\}.$$

We then obtain the following:

$$\begin{aligned}
\int g(\hat{\Sigma};\hat{\Sigma})\mathrm{d}\hat{\Sigma} &\leq \frac{\pi^{\frac{D^2}{2}}}{\Gamma_D\left(\frac{D}{2}\right)}\int g(\hat{\boldsymbol{\lambda}})\prod_{1\leq i<j\leq D}|\hat{\lambda}_i-\hat{\lambda}_j|\mathrm{d}\hat{\boldsymbol{\lambda}}\\
&< \frac{\pi^{\frac{D^2}{2}}\prod_{i\in[D-1]}\epsilon_{2i}^{D-i}}{\Gamma_D\left(\frac{D}{2}\right)}\int g(\hat{\boldsymbol{\lambda}})\mathrm{d}\hat{\boldsymbol{\lambda}}\\
&= \frac{\pi^{\frac{D^2}{2}}\prod_{i\in[D-1]}\epsilon_{2i}^{D-i}}{\Gamma_D\left(\frac{D}{2}\right)\Gamma_D\left(\frac{n}{2}\right)}\left(\frac{n}{2e}\right)^{\frac{nD}{2}}\prod_{j\in[D]}\int_{\epsilon_{1j}}^{\epsilon_{2j}}\hat{\lambda}_j^{\frac{-D-1}{2}}\mathrm{d}\hat{\lambda}_j\\
&= \frac{\pi^{\frac{D^2}{2}}\prod_{i\in[D-1]}\epsilon_{2i}^{D-i}}{\Gamma_D\left(\frac{D}{2}\right)\Gamma_D\left(\frac{n}{2}\right)}\left(\frac{n}{2e}\right)^{\frac{nD}{2}}\left(\frac{2}{D-1}\right)^{D}\prod_{j\in[D]}\left(\epsilon_{1j}^{\frac{1-D}{2}}-\epsilon_{2j}^{\frac{1-D}{2}}\right)\\
&\leq \frac{\pi^{\frac{D^2}{2}}\prod_{i\in[D-1]}\epsilon_{2i}^{D-i}\prod_{j\in[D]}\epsilon_{1j}^{\frac{1-D}{2}}}{\Gamma_D\left(\frac{D}{2}\right)\Gamma_D\left(\frac{n}{2}\right)}\left(\frac{n}{2e}\right)^{\frac{nD}{2}}\left(\frac{2}{D-1}\right)^{D}.
\end{aligned}$$

The first inequality is derived from Thm. 2.13 [47].

The main difference between our upper bound and that in [50, 51] is as follows:

- We fixed the mean to the origin of the Euclidean space, whereas the previous studies did not.
- We removed the restriction $\epsilon_{\max} < 1$ and $\pi^{\frac{D^2}{2}}\epsilon_{\max}^{D(D-1)}/\Gamma_D(\frac{D}{2}) < 1$ from $\bar{\mathcal{Z}}^D(\boldsymbol{\epsilon}_1,\boldsymbol{\epsilon}_2)$, where $\epsilon_{\max} > 0$ is set such that $\epsilon_{2j} \leq \epsilon_{\max}$ for all $j \in [D]$. Below we explain the reason. In the discussion in [51], the difference between the code-lengths of any two data is scale-invariant in Gaussian mixture models (GMMs). In other words, the result of model selection is scale-invariant. Thus, we can perform model selection on rescaled data on GMMs, e.g., to multiply $1/\alpha$, etc. However, in hyperbolic embeddings, the difference of the code-lengths is not scale-invariant due to its non-Euclidean nature. Thus, in our model, it is necessary to remove the restrictions and set $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$ to appropriate values such that the raw data are included in $\bar{\mathcal{Z}}^D(\boldsymbol{\epsilon}_1,\boldsymbol{\epsilon}_2)$.

Appendix C Code and experimental details

C.1 Environment

Table 10 lists the computing environments used in our experiments.

C.2 Code

The code used in our experiments can be found in the following GitHub repository https://github.com/IbarakikenYukishi/poincare_embedding/tree/KAIS2023. Please refer to the README for the instructions.

Heuristics for estimating likelihood and Fisher information
For the calculation of each criterion, because $-\log p(\boldsymbol{y} \mid \boldsymbol{z};\hat{\beta}(\boldsymbol{y},\boldsymbol{z}),\hat{\gamma}(\boldsymbol{y},\boldsymbol{z}))$ for large n requires considerable computational time to calculate, we approximated it as follows:

$$-\log p(\boldsymbol{y} \mid \boldsymbol{z};\hat{\beta}(\boldsymbol{y},\boldsymbol{z}),\hat{\gamma}(\boldsymbol{y},\boldsymbol{z})) \approx -\frac{|\boldsymbol{y}|}{|\boldsymbol{y}'|}\log p(\boldsymbol{y}' \mid \boldsymbol{z};\hat{\beta}(\boldsymbol{y},\boldsymbol{z}),\hat{\gamma}(\boldsymbol{y},\boldsymbol{z})),$$

Table 10 Computing environment in our experiments

Attribute	Environment
OS	Ubuntu 18.04 LTS
CPU	AMD EPYC 7452 (32 core) 2.35 GHz
GPU	GeForce RTX 3090 × 4
Memory	512GB
Programming language	Python: 3.6.9. with Anaconda
Library	PyTorch v1.8.1 with CUDA 11.1

where $\boldsymbol{y}'$ is sampled uniformly at random over $\boldsymbol{y}$. Similarly, we approximated $I_n(\beta, \gamma)$ as follows:

$$I_n(\beta,\gamma) \approx \frac{2}{n'(n'-1)} \begin{pmatrix} \sum_{(i,j)\in\Lambda_{S_{n'}}} \frac{d_{z_i z_j}^2}{4\cosh^2\left(\frac{\beta d_{z_i z_j}-\gamma}{2}\right)} & \sum_{(i,j)\in\Lambda_{S_{n'}}} \frac{-d_{z_i z_j}}{4\cosh^2\left(\frac{\beta d_{z_i z_j}-\gamma}{2}\right)} \\ \sum_{(i,j)\in\Lambda_{S_{n'}}} \frac{-d_{z_i z_j}}{4\cosh^2\left(\frac{\beta d_{z_i z_j}-\gamma}{2}\right)} & \sum_{(i,j)\in\Lambda_{S_{n'}}} \frac{1}{4\cosh^2\left(\frac{\beta d_{z_i z_j}-\gamma}{2}\right)} \end{pmatrix},$$

where $S_{n'} \subset [n]$ was sampled uniformly at random over $[n]$, $|S_{n'}| = n'$, and $\Lambda_{S_{n'}} := \{(i, j) \mid (i, j) \in S_{n'} \times S_{n'}, i < j\}$.

C.3 Training detail

For all experiments, the training details were the same.

Embeddings
First, the Cartesian coordinates of each node were independently initialized uniformly at random in $[-0.001, 0.001]^{D+1} \subset \mathbb{R}^{(D+1)\times(D+1)}$. They were then projected onto the hyperboloid plane using Eq. (1).

When learning the embeddings, ten negative samples were sampled per positive sample for mini-batches. The number of epochs was 800, and we set $\eta_\beta^{(t)} = 0.001$ and $\eta_\gamma^{(t)} = 0.001$ for all epochs. Similar to [16], the learning process of embeddings is composed of two steps.

- In the first step, $-\log p(\boldsymbol{y} \mid \boldsymbol{z}; \beta, \gamma)$ was optimized for all models with learning rate $\eta_z^{(t)} = 0.1$.
- In the second step, we optimized the joint likelihoods $-\log p(\boldsymbol{y}, \boldsymbol{z}; \beta, \gamma, \sigma)$ and $-\log p(\boldsymbol{y}, \boldsymbol{z}; \beta, \gamma, \Sigma)$ for DNML-PUD and DNML-WND, respectively, and $-\log p(\boldsymbol{y} \mid \boldsymbol{z}; \beta, \gamma)$ for the AIC and BIC. The learning rate $\eta_z^{(t)}$ was 34.375.

The first step and second step are composed of 100 and 700 epochs, respectively, for the embeddings associated with $-\log p(\boldsymbol{y} \mid \boldsymbol{z}; \beta, \gamma)$ and $-\log p(\boldsymbol{y}, \boldsymbol{z}; \beta, \gamma, \sigma)$. For the embeddings associated with $-\log p(\boldsymbol{y}, \boldsymbol{z}; \beta, \gamma, \Sigma)$, the first step and second step are composed of 110 and 690 epochs, respectively.

Parameters
We set the parameters as follows: $\sigma_{\max} = 1.0$, $\sigma_{\min} = 0.1$, $\beta_{\max} = 10.0$, $\beta_{\min} = 0.1$, $\gamma_{\max} = 10.0$, $\gamma_{\min} = 0.1$, $\beta^{(0)} = 1.0$, $\gamma^{(0)} = \log n$, $\sigma^{(0)} = 1.0$, and $C = 1000$. Then, R was taken as the same value when the graph was generated for the artificial datasets. For real-world datasets, $R = \log n + 5$, where n denotes the number of nodes. To calculate the upper bound

on parametric complexity, we set $\epsilon_{1j} = 0.000001$ and $\epsilon_{2j} = 1000$ for all $j \in [D]$. For numerical integration, Gaussian quadrature [52] was used.

References

1. Theocharidis A, Van Dongen S, Enright AJ, Freeman TC (2009) Network visualization and analysis of gene expression data using biolayout express 3D. Nat Protoc 4(10):1535–1550
2. Freeman LC (2000) Visualizing social networks. J Soc Struct 1(1):4
3. Cancho RFI, Solé RV (2001) The small world of human language. Proc R Soc Lond Ser B Biol Sci 268(1482):2261–2265
4. Goyal P, Ferrara E (2018) Graph embedding techniques, applications, and performance: a survey. Knowl Based Syst 151:78–94
5. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) Line: large-scale information network embedding. In: Proceedings of the 24th international conference on World Wide Web, pp 1067–1077
6. Grover A, Leskovec J (2016) node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 855–864
7. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907
8. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2017) Graph attention networks. arXiv preprint arXiv:1710.10903
9. Yin Z, Shen Y (2018) On the dimensionality of word embedding. In: Proceedings of the 32nd international conference on neural information processing systems, pp 895–906
10. Gu W, Tandon A, Ahn Y-Y, Radicchi F (2021) Principled approach to the selection of the embedding dimension of networks. Nat Commun 12(1):1–10
11. Luo G, Li J, Peng H, Yang C, Sun L, Yu PS, He L (2021) Graph entropy guided node embedding dimension selection for graph neural networks. Main Track, pp 2767–2774
12. Hung PT, Yamanishi K (2021) Word2vec skip-gram dimensionality selection via sequential normalized maximum likelihood. Entropy 23(8):997
13. Wang Y (2019) Single training dimension selection for word embedding with PCA. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), pp 3597–3602
14. Krioukov D, Papadopoulos F, Kitsak M, Vahdat A, Boguná M (2010) Hyperbolic geometry of complex networks. Phys Rev E 82(3):036106
15. Yang W, Rideout D (2020) High dimensional hyperbolic geometry of complex networks. Mathematics 8(11):1861
16. Nickel M, Kiela D (2017) Poincaré embeddings for learning hierarchical representations. Adv Neural Inf Process Syst 30:6338–6347
17. Ganea O, Bécigneul G, Hofmann T (2018) Hyperbolic entailment cones for learning hierarchical embeddings. In: International conference on machine learning. PMLR, pp 1646–1655
18. Nickel M, Kiela D (2018) Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In: International conference on machine learning. PMLR, pp 3779–3788
19. Almagro P, Boguna M, Serrano M (2021) Detecting the ultra low dimensionality of real networks. arXiv preprint arXiv:2110.14507
20. Akaike H (1974) A new look at the statistical model identification. IEEE Trans Autom Control 19(6):716–723
21. Schwarz G (1978) Estimating the dimension of a model. Ann Stat 6:461–464
22. Nagano Y, Yamaguchi S, Fujita Y, Koyama M (2019) A wrapped normal distribution on hyperbolic space for gradient-based learning. In: International conference on machine learning. PMLR, pp 4693–4702
23. Rissanen J (1978) Modeling by shortest data description. Automatica 14(5):465–471
24. Rissanen J (2012) Optimal estimation of parameters. Cambridge University Press, Cambridge. https://doi.org/10.1017/CBO9780511791635
25. Yamanishi K (1992) A learning criterion for stochastic rules. Mach Learn 9(2–3):165–203
26. Yamanishi K, Wu T, Sugawara S, Okada M (2019) The decomposed normalized maximum likelihood code-length criterion for selecting hierarchical latent variable models. Data Min Knowl Discov 33(4):1017–1058
27. Fellbaum C (ed) (1998) WordNet: an electronic lexical database. Language, speech, and communication. MIT Press, Cambridge. https://doi.org/10.1017/CBO9780511791635

28. Yuki R, Ike Y, Yamanishi K (2022) Dimensionality selection of hyperbolic graph embeddings using decomposed normalized maximum likelihood code-length. In: 2022 IEEE international conference on data mining (ICDM). IEEE Computer Society, Los Alamitos, pp 666–675. https://doi.org/10.1109/ICDM54844.2022.00077
29. Fletcher PT, Lu C, Pizer SM, Joshi S (2004) Principal geodesic analysis for the study of nonlinear statistics of shape. IEEE Trans Med Imaging 23(8):995–1005
30. Pennec X (2018) Barycentric subspace analysis on manifolds. Ann Stat 46(6A):2711–2746
31. Chami I, Gu A, Nguyen DP, Ré C (2021) Horopca: hyperbolic dimensionality reduction via horospherical projections. In: International conference on machine learning. PMLR, pp 1419–1429
32. Gao Y, Yang H, Zhang P, Zhou C, Hu Y (2020) Graph neural architecture search. IJCAI 20:1403–1409
33. Ratcliffe JG, Axler S, Ribet K (1994) Foundations of hyperbolic manifolds, vol 149. Springer, New York
34. Kitsak M, Aldecoa R, Zuev K, Krioukov D (2020) Random hyperbolic graphs in $d+1$ dimensions. arXiv preprint arXiv:2010.12303
35. Barabási A-L (2013) Network science. Philos Trans R Soc A Math Phys Eng Sci 371(1987):20120375
36. Shtar'kov YM (1987) Universal sequential coding of single messages. Probl Pereda Inform 23(3):3–17
37. Buckley C, Voorhees EM (2004) Retrieval evaluation with incomplete information. In: Proceedings of the 27th annual international ACM SIGIR conference on research and development in information retrieval, pp 25–32
38. Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T (2008) Collective classification in network data. AI Mag 29(3):93–93
39. Namata G, London B, Getoor L, Huang B, Edu U (2012) Query-driven active surveying for collective classification. In: 10th International workshop on mining and learning with graphs, vol 8, p 1
40. Leskovec J, Sosič R (2016) Snap: a general-purpose network analysis and graph-mining library. ACM Trans Intell Syst Technol (TIST) 8(1):1
41. Jeong H, Mason SP, Barabasi AL, Oltvai ZN (2001) Lethality and centrality in protein networks. arXiv preprint arXiv:cond-mat/0105306
42. Sala F, De Sa C, Gu A, Ré C (2018) Representation tradeoffs for hyperbolic embeddings. In: International conference on machine learning. PMLR, pp 4460–4469
43. Chami I, Ying Z, Ré C, Leskovec J (2019) Hyperbolic graph convolutional neural networks. In: Advances in neural information processing systems, vol 32
44. Liu Q, Nickel M, Kiela D (2019) Hyperbolic graph neural networks. In: Advances in neural information processing systems, vol 32
45. Penrose M (2003) Random geometric graphs, vol 5. OUP, Oxford
46. Allen-Perkins A (2018) Random spherical graphs. Phys Rev E 98(3):032310
47. Mathai AM (1997) Jacobians of matrix transformations and functions of matrix argument. World Scientific, New York
48. Myung PDGIJ, Pitt MA (2005) Advances in minimum description length: theory and applications
49. Rissanen JJ (1996) Fisher information and stochastic complexity. IEEE Trans Inf Theory 42(1):40–47
50. Hirai S, Yamanishi K (2013) Efficient computation of normalized maximum likelihood codes for Gaussian mixture models with its applications to clustering. IEEE Trans Inf Theory 59(11):7718–7727
51. Hirai S, Yamanishi K (2017) Upper bound on normalized maximum likelihood codes for Gaussian mixture models. arXiv preprint arXiv:1709.00925
52. Vetterling WT, Teukolsky SA, Press WH (1992) Numerical recipes: example book (C), 2nd edn. Press Syndicate of the University of Cambridge, Cambridge

Ryo Yuki is currently a Ph.D. student under the supervision of Prof. Kenji Yamanishi at the Graduate School of Information Science and Technology, the University of Tokyo. His research interests include graph embedding and statistical model selection.

Yuichi Ike received his Ph.D. in mathematical science at the University of Tokyo in 2018. He started his career as a researcher at Fujitsu Laboratories Ltd. In 2021, he moved to Graduate School of Information Science and Technology, The University of Tokyo. He is currently an associate professor at Institute of Mathematics for Industry, Kyushu University. His main research interests are topological data analysis, topological/geometrical methods in machine learning, microlocal sheaf theory, and symplectic geometry.

Kenji Yamanishi received the master degree from the University of Tokyo in 1987. He received the Dr. Eng. degree from the University of Tokyo in 1992. He joined NEC Corporation in 1987, and his final position in 2008 was a fellow in the Internet Systems Research Laboratories and a department head of Data Mining Technology Center of NEC Corporation. He worked for the NEC Research Institute in the USA as a visiting scientist from 1992 to 1995. He joined the University of Tokyo in 2009 and has been working as a professor of the Graduate School of Information Science and Technology of the University of Tokyo. His research interests include information-theoretic machine learning, data science and big data analysis. Ryo Yuki is currently a Ph.D. student under the supervision of Prof. Kenji Yamanishi at the Graduate School of Information Science and Technology, the University of Tokyo. His research interests include graph embedding and statistical model selection.