Cyprus University of Technology

HANGZHOU DIANZI UNIVERSITY

Master's thesis

Comparison of hyperbolic embedding methods for real-world networks: Machine Learning versus Network Science

ZHOU HAOJIE

Limassol, May 2025

MSc in Electronics Science and Technology

CYPRUS UNIVERSITY OF TECHNOLOGY

Faculty of Engineering and Technology

Department of Electrical Engineering, Computer Engineering, and Informatics

Master's thesis

Comparison of hyperbolic embedding methods for real-world networks: Machine Learning versus Network Science

ZHOU HAOJIE

Supervisor

Fragkiskos Papadopoulos

Limassol, May 2025

Approval Form

Master's thesis

Comparison of hyperbolic embedding methods for real-world networks: Machine Learning versus Network Science

Presented by

ZHOU HAOJIE

Supervisor: Fragkiskos Papadopoulos

Member of the committee: Michael Sirivianos

Member of the committee: Panagiotis Ilia

Cyprus University of Technology
Limassol, May 2025

Copyrights

The approval of the dissertation by the Department of Electrical Engineering, Computer Engineering, and Informaticsdoes not necessarily imply the approval by the Department of the views of the writer.

Acknowledgements

I would like to express my heartfelt gratitude to Professor Fragkiskos Papadopoulos for his meticulous guidance throughout the preparation of this thesis. He is an exceptionally responsible mentor who always offered timely and patient support whenever I encountered difficulties in my studies and research. Moreover, Professor Papadopoulos is remarkably approachable, making every conversation with him feel relaxed and pleasant, like chatting with a friend. It has been a fulfilling and meaningful year working under his supervision, and I am truly grateful for his invaluable support.

During my year in Cyprus, I had the wonderful opportunity to meet people from diverse backgrounds and experience the beauty of this remarkable place. I fell in love with the slow-paced lifestyle, the breathtaking scenery, and, most of all, the friendliness of the people. I am especially thankful to Professor Stelios, his wife Giselle, and their adorable son George. Their care and support brought great warmth to my life, making me feel at home even when I was far away from my own country. I would also like to sincerely thank the Cyprus University of Technology for providing excellent academic and technical support. It is truly an outstanding institution, and I feel honored to have been a part of it. This year in Cyprus has been an incredibly enriching experience, one that I will treasure for the rest of my life.

ABSTRACT

As a core topic in the field of artificial intelligence, learning has always been one of the key steps towards achieving strong artificial intelligence. It aims to transform complex data structures into concise numerical representations, so that machine learning models can better discover patterns and rules from them. Graph data is a data structure used to represent complex relationships between entities, consisting of nodes and edges, where nodes represent entities or objects and edges represent relationships or connections between nodes. Graph data has been widely applied in various fields [1] [2], such as social networks, transportation networks, knowledge graphs, etc. Many machine learning or deep learning methods typically attempt to map graph data into a low dimensional vector space [3] for easier processing and analysis. This mapping process is commonly referred to as graph embedding or graph representation learning, aimed at effectively preserving the relationships and attribute information between nodes in the original graph data. Through graph representation learning, complex tasks can be analyzed and processed more efficiently. For example, graph representation learning is widely used in recommendation systems to handle complex user item interactions and improve recommendation quality [4].

The hyperbolic embedding of networks is an effective means of processing graph data information, which has advantages in expressing hierarchy [5] and capturing complex relationships in graph data. With the rapid development of network science and advances in mathematical theory, especially the negative curvature property of hyperbolic geometry, an ideal framework has been provided for simulating the infinite expansion and hierarchical structure of networks. Combining the development of deep learning and machine learning techniques, hyperbolic geometry embedding has become an effective means of solving precise network representations in practical applications such as social network analysis and recommendation systems.

This paper conducts a series of studies on comparing three main popular hyperbolic embedding methods. It primarily considers machine learning methods, specifically the Poincare method and the Lorentz method, and model-based methods, primarily the D-Mercator method. Model-based methods from the network science community have advantages in interpretability and accurately capturing the hierarchical structure and clustering of networks, thereby maintaining an accurate description of the global network topology. In contrast, machine learning models better adapt to local structural patterns when data is sufficient. The choice of method often depends on the specific application scenario and research objectives. Therefore, the contribution of this thesis lies in quantitatively comparing the embedding quality of real-world networks in hyperbolic space, obtained through three main hyperbolic embedding methods, across different research communities (e.g., computer scientists vs. statistical physicists) and elucidating their performance in two key applications of interest.Specifically:

1. We consider and compare three hyperbolic embedding methods—the Poincare method, the Lorentz method, and the D-Mercator method, applying them to the topology of the Autonomous Systems Internet (IPv6) [6]. We calculate the rank correlation coefficient and Pearson coefficient between between the hyperbolic node distances obtained in the different embeddings. With the Pearson correlation coefficient we measure the linear correlation between two sets of hyperbolic distances, while the rank correlation coefficient is a method for measuring the degree of similarity between the hyperbolic distance rankings of the same pair of nodes across embeddings, which captures the

correlation among the hyperbolic distance orderings between embeddings.

2. We also study the performance of the embeddings obtained by the different methods in two main applications of interest: (i) greedy routing [7] [8], where we evaluate both the routing success rate and the average number of hops; and (ii) link prediction [9], conducted under different link removal rates and evaluated using standard metrics including ROC/AUC and PR/AUPR.

Keywords: Hyperbolic Geometry, Greedy Routing, Network Embedding, Graph Data

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ROC	Receiver Operating Characteristic
TPR	True Positive Rate
FPR	False Positive Rate
TR	True Positives
FN	False Negatives
AUC	Area Under the ROC Curve
PR	Precision-Recall
AUPR	Area Under the Precision-Recall
AS	Autonomous System

1 Introduction

1.1 Aims and Objectives

As a core topic in the field of artificial intelligence, learning has always been one of the key steps towards achieving strong artificial intelligence. It aims to transform complex data structures into concise numerical representations, so that machine learning models can better discover patterns and rules from them.Graph data is a prevalent data structure in real-world scenarios, widely utilized across various domains and applications due to its ability to intuitively capture diverse relationships and interactions. Khoshraftar et al. (2022) [10] provided a comprehensive survey on graph representation learning methods, highlighting the significance of graph data across various application domains. For instance, in recommendation systems, graph data represents interactions between users and items, such as clicks, purchases, and ratings, as well as relationships between users and items or among items themselves [11]. Knowledge graphs, as a specialized form of graph data, further enrich this representation by encapsulating semantic relationships between entities. Effectively modeling, analyzing, and uncovering the latent features of graph data is a central challenge in the field of graph data mining. To address this, graph representation learning aims to encode nodes or edges within graph data into low-dimensional vectors that preserve the original structure and attributes, facilitating downstream tasks. By leveraging these vector representations, recommendation systems can extract meaningful structural and relational information from user-item interaction graphs, generate latent representations reflecting user preferences and item attributes, and provide personalized recommendations. Moreover, integrating knowledge graphs into recommendation systems enhances recommendations by incorporating supplementary semantic relationships, thereby uncovering hidden user interests and preferences. Guo et al. (2020) [4] provided a detailed survey on knowledge graph-based recommender systems, highlighting how semantic relationships in knowledge graphs can be leveraged to improve recommendation performance.

Although significant progress has been made in this field, most existing representation learning methods are still limited to using Euclidean space as the embedding carrier. This method is inadequate when dealing with complex network data with hierarchical structures [12]. For example, real-world networks such as social networks, biological networks, and knowledge graphs often present a hierarchical tree-like structure [13], which Euclidean space cannot effectively capture. The development of hyperbolic network embeddings is based on the need for a deeper understanding of the structural characteristics of such complex networks, as well as the recognition of the limitations of traditional Euclidean space embedding methods in expressing hierarchy and capturing complex relationships. With the rapid development of network science and advances in mathematical theory, the negative curvature property of hyperbolic geometry has provided an ideal framework for simulating the infinite expansion and hierarchical structure of networks [5] [14] [15]. With advances in hyperbolic geometry and network modeling, hyperbolic embedding has become an effective technique for capturing hierarchical structures in real-world networks, and has been successfully applied in tasks such as social network analysis and recommendation systems.

Hyperbolic geometry is a non Euclidean geometry that describes spaces with negative curvature, where parallel lines can be infinitely many and the sum of interior angles of triangles is less than 180 degrees.

The definition of hyperbolic space is a type of space with a negative constant curvature. In hyperbolic space, as the radius increases, the circumference and area of a circle grow much faster than in Euclidean space. For example, in hyperbolic space, the ratio of the circumference of a circle increases exponentially with its radius. The characteristics of this space make hyperbolic geometry very suitable for characterizing complex networks [5] with hierarchical and scale free properties. These characteristics of hyperbolic geometry provide a theoretical basis for hyperbolic embedding, enabling it to reveal the inherent hierarchical structure and topological properties of networks. For example, the exponential dilation property in hyperbolic space can be used to more naturally represent networks with hierarchical structures. In addition, hyperbolic embedding has demonstrated excellent performance in multiple fields such as recommendation systems, knowledge graphs [14], natural language processing [16], computer vision, and bioinformatics.

In summary, there are many advantages to using hyperbolic space embedding methods to process graph data: (1) Natural representation of hierarchical structures [5]: Graph data often contains hierarchical or hierarchical structures, such as community structures in social networks or classification levels in biological networks. Hyperbolic space can naturally represent these hierarchical structures, allowing the hierarchical relationships in graph data to be intuitively reflected in embedding vectors. (2) Improving embedding efficiency [14] [15]: The exponential growth property of hyperbolic space allows for more efficient utilization of dimensions when embedding graph data. This means that the complex structure of graph data can be represented with fewer dimensions, thereby improving embedding efficiency and reducing storage and computational costs. (3) Enhanced classification and clustering performance [17]: In hyperbolic space, similar nodes (such as those with similar features or behaviors) will be closer to each other, while dissimilar nodes will be farther apart. This geometric attribute helps improve the accuracy of node classification and clustering in graph data, such as identifying user preferences in recommendation systems or discovering communities in social networks. (4) Optimized routing and link prediction [7]: In network design and analysis, such as Internet routing or biological information networks, hyperbolic embedding can help optimize routing strategies and improve the accuracy of link prediction, by better understanding the geometric relationship between nodes. [18] In this thesis we will conduct a study on the embedding quality of graph data with hierarchical structure after embedding into hyperbolic space using different hyperbolic embedding methods. The purpose is to compare the embedding results of three major methods on the same dataset, explore the relationship between them, and attempt to quantify the quality of embedding through experiments such as greedy routing and missing link prediction.

1.2 Current State of Research

There are currently many methods for embedding real networks into hyperbolic space, which are mainly divided into two categories: machine learning [14] [15] methods and model-based methods [19]. The core idea of hyperbolic embedding method is to map nodes in the network to hyperbolic space, where the distance between nodes reflects their relationship in the original network. In machine learning, these mappings are achieved through optimization algorithms with the goal of minimizing the distance in the embedding space and the relationships in the original network.Machine learning-based hyperbolic embedding methods typically do not rely on complex optimization processes because they leverage gradient descent algorithms that iteratively adjust the embedding positions by computing gradients of the loss

function with respect to the hyperbolic coordinates [20]. This approach simplifies the optimization by breaking down the problem into a series of local updates, where at each step the algorithm moves the embeddings in the direction that most reduces the discrepancy between the hyperbolic distances and the original network relationships. The efficiency of gradient descent stems from its ability to handle high-dimensional parameter spaces while avoiding the need for computationally expensive global optimization procedures [21].

Gradient descent is an optimization algorithm used to minimize the objective function, typically used in machine learning and artificial intelligence to train models. It iteratively adjusts parameters and searches along the opposite direction of the objective function gradient (i.e. the steepest descent direction) to find the local minimum of the function. The key to gradient descent is to calculate the gradient of the objective function with respect to the parameters, which points towards the direction where the function grows the fastest. In each iteration, the algorithm updates the parameters to move along the negative direction of the gradient, with the step size determined by the learning rate. Gradient descent can be applied to various models and loss functions, including linear regression, neural networks, etc., and is one of the most common parameter optimization methods in machine learning. Its main advantages are simplicity, intuitiveness, and the ability to easily parallelize large-scale data sets.

Next, we will overview two methods of embedding real networks into hyperbolic space using machine learning:

1. Poincare disk model

The Poincare disk model is a conformal representation of hyperbolic space that captures the exponential growth characteristics of hyperbolic space through specific distance measurement formulas. In this model, the points are located within the unit disk, while the line (geodesic) is the vertical arc on the disk boundary or the diameter within the disk. This model has a constant negative curvature of -1, which allows it to intuitively represent the hierarchical and scale free properties in hyperbolic space. The Poincare disk model is particularly suitable for representing networks with hierarchical and tree like structures, as it can naturally capture the exponential growth characteristics of these structures. These characteristics make the Poincare disk model a powerful tool for understanding and analyzing complex network structures.

In Nickel et al.'s study [14], the Poincare disk model was used to learn hierarchical representations from symbolic data. This method is particularly suitable for datasets with potential hierarchical structures, such as text and graph structured data. In order to learn these embeddings, the author proposes an effective algorithm based on Riemann optimization [22]. This algorithm optimizes embedding by minimizing a specific loss function that encourages semantically similar objects to approach each other in the embedding space. Through experiments, the author has demonstrated that Poincare embeddings are significantly superior to traditional Euclidean embeddings in terms of representation and generalization ability on data with potential hierarchy.

The authors first evaluated the performance of Poincare embedding on the WORDNET dataset, which contains a large number of nouns and their superclass relationships. The experimental results show that Poincare embedding can effectively reconstruct data and perform well in link prediction tasks. In addition, the authors conducted link prediction experiments on social network datasets by removing some links from the dataset and embedding using the Poincare disk model again. The likelihood of link existence is

reflected by the distance between links. Experimental results show that although some links are removed, the quality of the embedded network is still good, proving the effectiveness of Poincare embedding in predicting network links, especially in low dimensional spaces.

Another important application of Poincare embedding is in lexical implication tasks. The application of Poincare embedding method in lexical implication tasks demonstrates its unique advantages in handling hierarchical relationships. Vocabulary implication is a key issue in the field of natural language processing, which involves determining the semantic relationship between two terms, such as whether one term is conceptually a subset of the other term. This relationship often manifests as a hierarchical structure, for example, 'cat' is an 'animal'. In the paper, the author evaluates the "is" relationship between noun pairs using the Poincare embedding method. This method utilizes the geometric properties of hyperbolic space to capture the hierarchical relationships between vocabulary. In hyperbolic space, each word is mapped to a point, and the similarity or implication relationship between words is measured by the distance between points. Through this approach, Poincare embeddings can not only capture direct relationships between vocabulary, but also reflect deeper semantic hierarchies. The application of Poincare embedding method in lexical implication tasks not only demonstrates its ability to handle hierarchical relationships, but also demonstrates its potential in capturing and predicting complex network structures.

In summary, Poincare embedding provides a powerful new tool for processing symbolic data with hierarchical structures. It can more naturally represent the hierarchical structure of data by embedding it into hyperbolic space, while improving the quality and generalization ability of embedding through optimization algorithms. This method has demonstrated excellent performance in multiple tasks, providing new directions for future research.

2. Lorentz model

The Lorentz model is a type of model in hyperbolic geometry, proposed by German mathematician Hermann Minkowski. It utilizes the concept of Lorentz transformation, a mathematical tool in special relativity that describes the relativity of time and space. In this model, points in hyperbolic space are composed of N-dimensional coordinates and additional time or correlation coordinates, forming an N+1-dimensional vector. The key advantage of this representation lies in its distance measurement, where the Lorentz distance inherently combines spatial and temporal components through the Lorentz inner product. This unified processing method achieves the constraint of points to the hyperbolic surface in class coordinates, avoiding the numerical instability of the Poincare sphere model near the boundary. The geodesic distance (shortest path) can be calculated in a closed form without the need for iterative approximation. The conformal property of this model maintains angle invariance, making it very useful when dealing with complex hierarchical data. The Lorentz model is widely used in physics and mathematics, especially in capturing complex network structures and processing machine learning applications such as hierarchical text and graph data.

In Kiela et al.'s study [15], a hypersphere embedding method based on the Lorentz model was proposed to discover hierarchical relationships from large-scale unstructured similarity data. This method is particularly suitable for discovering hierarchical relationships between concepts from pairwise similarity scores, such as in biological classification, social network analysis, language evolutionary trees [23], and other scenarios. The advantage of the Lorentz model lies in its ability to efficiently perform Riemann optimization, which makes it possible to directly calculate geodesics on hyperspheres, thereby avoiding

numerical instability that occurs in the Poincare sphere model. In addition, the Lorenz model can restore the correct hierarchical relationships from embeddings while maintaining the order of similarity.

In order to learn high-quality embeddings, the authors propose a new optimization method based on the Lorentz model. This method optimizes by directly following the geodesic on the hypersphere, instead of only performing first-order approximations as in the Poincare model [14]. The experimental results show that this method is particularly effective in low dimensional spaces and can provide higher quality embeddings than Poincare embeddings. To verify this result, two models were applied on two real-world datasets: one on the organizational structure of companies, and the other on the historical relationships of language families. Embedding organizational structure data can reveal the hierarchical structure within the company; Embedding on language family data can discover historical connections between languages. These experimental results demonstrate the effectiveness of the Lorentz model in discovering meaningful hierarchical structures.

Overall, the proposed hyper spherical embedding method based on the Lorentz model provides a powerful tool for discovering hierarchical relationships from unstructured data. The main contributions of this article [15] are threefold: firstly, a new hypersphere embedding model is proposed, which is based on the Lorentz model and can effectively learn hierarchical representations; Secondly, an efficient Riemann optimization method was proposed, which can directly optimize in high-dimensional space and avoid numerical instability; Finally, experimental results on two real-world datasets demonstrate that this method can effectively discover hierarchical relationships from pairwise similarity scores. This method can restore the correct hierarchical relationships from embeddings while maintaining similarity order, which is of great significance for understanding the structure and evolution of complex systems.

Next, overview a type of hyperbolic embedding method originating from the network science community and preceding the work of Kiela et al. Papadopoulos et al. proposed a network mapping method called HyperMap [24], which can embed complex real-world networks into hyperbolic space. HyperMap is based on a rigorous geometric theory of complex networks, which models complex networks as random geometric graphs in hyperbolic space. This method estimates the hyperbolic coordinates of new nodes in a growing network by replaying the geometric growth process of the network, thereby maximizing the likelihood of network snapshots in the model. This method utilizes the Popularity Similarity Optimization (PSO) model, which is a model that describes the growth of complex networks and can reproduce various structures and dynamic characteristics of real networks. The PSO model assumes that the probability of a new node in the network connecting to other nodes based on its position in hyperbolic space is a decreasing function of hyperbolic distance. The HyperMap method utilizes this theory to infer the order of node appearance through maximum likelihood estimation (MLE) [25], and based on this, replays network growth to find hyperbolic coordinates that maximize local likelihood for each new node.

HyperMap has been applied to the autonomous system (AS) topology of the Internet, and found that this method can produce meaningful results, such as identifying soft communities of ASs belonging [5] in the same geographical region. In addition, the network map obtained using HyperMap can predict missing links in the Internet with high accuracy, surpassing popular existing methods. The network map constructed by HyperMap also exhibits high navigability, which means that most greedy geometric routing paths are successful and have low stretch. HyperMap also has outstanding performance in predicting missing links. Compared to several classic link prediction techniques such as common neighbors, de-

gree product, inverse shortest path, Katz index, and hierarchical random graph model, HyperMap shows stronger predictive ability, especially in predicting ”difficult to predict” links (links between low degree nodes without common neighbors).

However, the HyperMap method still has significant limitations and room for improvement, such as accurate estimation of the angular coordinates of early nodes and improvement in the process of maximizing the likelihood function. Nevertheless, HyperMap performs well in embedding accuracy and computational efficiency, providing a powerful tool for understanding network structure and dynamics. This work not only advances our understanding of network geometry, but also provides new perspectives and methods for the field of network science.

Robert Jankowski et al. [19]developed another method which combines MLE like HyperMap with Laplacian Eigenmaps, called D-Mercator for embedding complex real-world networks into multidimensional hyperbolic spaces. The D-Mercator method is model-based and can map the network to a (D+1) - dimensional hyperbolic space, where the similarity subspace is represented as a D-dimensional sphere. The proposal of this method is to solve the problem of better describing real-world networks through multi-dimensional latent geometric models.

The core of the D-Mercator method lies in utilizing the properties of hyperbolic geometry, which are particularly useful in describing networks with hierarchical structures. In hyperbolic space, the probability of connections between nodes follows a form similar to the law of gravity, where the ”hidden degree” of nodes and their position on the D-dimensional sphere determine the likelihood of their interactions. This method can not only capture the local connectivity patterns of the network, but also reveal the global structural features of the network, such as community structure and network navigability.

The D-Mercator method provides a new perspective for studying the navigability of networks by embedding them into multidimensional hyperbolic spaces. In this approach, the navigability of the network can be evaluated by applying Greedy Routing in hyperbolic space. Greedy routing is a simple routing strategy that selects the neighbor closest to the target node as the next routing point at each step. The success or failure of this method, as well as the efficiency of the routing path (such as path length and the proportion of successfully reached targets), are indicators of network navigability and of the embedding quality (the higher the performance of greedy routing, the more congruent is the embedding with the network topology). In the D-Mercator method, the navigability of the network depends not only on the local connectivity between nodes, but also on the global structure of the network in hyperbolic space. Therefore, by optimizing the embedding of the network in hyperbolic space, the navigability of the network can be improved, thereby enhancing the information transmission efficiency of the network while maintaining its structural characteristics. This method provides a new tool for understanding and improving the navigability of complex networks, especially in network applications that require efficient path planning and information dissemination.

Researchers have validated the accuracy of the D-Mercator method in embedding quality and model parameter inference through testing on synthetic networks. They found that the embeddings obtained using the D-Mercator method can highly accurately restore the original coordinates of the network and correctly determine all other model parameters, including hidden degree and inverse temperature parameters. In addition, D-Mercator is able to identify the dimensions used to generate synthetic networks without prior knowledge of that dimension.

In the application to real-world networks, the D-Mercator method demonstrates its advantages in multidimensional representation. For example, in AddHealt 's [26] research on networks, this method can clearly distinguish student groups of different grades in a two-dimensional hyperbolic space, which cannot be achieved in a one-dimensional representation. The D-Mercator method has also demonstrated its effectiveness in community detection and greedy routing tasks in other networks, such as the International Apple Trade Network (FAO apples) and the Global Flight Network (OpenFlights).

The proposal of the D-Mercator method provides a new tool for the field of network science, which can not only improve our understanding of network structure, but also play an important role in machine learning, community detection, and network dynamics research. By embedding networks in appropriate dimensions, researchers can more accurately capture the complexity and diversity of networks, providing new perspectives and methods for the study of complex systems.

By comparing hyperbolic embedding methods from the machine learning and network science communities, we can gain a deeper understanding of their characteristics. In the field of machine learning, hyperbolic embedding methods are commonly used for data visualization, clustering, and dimensionality reduction, while the network science communities focus more on capturing the geometric properties and topological structure of the network. The core idea of hyperbolic embedding is to map nodes in the network to hyperbolic space, where the distance between nodes reflects their relationship in the original network. In machine learning, these mappings are achieved through optimization algorithms with the goal of minimizing the distance in the embedding space and the relationships in the original network. In the network science community, hyperbolic embedding relies more on geometric models of the network, which attempt to capture its dynamic and structural characteristics. In machine learning, hyperbolic embedding methods may not rely on complex optimization processes, which may include gradient descent and automatic encoding of neural network structures. In the network science community, hyperbolic embedding methods may focus more on the accuracy of the model and the consistency of the physical model of the network. For example, hyperbolic embedding methods in the network science community may be based on geometric models of the network that attempt to capture the growth and evolution process of the network.

The latest method from network science for hyperbolic network embedding is a novel method called FiD Mercator [27] published by Jankowski et al., which is a model-based ultra-low dimensional dimensionality reduction technique used to embed complex networks into hyperbolic space. FiD Mercator created a D-dimensional map that describes the network by integrating node features (i.e. node attributes or quality descriptions) with the network structure. This method effectively utilizes features as initial conditions to guide node coordinate search towards the optimal solution. FiD Mercator is an extension of D-Mercator that not only considers network structure but also node characteristics. FiD Mercator improves the accuracy and robustness of embedding by integrating node features during the embedding process, especially when the features are highly correlated with the network structure. By applying FiD Mercator on multiple real-world network datasets, including citation networks, social networks, product networks, and web networks. The comparison of experimental results with D-Mercator's results shows that FiD Mercator performs better in link prediction and node classification tasks than the D-Mercator method that relies solely on network structure. This indicates that when node features are highly correlated with network topology, integrating feature information into network embedding can significantly improve embedding

quality.

By combining network topology and node characteristics, FiD Mercator can provide richer and more accurate descriptions for complex networks, and improve performance in various downstream tasks. The development of this method provides new tools for the field of network science, which helps to better understand and predict the behavior of complex networks. Future research may explore how to intelligently select feature sets and integrate non binary features into analysis to further optimize the performance of network embedding and related tasks.

1.3 Contribution

This paper conducts a series of studies on comparing three main popular hyperbolic embedding methods. It primarily considers machine learning methods, specifically the Poincare method and the Lorentz method, and model-based methods, primarily the D-Mercator method. Model-based methods from the network science community have advantages in interpretability and accurately capturing the hierarchical structure and clustering of networks, thereby maintaining an accurate description of the global network topology. In contrast, machine learning methods exhibit better local adaptability when data is abundant and computational resources are sufficient. The choice of method often depends on the specific application scenario and research objectives. Therefore, the contribution of this thesis lies in quantitatively comparing the embedding quality of real-world networks in hyperbolic space, obtained through three main hyperbolic embedding methods, across different research communities (e.g., computer scientists vs. statistical physicists) and elucidating their performance in two key applications of interest.Specifically:

1. We consider and compare three hyperbolic embedding methods—the Poincare method, the Lorentz method, and the D-Mercator method, applying them to the topology of the Autonomous Systems Internet (IPv6) [6]. The comparison allows us to understand the strengths and limitations of each method in preserving the intrinsic structure and relationships of the graph. We calculate the rank correlation coefficient and Pearson coefficient between between the hyperbolic node distances obtained in the different embeddings. With the Pearson correlation coefficient we measure the linear correlation between two sets of hyperbolic distances, while the rank correlation coefficient is a method for measuring the degree of similarity between the hyperbolic distance rankings of the same pair of nodes across embeddings, which captures the correlation among the hyperbolic distance orderings between embeddings.

2. This article not only investigates the linear or nonlinear relationship between the results of embedding different types of hyperbolic embedding methods into the same dataset, but also quantitatively studies the quality of embedding results. This article considers methods such as greedy routing and missing link prediction. For greedy routing, we randomly set the starting node and target node, and each time select the neighboring node with the shortest hyperbolic distance from the target node among the current node's neighboring nodes as the next node, and check whether it can ultimately reach the target node. We can quantify embedding quality using success rate and average hop count; For missing link prediction, Randomly remove some links while maintaining network connectivity, and then re embed the remaining links into hyperbolic space, and the hyperbolic distance between nodes is calculated pairwise and sorted from small to large. The hyperbolic embedding quality

can be quantified by the negative correlation between these hyperbolic distances and the probability of missing links. These quantitative indicators provide a method to evaluate and compare the performance of hyperbolic embedding methods in practical network routing tasks, thereby helping researchers and practitioners better understand and utilize graph data embedding in hyperbolic space.

1.4 Structure of the Thesis

This thesis compares hyperbolic embedding methods for real-world networks and consists of five chapters. The specific organization and arrangement of each chapter is as follows:

The first chapter is an introduction. Firstly, the research background and significance of this topic are introduced; Secondly, the research status of hyperbolic embedding methods in machine learning and network science communities are introduced; Subsequently, the main contributions of this paper are introduced; Then, the organizational structure of this paper is introduced; Finally, a brief summary is made for the entire introduction section.

The second chapter is related theoretical knowledge. Firstly, the basic concepts of graph data are introduced; Secondly, geometric related concepts are introduced; Then Euclidean geometry and hyperbolic geometry are introduced, along with two isometric models of hyperbolic geometry, including the Lorentz model and the Poincare sphere model. Subsequently, the hyperbolic embedding algorithm model from network science is introduced.

The third Chapter is about research methods, mainly proposing corresponding solutions to the problems we are studying. Firstly, we introduced the AS (Autonomous Systems) dataset used, and then embedded the same dataset using machine learning methods (Poincare method and Lorentz method) and network science methods (D-Mercator method). We calculate the Pearson coefficient and the rank correlation coefficient (e.g., Spearman's ρ) of the obtained results to assess whether there is a linear or monotonic nonlinear relationship between the results of different methods. Finally, we quantify the embedding quality of the two methods by calculating the success rates of greedy routing and missing link prediction.

The fourth Chapter is Results and Discussion, mainly carrying on the calculations and experiments mentioned in Chapter 3. We compare the relationship and embedding quality of the three methods through quantitative results. Determine whether there are linear or nonlinear relationships between the results obtained by processing the same data using three different methods, based on the Pearson coefficient and the rank correlation coefficient; Quantify the embedding quality of three methods by comparing the results of greedy routing and missing link prediction. Discussing the advantages and disadvantages of three methods based on the results can better help people.

The fifth chapter is a summary and outlook. Firstly, a summary of the research background and content of this article has been made. Secondly, based on the current research status, future directions worth further investigation were explored.

1.5 Summary

This chapter first introduces the advantages of hyperbolic embedding based on the limitations of current Euclidean space representation data, and then presents the objectives and work of this paper. Then, by introducing the current status of hyperbolic embedding technology and comparing the advantages and disadvantages of various technologies, we have gained a preliminary understanding of hyperbolic embedding technology. Furthermore, the main contributions of this thesis and the general content of each part of the overall article were introduced, followed by a summary.

2 Related theoretical knowledge

2.1 Graph Data

Graph data is a data structure composed of nodes and edges between nodes, which can reflect complex data in the real world. As early as 2009, Papadopoulos et al. [28]demonstrated in their paper 'Curvature and Temperature of Complex Networks' that real-world graph data often possess latent hierarchical properties [29] [30] [31] or unobservable underlying structures, which significantly influence the interpretation and analysis of such data. Therefore, understanding and utilizing the hierarchical structure in graph data for hierarchical modeling can help to more comprehensively grasp the intrinsic structure and relationships of graph data, thereby improving the accuracy and interpretability of the model. For example, social networks, air transportation networks, and so on. Papadopoulos et al. (2012) [32]proposed a hyperbolic geometry-based network generation model that integrates two key factors—node popularity and similarity—to generate graph structures with realistic network properties. In the diagram, nodes represent entities in real life; If the graph represents a social network, then nodes represent individual users. Edge represents the relationship between every two entities; For social networks, the edges between nodes represent the existence of communication or shared interests between them. Graphs can be divided into two categories: isomorphic graphs and heterogeneous graphs. Isomorphic graphs only contain a single type of node and edge, while heterogeneous graphs contain multiple types of nodes and edges, which enables heterogeneous graphs to more accurately map complex network structures in the real world. Graph data has a wide range of applications in various fields, including recommendation systems, traffic flow prediction, and community detection.

A graph G=(V, E), composed of a set of nodes V and an edge set E, where $(u, v) \in E$ represents the connection between nodes u and v, reflecting their specific relationship. The graph can be directed or undirected, depending on whether the edges have directionality.

The relationships between nodes in a graph are usually represented by adjacency matrices. For a graph with n nodes, its adjacency matrix A is an n × n matrix. If there is an edge between node i and node j, then Aij=1; If there are no edges, Aij=0. This representation method allows us to efficiently process and analyze graph data in matrix form.

Node features are vectors or matrices that encode or represent nodes in a graph. In practical applications, graph nodes usually have a variety of features. For example, in social networks, node features can include a user's friend list, interests, hobbies, and so on. By utilizing machine learning, deep learning [33] [34], and other methods, node features can be further processed to extract richer node features, thereby improving model performance.

In this thesis, we use AS graph data. AS (Autonomous System) refers to a collection of one or more networks that are logically managed and exhibit a consistent routing strategy externally. Each AS is identified by a unique number (ASN, Autonomous System Number), which is used to identify and make routing decisions in the Internet. Autonomous systems are widely used in operator networks, enterprise networks and data centers to achieve hierarchical and efficient management of routes in the Internet. They exchange routing information with other ASs through protocols such as BGP (Border Gateway Protocol),

thus ensuring the interconnection of the global Internet [35].

2.2 Geometrical Concept

In order to better understand hyperbolic space, this section will introduce some major geometric concepts. For more detailed knowledge of geometric theory, please refer to relevant standard textbooks. [36] [37]

1. Manifolds: Manifolds are geometric objects that are locally similar to Euclidean spaces. In a manifold M of dimension d, each point's neighborhood can be approximated as a local approximation of d-dimensional Euclidean space $\mathbb{R}^d$. For example, the Earth can be modeled as a spherical object, and its local regions can be approximated using $\mathbb{R}^2$.

2.Tangent Space: In a Riemannian manifold M, the tangent space T_uM at a point u is a d-dimensional vector space that provides a first-order approximation of the manifold M at u. It is the local linearization of the manifold at point u, capturing the linear behavior of the manifold in the vicinity of that point.

3.Riemannian metric: The Riemannian metric is a core concept in Riemannian geometry, providing an intrinsic measure for each point on a manifold, allowing us to measure lengths and angles in the vicinity of that point. Specifically, at any point u on a manifold M, the Riemannian metric g defines the inner product on the tangent space T_uM, which permits us to compute the length of vectors and the angle between two vectors.

4.Riemannian Manifold: Riemannian Manifold, also known as a Riemannian metric space, is a fundamental concept in Riemannian geometry. It is a smooth manifold equipped with a Riemannian metric, which allows us to define geometric concepts such as distance, angles, and volume on the manifold. Formally, a d-dimensional smooth manifold M equipped with a Riemannian metric tensor g is defined as a Riemannian manifold, denoted as (M, g). [38]

5.Geodesics: In Riemannian geometry, geodesics are the shortest paths connecting two points, generalizing the concept of a straight line in Euclidean space.

6.Exponential Map: The exponential map is a method of mapping from the tangent space at a point on a Riemannian manifold to the manifold itself, denoted as $\exp_u : T_uM \to M$. On a Riemannian manifold M, for any vector t in the tangent space T_uM at point $u \in M$, the exponential map $\exp_u(t)$ generates a point on the shortest geodesic starting from u and traveling in the direction and length specified by the vector t. The definition of the exponential map is particularly useful in gradient descent and optimization algorithms, as it provides a way to perform parameter updates along geodesics, naturally preserving the intrinsic geometric structure of the data.

7.logarithmic map: The logarithmic map can be considered as the inverse process of the exponential map, that is $\log_u : M \to T_uM$. For two points u and z on the Riemannian manifold M, if there exists a unique shortest geodesic between them, then the logarithmic map $\log_u(z)$ projects the point z on the manifold M back to a vector in the tangent space T_uM at point u. The direction and magnitude of this vector represent the direction and distance from z along the shortest geodesic to u.

2.3 Euclidean Space

Euclidean geometry, a geometry with zero sectional curvature, serves as the cornerstone of many mathematical theories and has significantly shaped our understanding of the spatial relationships. It is based on five fundamental postulates that define the behavior of points, lines, and planes:

1. Through any two distinct points, there is exactly one straight line.
2. A line segment can be extended in both directions to form a line that is infinitely long.
3. Given any point not on a given line, there is exactly one line through the point that does not intersect the given line; this line is unique and is called a parallel to the given line.
4. All right angles are congruent (equal in measure).
5. For any triangle, the sum of the interior angles is exactly 180 degrees.
6. If two lines are each intersect a third line, and the sum of the interior angles on the same side of the third line is less than two right angles, then the two lines, if extended indefinitely, will meet on that side of the third line; this is the basis for the concept of parallelism.

These postulates form the basis for Euclidean geometry and are essential for understanding the properties of geometric figures and for solving geometric problems.

In d-dimensional Euclidean space, to calculate the distance between two points as well as the angle between vectors or lines, given two vectors $u, v \in \mathbb{R}^d$, their standard inner product is defined as:

$$\langle u, v \rangle = \sum_{i=1}^{d} u_i v_i \tag{2.1}$$

By utilizing the standard inner product, the length of vector u can be obtained as:

$$\|u\| = \sqrt{\langle u, u \rangle} = \sqrt{\sum_{i=1}^{d} (u_i)^2} \tag{2.2}$$

The length function of vector u satisfies the properties of a norm, and thus is also referred to as the Euclidean norm on Euclidean space. The Euclidean distance can be defined using the Euclidean norm as:

$$d(u, v) = \|u - v\| = \sqrt{\sum_{i=1}^{d} (u_i - v_i)^2} \tag{2.3}$$

Given two points $u, v \in \mathbb{R}^n$ in Euclidean space and a vector $t \in T_u\mathbb{R}^n$ in the tangent space, the exponential map and logarithmic map of Euclidean space are defined as:

$$\exp_u(t) = u + t \tag{2.4}$$

$$\log_u(v) = v - u \tag{2.5}$$

2.4 Hyperbolic Geometry

Hyperbolic Geometry, also known as Lobachevskian Geometry, is a type of non-Euclidean geometry independently discovered by the Russian mathematician Nikolai Ivanovich Lobachevsky and the Hungarian mathematician János Bolyai. The main difference between this geometry and Euclidean geometry is that it abandons the parallel postulate of Euclidean geometry, which means that in hyperbolic geometry, through a point not on a given line, multiple lines can be drawn parallel to the given line.

Some fundamental characteristics of hyperbolic geometry include:

- **Existence of Parallel Lines**: In hyperbolic geometry, given a line and a point not on that line, an infinite number of lines can be drawn through the point that are parallel to the given line.
- **Sum of Angles in a Triangle**: Due to the constant negative curvature of hyperbolic space, the sum of the interior angles of a triangle in hyperbolic geometry is always less than 180 degrees (or π radians).
- **Hyperbolic Plane**: Hyperbolic geometry is typically studied on the hyperbolic plane, a special kind of geometric space that can be imagined as an infinitely large surface where the straight lines are geodesics, the shortest paths between two points on the surface.
- **Hyperbolic Distance**: In hyperbolic geometry, the distance between two points is defined using hyperbolic metrics. Because hyperbolic space is curved with constant negative curvature, the radial approach to distance measurement varies, which is different from the metrics used in Euclidean geometry.

(1)Poincare model

The Poincare disk model is a representation of hyperbolic space that maps hyperbolic space to an open unit disk. In this model, points in hyperbolic space are represented as points inside a disk, while the boundaries of the disk correspond to points at infinity in hyperbolic space. The visualization of the Poincare disk model is shown in Figure 2.1, where the distance between each edge of the nodes is equal. Since this model is presented in two-dimensional Euclidean space, it appears that the closer it is to the center of the circle, the greater the distance, while the distance between the nodes at the original center of the circle is relatively small. So if it is known that the distance between nodes is equal, it can be seen that nodes closer to the edge have a greater distance.

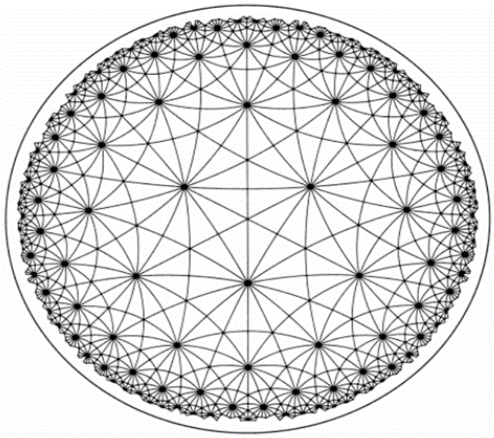


Figure 2.1: Poincare disk model

As shown in Figure 2.2, Suppose we have a hyperbolic world confined within the boundary of a unit circle, and this world is governed by two important physical laws:

1. If an object X is at a distance d from the origin O, then the temperature of the object is given by $1-d^2$.

2. The size of an object is directly proportional to its temperature.

In this hyperbolic world, the temperature T of an object X as a function of its distance d from the origin O can be expressed as:

$$T = 1 - d^2 \tag{2.6}$$

Let S represent the size of the object, which is proportional to its temperature. If we denote the proportionality constant by k, then the size S can be written as:

$$S = kT = k(1 - d^2) \tag{2.7}$$

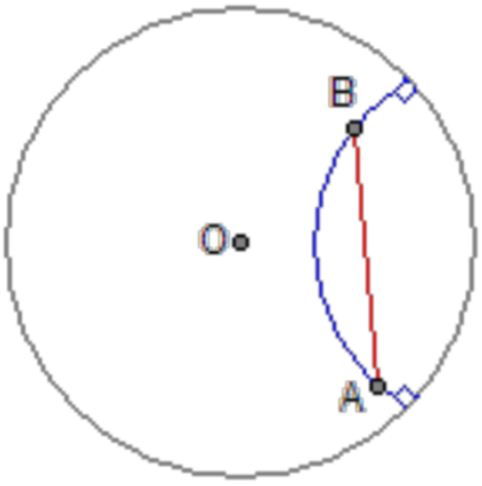


Figure 2.2: Hyperbolic space represented by unit circles

The Poincare model B^d is a manifold equipped with the Riemannian metric $g^B = \lambda_u^2 g^E$, where $\lambda_u^2 =$

$\frac{2}{1-\|u\|^2}$ and g^E is the Euclidean metric. For a d-dimensional hyperbolic space, the Poincare ball model can be defined within the unit ball of d-dimensional Euclidean space:

$$B^d = \{u \in \mathbb{R}^d : \|u\| < 1\} \tag{2.8}$$

Here, $\|\cdot\|$ denotes the Euclidean norm. The tangent space $T_u B^d$ at a point u on B^d is a d-dimensional Euclidean space, which is closest to the neighborhood of point u in B^d. Given two points $u, v \in B^d$ in the Poincare ball, and a vector $t \in T_u B^d$, the exponential map $\exp_u(t) : T_u B^d \to B^d$ and the logarithmic map $\log_u(v) : B^d \to T_u B^d$ can be expressed as follows:

$$\exp_u(t) = u \oplus \left(\tanh\left(\frac{\|t\|}{1 - \|u\|^2} \right) \frac{t}{\|t\|} \right) \tag{2.9}$$

The logarithmic map $\log_u(v)$ is defined as:

$$\log_u(v) = (1 - \|u\|^2) \cdot \tanh^{-1}(\| - u \oplus v\|) \frac{-u \oplus v}{\| - u \oplus v\|} \tag{2.10}$$

where $\oplus$ denotes the Möbius addition.

Formally, the distance between two points $u, v \in B^d$ in the Poincare ball is defined as:

$$d_B(u, v) = \text{arcosh}\left(1 + 2\frac{\|u - v\|^2}{(1 - \|u\|^2)(1 - \|v\|^2)}\right) \tag{2.11}$$

One of the characteristics of the Poincare ball model is its angle-preserving property. It maintains the measurement of angles, meaning that the intersection angle of two curves in hyperbolic space is equivalent to the intersection angle of their representations in the Poincare ball model. Additionally, in the Poincare ball model, the boundary of the ball corresponds to the points at infinity in hyperbolic space. Although the boundary of the ball is not part of hyperbolic space itself, it plays a significant role in helping us understand the structure of hyperbolic space.

(2)Lorentz model

The Lorentz model, also known as the hyperboloid model, is a way to represent hyperbolic geometry. It is equipped with a metric tensor g^H and is defined as the upper sheet of a two-sheeted hyperboloid in $\mathbb{R}^{d+1}$:

$$H^d = \{u \in \mathbb{R}^{d+1} : \langle u, u \rangle_L = -K, u_0 > 0\} \tag{2.12}$$

where $\langle \cdot, \cdot \rangle_L$ denotes the Lorentzian inner product, and the curvature is $-1/K$ with $K > 0$. For two points $u, v \in \mathbb{R}^{d+1}$ on the hyperboloid, the Lorentzian inner product is defined as:

$$\langle u, v \rangle_L = u^T g^H v = -u_0 v_0 + \sum_{i=1}^{d} u_i v_i \tag{2.13}$$

Here, g^H is a diagonal matrix with all elements being 1 except for the first element, which is -1.

The distance between two points $u, v \in H^d$ in the Lorentz model is given by:

$$d_H^K(u,v) = \sqrt{K}\text{arcosh}\left(-\frac{\langle u,v\rangle_L}{K}\right) \tag{2.14}$$

For points $u, v \in H^d$, and a vector $t \in T_u H^d$ with $t \neq 0$ and $v \neq u$, the exponential map $\exp_u^K : T_u H^d \to H^d$ and the logarithmic map $\log_u^K : H^d \to T_u H^d$ are defined as:

$$\exp_u^K(t) = \cosh\left(\frac{\|t\|_L}{\sqrt{K}}\right) u + \sqrt{K}\sinh\left(\frac{\|t\|_L}{\sqrt{K}}\right)\frac{t}{\|t\|_L} \tag{2.15}$$

$$\log_u^K(v) = d_H^K(u,v)\frac{v + \frac{1}{K}\langle u,v\rangle_L u}{\|v + \frac{1}{K}\langle u,v\rangle_L u\|_L} \tag{2.16}$$

where $\|t\|_L = \sqrt{\langle t,t\rangle_L}$ is the Lorentzian norm of t. The tangent space at point u in the Lorentz model H^d is defined as the d-dimensional vector space that approximates H^d to the first order at u:

$$T_u H^d = \{t \in \mathbb{R}^{d+1} : \langle t,u\rangle_L = 0\} \tag{2.17}$$

The Lorentz model provides an effective space for Riemannian optimization and has the advantage of numerical stability. It is particularly useful for learning hierarchical structures from large-scale unstructured similarity scores. The model's ability to avoid numerical instability and its capacity for Riemannian optimization make it a valuable tool for handling complex relationships in large datasets.

2.5 Mercator

We have provided a detailed introduction to the concept and model of hyperbolic space above, and many existing algorithm models are derived from classical hyperbolic geometry models. For example, Maximilian Nickel et al. proposed an algorithm model based on the Poincare disk model, which was derived from the Poincare disk model. Here we will provide a detailed introduction to the hyperbolic embedding method used in this article, which is the D-Mercator method.

The D-Mercator is a model-based embedding method that produces multidimensional maps of real networks into the $(D+1)$-hyperbolic space, where the similarity subspace is represented as a D-sphere. It is an extension of the Mercator tool, which embeds networks into the hyperbolic plane, and is particularly suited for networks that are better described by a multidimensional formulation of the underlying geometric model.

The D-Mercator [19] method is based on the multidimensional formulation of the geometric soft configuration model, the $S^D = H^{D+1}$ model, which is a multidimensional generalization of the S^1 model. For example, the similar subspace of the hyperbolic space with D=2 in Figure 2.3 is equivalent to the surface of a sphere in three-dimensional Euclidean space. Here, the hyperbolic space will be visualized and represented more vividly in Euclidean space. After embedding data into hyperbolic space, nodes will be embedded on the surface of a sphere, representing individual data, just like in the graph. Then the hyperbolic distance between nodes here is determined by the angular coordinates and radial distance. In

this model, a node i is assigned a hidden variable representing its popularity, influence, or importance, denoted κ_i and named hidden degree. It is also assigned a position in the D-dimensional similarity space chosen uniformly at random, and represented as a point on a D-dimensional sphere.

The connection probability between a node i and a node j takes the form of a gravity law:

$$p_{ij} = \frac{1}{1 + \left(\frac{R\Delta\theta_{ij}}{\mu\kappa_i\kappa_j}\right)^{\beta}} \tag{2.18}$$

where $\Delta\theta_{ij} = \arccos\left(\frac{v_i \cdot v_j}{R^2}\right)$ represents the angular distance between nodes i and j in the D-dimensional similarity space, $\beta > D$ is the inverse temperature that calibrates the coupling of the network topology with the underlying metric space, and μ controls the average degree of the network. The parameter μ controls the average degree of the network and is defined as:

$$\mu = \frac{\beta\Gamma\left(\frac{D}{2}\right)\sin\left(\frac{D\pi}{2\beta}\right)}{2\pi^{1+\frac{D}{2}}\left(1+\frac{D}{2}\right)\langle k\rangle} \tag{2.19}$$

where Γ is the Gamma function, D is the dimensionality of the similarity subspace, and $\langle k\rangle$ is the average degree of the network. The hyperbolic distance d between two points with radial coordinates r_i and r_j, and angular separation $\Delta\theta$ can be expressed as:

$$d = \cosh(r_i)\cosh(r_j) - \sinh(r_i)\sinh(r_j)\cos(\Delta\theta) \tag{2.20}$$

The hyperbolic radius r_i is given by:

$$r_i = \hat{R} - \frac{2}{D}\ln\left(\frac{\kappa_i}{\kappa_0}\right), \tag{2.21}$$

where $\hat{R}$ is defined as:

$$\hat{R} = 2\ln\left(\frac{2R}{\left(\mu\kappa_0^2\right)^{1/D}}\right). \tag{2.22}$$

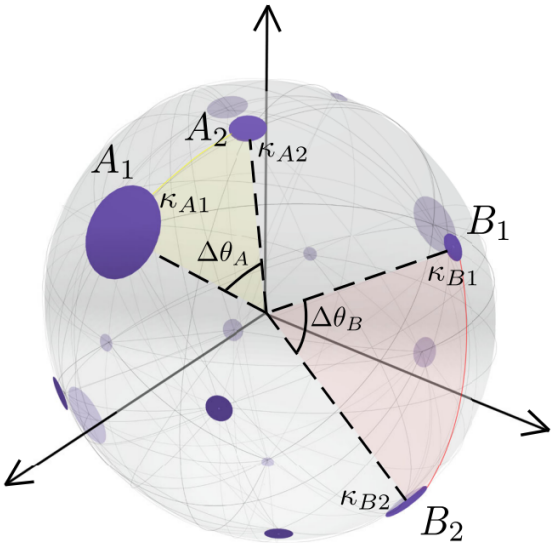


Figure 2.3: Geometric soft configuration model (Source: Jankowski et al., 2023)

2.6 Summary

This chapter mainly introduces some relevant theoretical knowledge contained in this thesis. Specifically, this chapter first provides an overview of the basic concepts and applications of graph data, and explains what the AS dataset used in this thesis is. Then, the basic concepts and mathematical background knowledge related to hyperbolic geometry were introduced, as well as the relevant content of Euclidean geometry, which can be compared to understand the difference between hyperbolic space and Euclidean space. Then, two isometric models of hyperbolic geometry were introduced successively, including the Lorentz (hyperbolic) model and the Poincare model. Subsequently, in order to better introduce the experiments conducted in the following text, the D-Mercator model from network science used in this article was emphasized, and its various formulas for hyperbolic distance calculation were detailed.

3 Quantifying Embedding Quality

3.1 Greedy Routing

Greedy routing is a strategy used in networks to find a path from a source node to a target node. In this method, each node forwards a message to its neighbor that is closest (according to some distance metric) to the target node. This process is repeated until the message reaches the target node. Greedy routing was systematically analyzed by Boguñá et al. (2009) [39], who demonstrated its effectiveness in complex networks. Furthermore, Krioukov et al. (2010) [5] showed that when networks are embedded into hyperbolic space, greedy routing can achieve near-optimal efficiency due to the inherent hierarchical and geometric properties of hyperbolic embeddings. The key characteristics of greedy routing include:

- **Local Decision Making:** Each node makes forwarding decisions based only on information about its immediate neighbors, without requiring global network knowledge.
- **Simplicity:** The algorithm is straightforward to implement because it relies on local information.
- **Efficiency:** In some network topologies, greedy routing can quickly find a path to the target node, especially when the network has favorable structural properties.
- **Potential Suboptimal Solutions:** Greedy routing does not guarantee finding the globally optimal path; it may become trapped in local minima [40], in which case the path is considered unsuccessful.
- **Dynamic Adaptability:** Greedy routing can adapt to dynamic changes in the network since it depends only on current neighbor information.
- **Effectiveness in Specific Networks:** Greedy routing can be very effective in certain types of networks, such as small-world networks [41] or scale-free networks [42], due to their highly clustered nature, which allows greedy routing to quickly find a path to the target.

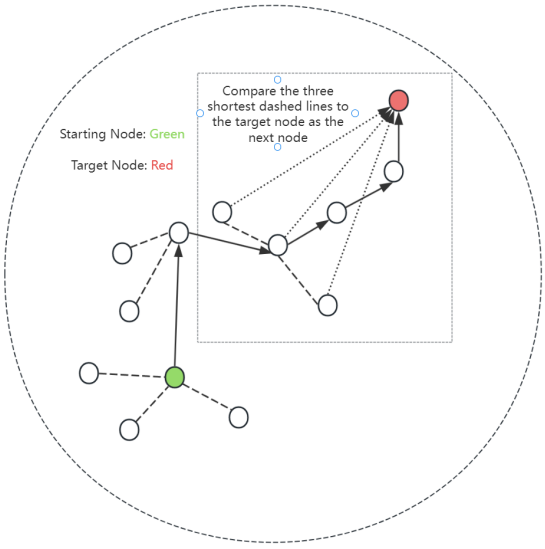


Figure 3.1: Greedy Routing Flowchart

The specific process of greedy routing is illustrated in Figure 3.1. After completing the node embedding, we obtain the coordinate information of the nodes in hyperbolic space. We randomly select two nodes as the starting node and the target node, respectively. Starting from the starting node, the data packet is carried, and the distance from its neighboring nodes to the target node is calculated. The data packet is then sent to the neighbor node that is closest to the target node. For each node storing the data packet, we consider its neighbors. We then calculate the hyperbolic distance from each neighbor to the target and forward the data packet to the neighbor with the smallest distance, repeating this process until the target node is reached. If a data packet visits the same node twice, it will be discarded, considered a routing failure, and a new routing cycle will begin.

If the embedding quality is good, the success rate of greedy routing will be high, indicating that the embedding method can effectively capture the hierarchical relationships between nodes and accurately reflect the connection patterns between nodes and their neighbors. Specifically, high-quality embedding maps the relationships between nodes in the topological structure into hyperbolic space, enabling greedy routing to quickly find the optimal path based on hyperbolic distance. Conversely, if the embedding quality is poor, the success rate of greedy routing will be low, indicating that the embedding method fails to effectively capture the hierarchical relationships between nodes or accurately reflect the connection patterns between nodes and their neighbors, and even struggles to distinguish the importance or weights of interconnected nodes.

3.2 Missing Link Prediction

Missing link prediction is an important method for evaluating the quality of network embedding. The core idea is to randomly remove some connected node pairs (i.e. links) and use embedding methods to regenerate the hyperbolic coordinates of nodes, in order to predict whether these removed links can be accurately restored. Link prediction was originally proposed by Liben-Nowell and Kleinberg (2003) [43] to study the evolution and structure of complex networks. With the development of graph representation learning, embedding-based methods have become widely adopted. Early works such as DeepWalk (Perozzi et al., 2014) [44] and node2vec (Grover and Leskovec, 2016) demonstrated that effective node embeddings can significantly improve link prediction performance. The specific procedure is as follows:

1. **Data Preprocessing**: First, a certain proportion of connected node pairs (typically 10%, 20%, or 30%) are randomly removed from the initial graph dataset. These removed node pairs serve as positive examples (i.e., real existing links). The modified graph dataset is then used for the subsequent embedding process.
2. **Embedding Process**: The modified graph dataset is embedded into hyperbolic space using two different methods:
    - **Network Science Method**: The D-Mercator method is used to compute the hyperbolic coordinates of nodes.
    - **Machine Learning Methods**: The Poincare model and the Lorentz model are used to compute the hyperbolic coordinates of nodes.
3. **Constructing the Validation Set**:

- **Positive Examples**: The node pairs randomly removed from the initial graph dataset will serve as the positive examples for the subsequent validation set. However, it is important to note that removing a certain proportion of node pairs may result in the complete removal of nodes with small degrees (i.e., nodes with few neighbors). This means that after embedding into hyperbolic space, we may not obtain the hyperbolic coordinates for these nodes. Consequently, during subsequent evaluation, it will be impossible to calculate the hyperbolic distances for node pairs involving these nodes. Therefore, we need to use the node ID information from the embedded nodes and compare it with all the IDs in the randomly removed node pairs to identify those node IDs that exist only at one end. We then remove the node pairs associated with these nodes from the randomly removed node pair set. The remaining node pairs constitute the true positive examples for the validation set.
- **Negative Examples**: We consider all unconnected node pairs by exhaustively pairing all nodes and removing any existing links from the original dataset. Using all unconnected node pairs as negative samples better approximates real-world application scenarios and prevents evaluation fluctuations caused by random sampling.
- The positive and negative examples are combined into a validation set for subsequent evaluation.

4. **Calculating Hyperbolic Distance**: Based on the vectors of the embedded nodes, the hyperbolic distance between all node pairs in the validation set is calculated using the corresponding hyperbolic distance formulas (e.g., Poincare distance or Lorentz distance). The distances are then sorted in ascending order. In hyperbolic space, the probability of a link existing between two nodes is inversely proportional to their hyperbolic distance, meaning that a smaller hyperbolic distance indicates a higher probability of a link existing between the nodes.
5. **Evaluating Embedding Quality**: Compare the hyperbolic distance distribution between the positive samples (deleted real edges) in the validation set and all other unconnected negative samples in the embedding space. If the embedding quality is good, the positive sample edges in the validation set should be significantly concentrated in a smaller distance interval, while the negative sample edges should be distributed in a larger distance range. By plotting ROC and PR curves and calculating metrics such as recall, AUC, and AUPR, the performance of embedding methods in missing link prediction tasks can be quantified.

3.2.1 ROC Curve

The Receiver Operating Characteristic (ROC) curve is a graphical tool used to evaluate the performance of binary classification models. It demonstrates the classification capability of a model by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various thresholds. In this context, we will plot the ROC curves for the missing link prediction task using different methods. To achieve this, we first sort the validation sets of each method in ascending order based on hyperbolic distance. Then, we use each row's hyperbolic distance as a threshold to calculate the TPR and FPR within that threshold. This process is repeated until the last row's distance is used as the threshold. By plotting the TPR and FPR at each threshold on the graph, we obtain the ROC curve. In the context of missing link prediction:

- **True Positive Rate (TPR)**: The proportion of actual positive examples (removed links) that are correctly predicted as positive. The formula for TPR is:

$$\text{TPR} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \tag{3.1}$$

Here, TP means true positive, indicating the number of positive cases correctly predicted as positive cases; FN stands for false negative, indicating the number of positive cases that were incorrectly predicted as negative. The sum of the two is the sum of the number of positive examples in the validation set.

- **False Positive Rate (FPR)**: The proportion of actual negative examples (non-existent links) that are incorrectly predicted as positive. The formula for FPR is:

$$\text{FPR} = \frac{\text{False Positives (FP)}}{\text{False Positives (FP)} + \text{True Negatives (TN)}} \tag{3.2}$$

Here, FP means false positive, indicating the number of negative cases that were incorrectly predicted as positive; TN means true negative, indicating the number of negative cases correctly predicted as negative. The sum of the two is the sum of the number of negative examples in the validation set.

The ROC curve provides a visual assessment of the trade-off between sensitivity (TPR) and specificity (1 - FPR). A model with perfect classification ability will have an ROC curve that passes through the top-left corner of the plot, indicating a TPR of 1 and an FPR of 0. In Figure 3.2 below, we plot the ROC curve for the D-Mercator method with 10% of the links removed. The fact that the prediction curve lies above the diagonal demonstrates that the model has predictive capability. From the figure, we can see that the curve is significantly above the diagonal, indicating that the method can effectively distinguish between missing links and non-existent links, thereby achieving the goal of predicting missing links. The results for other proportions of missing links are also excellent, but they are not shown here to avoid clutter.

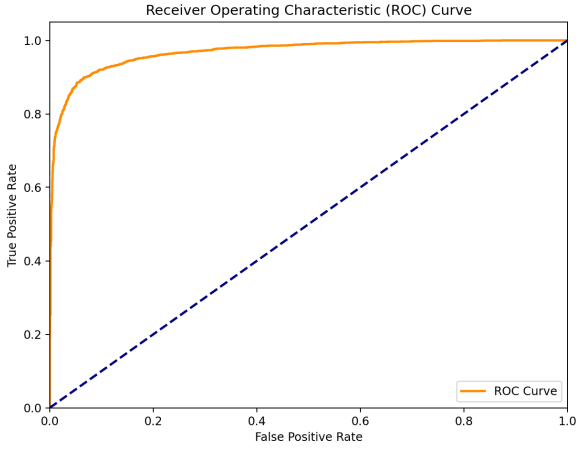


Figure 3.2: Receiver Operating Characteristic (ROC) of D-Mercator

3.2.2 AUC

AUC (Area Under the ROC Curve) is the area under the ROC curve used to quantify the overall performance of binary classification models. It is a scalar value between 0 and 1, which can comprehensively

reflect the classification ability of the model at different thresholds. In missing link prediction, AUC is a key metric for evaluating the quality of embedding methods. Specifically:

- **High-Quality Embedding**: If AUC is close to 1, the embedding method can effectively distinguish between missing links (positive examples) and non-existent links (negative examples), indicating that the method captures the network's topological structure and node relationships.
- **Low-Quality Embedding**: If AUC is close to 0.5, the embedding method fails to distinguish between missing links and non-existent links, suggesting that the method does not adequately reflect the network's topological properties.

AUC represents the probability that a randomly chosen positive example will be ranked higher than a randomly chosen negative example.It is calculated as:

$$\text{AUC} = \int_0^1 \text{TPR}(FPR)\, d(FPR) \tag{3.3}$$

AUC offers several significant advantages that make it a widely used metric for evaluating binary classification models. First and foremost, AUC is threshold-independent, meaning it provides a comprehensive assessment of a model's performance across all possible classification thresholds, rather than being limited to a single cutoff point. This is particularly valuable in scenarios where the optimal threshold is unknown or may vary depending on the application. Additionally, AUC is robust to class imbalance, making it suitable for datasets where the number of positive and negative examples is highly skewed. Unlike metrics such as accuracy, which can be misleading in imbalanced datasets, AUC focuses on the model's ability to rank positive examples higher than negative ones, offering a more reliable evaluation. Furthermore, AUC provides an intuitive and interpretable single scalar value, facilitating easy comparison between different models or embedding methods. In the context of missing link prediction, AUC's ability to capture the trade-off between sensitivity (TPR) and specificity (1 - FPR) makes it an ideal metric for assessing how well an embedding method can distinguish between missing links (positive examples) and non-existent links (negative examples). A high AUC value indicates that the model can effectively rank true missing links higher than non-existent ones, reflecting the embedding method's success in preserving the network's topological structure and node relationships. Overall, AUC's threshold independence, robustness to imbalance, and interpretability make it a powerful tool for evaluating and comparing the performance of classification models, especially in complex tasks like missing link prediction.

3.2.3 PR Curve

The Precision-Recall (PR) curve is a graphical tool used to evaluate the performance of binary classification models, particularly in imbalanced datasets. Unlike the ROC curve, the PR curve plots Precision against Recall at various thresholds, providing a clear view of the trade-off between these two metrics. Below is a detailed explanation of the PR curve:

- **Recall**: The proportion of correctly predicted positive cases in the actual positive cases (removed links) is represented by TPR, which is the recall rate. Therefore, the calculation formula is the same as Equation 3.1.
- **Precision**: The proportion of samples predicted as positive cases that are actually positive cases is

calculated using the following formula:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}} \tag{3.4}$$

The Precision-Recall (PR) curve illustrates a model's performance by plotting precision and recall values at various thresholds. To construct the PR curve, node pairs are first sorted in descending order based on their predicted scores, with each score treated as a potential threshold. For each threshold, recall (i.e., the true positive rate) and precision (i.e., the proportion of correctly predicted links among all predicted links above the threshold) are calculated. Each point on the PR curve corresponds to the precision and recall values at a specific threshold. Specifically, if the PR curve extends toward the top-right corner of the plot, it indicates that the embedding results achieve high precision and recall, suggesting that the embedding method or model is effective in distinguishing missing links from non-existent ones. In contrast, if the PR curve remains close to a horizontal line, it suggests that the embedding method fails to effectively differentiate between missing and non-existent links, indicating a need for further optimization of the embedding approach or adjustment of model parameters.

Ideally, the PR curve should extend horizontally from the top-left corner (0,1) to the top-right corner (1,1), and then drop vertically to the bottom-right corner (1,0), forming an "L"-shaped curve. As shown in Figure 3.3, this shape indicates that the model maintains perfect precision across all levels of recall. Although such an ideal curve is rarely achieved in practical applications, it serves as a theoretical upper bound for evaluating model performance. Furthermore, by comparing actual PR curves to this ideal, the closer a curve approaches the top-right corner, the better the corresponding model performs.

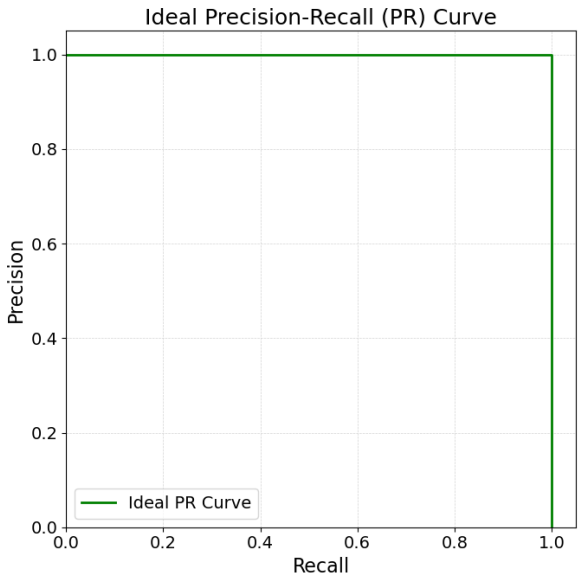


Figure 3.3: Precision-Recall(PR) of Poincare

The Precision-Recall (PR) curve offers several key advantages, making it a powerful tool for evaluating binary classification models, particularly in scenarios with class imbalance. Unlike the ROC curve, which can be overly optimistic in the presence of class imbalance, the PR curve specifically focuses on the performance of the positive class, making it more sensitive to datasets where positive examples are

rare. As a result, the PR curve complements the ROC curve, providing a multifaceted perspective on the quality of embedding methods for datasets. By plotting the relationship between Precision and Recall at various classification thresholds, the PR curve clearly illustrates the trade-off between these two critical metrics. Precision measures the proportion of predicted positive examples that are actually correct, which is particularly important in applications where false positives are costly. On the other hand, Recall measures the proportion of actual positive examples that are correctly identified, which is crucial for tasks where missing positive examples (e.g., missing links in a network) can have significant consequences. Overall, the PR curve's focus on the positive class, its sensitivity to class imbalance, and its ability to visualize the Precision-Recall trade-off make it an indispensable tool for evaluating the performance of classification models in real-world applications such as missing link prediction.

3.2.4 AUPR

AUPR (Area Under the Precision-Recall Curve) is a key metric for evaluating classification model performance, particularly suited for imbalanced datasets. Unlike ROC-AUC, AUPR places greater emphasis on the prediction quality of positive-class samples (typically the minority class), making it widely applicable in fields such as medical diagnosis, anomaly detection, and in formation retrieval.

The calculation of AUPR is based on the integral of the precision recall curve, and its core formula is as follows:

$$\text{AUPR} = \int_0^1 \text{Precision}(r)\, dr \tag{3.5}$$

Here r denotes the recall and $\text{Precision}(r)$ represents the precision at recall level r.

Due to the fact that the actual data is discrete, it is usually approximated by the 1-trapezoid rule:

$$\text{AUPR} \approx \sum_{k=1}^{n-1} (r_{k+1} - r_k) \cdot \frac{\text{Precision}(r_{k+1}) + \text{Precision}(r_k)}{2} \tag{3.6}$$

Here r_k and $\text{Precision}(r_k)$ represent the recall and precision at the k-th threshold respectively, and n denotes the total number of threshold points.

AUPR exhibits unique advantages that establish it as a core metric for evaluating classification models, particularly for class-imbalanced problems. First, AUPR is threshold-independent, providing a comprehensive assessment of the precision-recall trade-off across all possible classification thresholds rather than relying on a single threshold setting. This characteristic proves particularly valuable when the optimal decision boundary is ambiguous or requires dynamic adjustment. Second, AUPR is specifically designed for imbalanced data. When the positive class ratio is extremely low, AUPR offers a more realistic performance evaluation than the potentially inflated ROC-AUC by focusing on both prediction quality (precision) and coverage capability (recall) of positive samples. AUPR directly measures a model's ability to rank positive instances, ensuring high-confidence predictions prioritize covering true positives.

In missing link prediction tasks, AUPR precisely captures a model's discriminative power between sparse positive samples (true missing links) and abundant negative samples (non-existent links). A high AUPR value indicates that the embedding method can not only accurately identify missing links (high recall) but also maintain prediction reliability (high precision), reflecting how well the embedding space preserves

the network's topological structure.

3.3 Rank Correlation and Pearson Coefficient

The greedy routing and missing link prediction experiments discussed above provide effective ways to quantify the embedding quality of the Poincare, Lorentz, and D-Mercator methods on graph datasets. By examining the final results of these experiments, we can distinguish the performance of the three embedding methods and determine which one achieves better embedding quality. However, while these experiments allow us to individually assess the performance of each method—identifying which methods excel and which fall short—they do not reveal whether the embeddings generated by the three methods are correlated when applied to the same dataset. For instance, do the embeddings exhibit linear or nonlinear relationships? Are the results of one method consistent with those of another? Although the formulas used by the three methods to compute hyperbolic embeddings are different, we hypothesize that the embeddings of high-degree nodes (those with many neighbors) may exhibit similarities, which requires experimental validation. To address these questions, we introduce rank correlation coefficients (such as Spearman's rank correlation) and Pearson correlation coefficients, along with rank correlation scatter plots, to investigate whether the embeddings produced by the three methods are correlated. This analysis not only complements the previous experiments but also provides deeper insights into the relationships between the embedding methods, helping us understand whether they capture similar structural properties of the graph data.

3.3.1 Rank Correlation Coefficients

The rank correlation coefficient is a statistical measure used to assess the strength of a monotonic relationship between two variables. Unlike the Pearson correlation coefficient, which measures linear relationships, rank correlation coefficients are based on the ranks of the data rather than their raw values. This makes them particularly useful for analyzing non-linear but monotonic relationships, as well as datasets with outliers or non-normal distributions. Below, we introduce two commonly used rank correlation coefficients: Spearman's rank correlation coefficient [45] [46] and Kendall's rank correlation coefficient [47] [48].

- **Spearman's Rank Correlation Coefficient**: Spearman rank correlation coefficient is the most commonly used type of rank correlation coefficient, denoted as ρ, measures the strength and direction of the monotonic relationship between two variables. It is calculated as:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \tag{3.5}$$

  d_i is the difference in ranks between the two variables. In the context of graph data, after embedding the data into three different hyperbolic spaces to obtain the corresponding hyperbolic coordinates, the hyperbolic distances between each pair of nodes are calculated using the three different hyperbolic distance formulas and then ranked. Here, d_i represents the difference in the rankings of the hyperbolic distances for any given pair of nodes in any two hyperbolic spaces.n is the number of samples.

The Spearman rank correlation coefficient ranges from $[-1, 1]$, where $p = 1$ indicates a perfect positive correlation, meaning the ranks of the two variables are completely consistent; $p = -1$ indicates a perfect negative correlation, meaning the ranks of the two variables are completely opposite; and $p = 0$ indicates no correlation, meaning there is no monotonic relationship between the ranks of the two variables. One of the key advantages of the Spearman rank correlation coefficient is that it does not rely on the linear relationship of the data, making it suitable for nonlinear but monotonic relationships. Additionally, it is less sensitive to outliers because it is based on ranks rather than raw values.

- **Kendall's Rank Correlation Coefficient**: Kendall rank correlation coefficient is another commonly used rank correlation coefficient, denoted as τ, is another measure of rank correlation. It is calculated as:

$$\tau = \frac{C - D}{\frac{1}{2}n(n-1)} \tag{3.6}$$

The calculation of Kendall's rank correlation coefficient is based on the number of concordant pairs and discordant pairs. Here, C represents the number of concordant pairs, where the rank changes of the two variables are in the same direction; D represents the number of discordant pairs, where the rank changes of the two variables are in opposite directions; and n is the sample size. Specifically, for any two data points (x_i, y_i) and (x_j, y_j), if $x_i > x_j$ and $y_i > y_j$, or $x_i < x_j$ and $y_i < y_j$, these pairs are considered concordant because their rank changes are in the same direction. Conversely, if $x_i > x_j$ and $y_i < y_j$, or $x_i < x_j$ and $y_i > y_j$, these pairs are considered discordant because their rank changes are in opposite directions. By comparing the number of concordant and discordant pairs, Kendall's rank correlation coefficient measures the strength of the monotonic relationship between the two variables.

Kendall's rank correlation coefficient also ranges from $[-1, 1]$, with interpretations similar to those of Spearman's rank correlation coefficient. One of its key advantages is that it is more robust for smaller datasets, making it particularly useful when the sample size is limited. Additionally, it is suitable for non-parametric statistics, as it does not rely on assumptions about the distribution of the data, providing greater flexibility in various analytical scenarios.

Here, since our graph dataset has a relatively large sample size, Kendall's rank correlation coefficient is not suitable. Therefore, we choose Spearman's rank correlation coefficient to explore the nonlinear but monotonic relationships between the three embedding methods. Additionally, because the rank correlation coefficient is based on ranks rather than raw values, it is less sensitive to outliers, effectively mitigating the impact of possible anomalies in the embedding.

3.3.2 Pearson Correlation Coefficient

The Pearson correlation coefficient, also known as Pearson's r, is a measure of the linear correlation between two variables. It quantifies the degree to which a linear relationship exists between the variables, ranging from -1 to 1.The Pearson correlation coefficient is defined as the covariance of the two variables divided by the product of their standard deviations. Mathematically, it is expressed as:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{3.7}$$

x_i and y_i are the individual data points of the two variables.$\bar{x}$ and $\bar{y}$ are the means of the two variables.n is the number of data points.

Its range is $[-1, 1]$, where $r = 1$ indicates a perfect positive linear correlation, meaning the two variables increase or decrease together in a perfectly linear fashion; $r = -1$ indicates a perfect negative linear correlation, meaning one variable increases while the other decreases in a perfectly linear fashion; and $r = 0$ indicates no linear correlation, meaning there is no linear relationship between the variables. One of the key advantages of the Pearson correlation coefficient is its sensitivity to linear relationships, making it highly effective for detecting linear dependencies between variables. Additionally, the value of r provides a clear and intuitive measure of the strength and direction of the linear relationship, and it is widely used in various fields such as statistics, machine learning, and social sciences. However, it has some limitations: it is sensitive to outliers, which can significantly affect the value of r and lead to misleading conclusions, and it assumes a linear relationship, meaning it may fail to capture nonlinear relationships between variables.

In this study, the Pearson correlation coefficient is used to evaluate the linear relationship between embeddings generated by the different methods (i.e., the Poincare model, the Lorentz model, and the D-Mercator method). Specifically, for each method, nodes with degrees exceeding a certain threshold, i.e., those that are more central, are selected, and the hyperbolic distances between node pairs are calculated using their hyperbolic coordinates. The Pearson correlation coefficient is then computed between the distance matrices of different methods. A high r value (close to 1 or -1) indicates a strong linear relationship, suggesting that the methods capture similar structural properties of the graph data. A low r value (close to 0) indicates little to no linear relationship, suggesting that the methods capture different structural properties of the graph data.

3.4 Summary

This chapter focuses on the metrics we used to quantitatively compare the embedding quality of the considered hyperbolic embedding methods and to investigate whether the hyperbolic distances between node pairs, across embeddings obtained by embedding the same dataset using different methods, exhibit linear or nonlinear relationships. The dataset used in this study is the AS (Autonomous System) dataset, where nodes represent AS entities and edges represent connections between them. The types of methods compared are: (1) network science methods, represented by the D-Mercator method, and (2) machine learning methods, represented by the Poincare model and the Lorentz model.

First, we overviewed the specific process of greedy routing. If the embedding quality is good, the success rate of greedy routing will be high. Therefore, we used the embedding results of the dataset obtained by the three methods to quantify the embedding quality through greedy routing, thereby identifying which method performs best. Next, we described the process of predicting missing links. By plotting ROC and PR curves, calculating AUC and AUPR values, we quantified the quality of missing chain prediction, which further reflects the embedding quality.

The first two parts of the chapter focused on comparing the embedding quality of the three methods but did not explore whether the embeddings generated by these methods capture similar structural properties of the dataset, such as whether the hyperbolic distances exhibit linear or nonlinear relationships. To

address this, we used the Pearson correlation coefficient to investigate the linear relationships between the embeddings generated by the three methods. However, due to the limitations of the Pearson correlation coefficient—such as its sensitivity to outliers, inability to capture nonlinear relationships, and potential issues with inconsistent scales in the hyperbolic spaces of different methods—we also introduced rank correlation coefficients (Spearman's rank correlation) to explore nonlinear but monotonic relationships. Additionally, we plotted rank correlation scatter plots based on the rankings of nodes to visually assess the similarities and differences between the methods.

4 Results

4.1 Setup

Before presenting the results, let us first describe the graph dataset used. We use the data set of the autonomous system AS, which contains 10259 different nodes, indicating that there are actually 10259 different autonomous systems. There are still 27027 edges, which means that 10259 autonomous systems make up 27027 connections and information exchanges. The following results were obtained by applying the three embedding methods (D-Mercator, Poincare, Lorentz) onto this dataset and analyzing them. In the embedding phase, since the parameters in D-Mercator are automatically calculated within the program, we will not provide further explanation. In the embedding configuration stage of the Poincare and Lorentz models, we systematically evaluated multiple configurations, among which the optimal choice was to set the learning rate to 0.1, the number of negative samples to 100, the importance weight coefficient of negative samples to 1.8, the training period to 5000, the warm-up period to 100, and the margin to 0.2. Due to space constraints and in order to keep the main text concise, the full hyperparameter configurations for both the Poincaré and Lorentz embeddings are listed in Appendix I.

The three embedding methods all use three-dimensional hyperbolic space, which is an optimized choice based on the strong hierarchical characteristics of the autonomous system. The D-Mercator method automatically infers three dimensions as the optimal dimension through likelihood maximization, and its curvature parameter can be adapted to the hierarchical depth of the AS graph dataset; For Poincare and Lorentz methods, three-dimensional hyperbolic space can balance the completeness of hierarchical expression and model complexity, preventing overfitting.

4.2 Greedy Routing Results

We first present the greedy routing results. We calculate the success rate, the average hop count and the highest success rate of three greedy routing methods with 5000, 8000, and 10000 greedy routing attempts. Success rate, refers to the proportion of successful routing; the average hop count refers to the number of nodes that pass from the source node to the destination node when the route is successful; The highest success rate refers to the highest proportion of successful routing across different runs. Each greedy attempt is executed 50 times.

This table compares the performance of three models (D-Mercator, Poincare, and Lorentz) under different greedy routing attempts (5000, 8000, and 10,000), with key metrics including success rate, average hops, and highest success rate. From the results, the Lorentz model performs the best, achieving the highest success rate (approximately 0.9247–0.9260), the lowest average hops (approximately 4.8368–4.8736), and the highest peak success rate (approximately 0.9310–0.9336), making it suitable for scenarios requiring efficient routing. The Poincare model ranks second, with a success rate of 0.9234–0.9246 and average hops of 5.0407–5.0889, indicating competitive performance but slightly inferior to Lorentz in all metrics. The D-Mercator model ranks third, with a success rate of 0.8770–0.8794 and an average hop count of 5.5011–5.5346, suggesting limitations in local path optimization. Overall, as the number of at-

Table 4.1: Greedy routing performance

Greedy Routing Attempts	Model	Success rate	Average hops	Highest success rate
5000	D-Mercator	0.8794	5.5011	0.8852
	Poincare	0.9236	5.0407	0.9276
	Lorentz	0.9247	4.8736	0.9310
8000	D-Mercator	0.8783	5.5346	0.8837
	Poincare	0.9246	5.0484	0.9285
	Lorentz	0.9260	4.8368	0.9336
10000	D-Mercator	0.8770	5.5248	0.8829
	Poincare	0.9234	5.0889	0.9280
	Lorentz	0.9254	4.8523	0.9326

tempts increases, the performance of all models shows minimal variation, indicating algorithmic stability at larger attempt counts, with the Lorentz model consistently outperforming others.

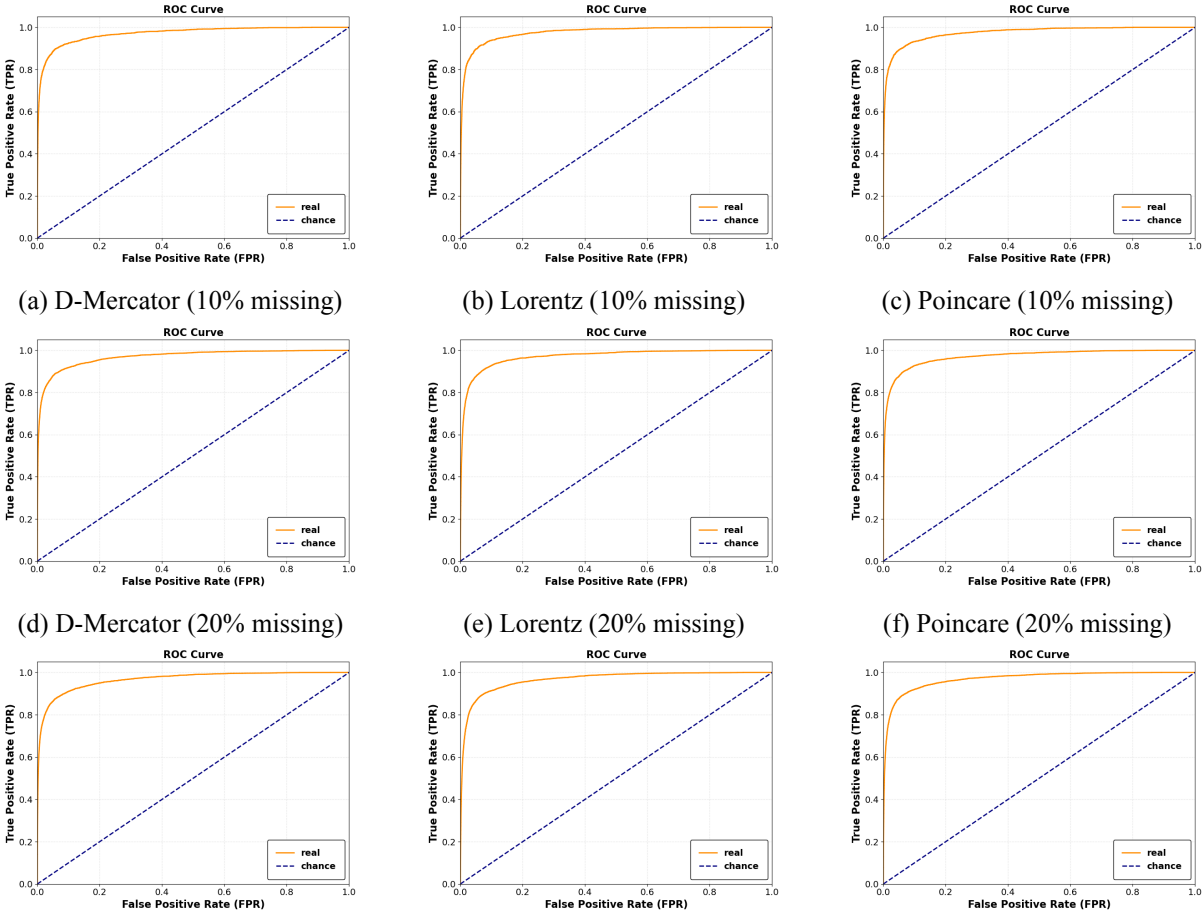
From the differences in routing success rate and average hop count observed in greedy routing among the three methods, we can analyze and conclude the following: Greedy routing inherently relies on local neighbor selection for path planning, a mechanism that prioritizes adaptability to local network structures. In this context, the D-Mercator method, based on the geometric model S^D, generates embeddings by maximizing network growth likelihood. While this approach ensures global topological coherence, its emphasis on global optimization limits its ability to capture fine-grained local structural details, resulting in routing efficiency slightly lower than machine learning methods (e.g., Lorentz). Nevertheless, D-Mercator maintains a relatively high success rate (0.8770–0.8794), underscoring its robustness in preserving global topology.

For machine learning methods (Lorentz/Poincare), both utilize the exponential growth property of hyperbolic space, placing high-level nodes near the center of the embedding space and leaf nodes near the boundary, thus preserving the hierarchical structure. By maximizing the proximity of similar nodes, both can effectively capture local topological relationships and support efficient greedy routing. However, the slight routing differences between them are due to the numerical instability caused by the boundary when calculating the Poincare distance, while the Lorentz model avoids such problems by calculating closed geodesics, making gradient updates smoother, especially for convergence in low dimensional scenarios.

Based on the experimental results and analysis, we conclude that in greedy routing tasks relying on local neighbor selection, the Lorentz model, with its high-dimensional hyperspherical embedding characteristics, can more accurately capture hierarchical relationships in complex networks, making it the optimal choice for complex path planning tasks. In comparison, although the Poincare model also performs well in hyperbolic space, its two-dimensional disk model has limitations in representing high-dimensional complex relationships, resulting in slightly lower routing efficiency than the Lorentz model. Meanwhile, as a physics-inspired global geometric model, D-Mercator maintains good topological consistency but sacrifices optimization of local structural details, leading to a noticeable trade-off in routing performance. Overall, these three models are suitable for different scenarios: Lorentz excels in high-dimensional complex networks, Poincare performs well in low-dimensional tasks requiring interpretability, and D-Mercator is more suitable for theoretical analysis tasks prioritizing global topological fidelity.

4.3 Missing Link Prediction Results

Next, we present the results of the missing link prediction. As shown in the figure below, the ROC curves for the three hyperbolic embedding methods (D-Mercator, Lorentz, and Poincare) are displayed for missing link rates of 10%, 20% and 30%. We observed that the ROC curves in the nine graphs were significantly higher than the diagonal, indicating that in all cases, geometric link prediction was significantly better than chance. This indicates that even if 10%, 20%, and 30% of the links are missing, the three embedding methods can maintain high embedding quality. This shows their strong ability to predict missing links. Furthermore, for the same missing link rate, the ROC curves of the different embedding methods are very close to each other, suggesting that not only is the missing link prediction quality high, but the performance of the three methods is also remarkably similar.



(a) D-Mercator (10% missing) (b) Lorentz (10% missing) (c) Poincare (10% missing)

(d) D-Mercator (20% missing) (e) Lorentz (20% missing) (f) Poincare (20% missing)

(g) D-Mercator (30% missing) (h) Lorentz (30% missing) (i) Poincare (30% missing)

Figure 4.1: ROC curves for D-Mercator, Lorentz, and Poincare embeddings under different missing rates (10%, 20%, and 30%).

However, by merely comparing the ROC curves, it is difficult to discern subtle differences due to the strong performance of all methods, and we cannot further explore the relationships between them. Therefore, to more precisely quantify the embedding quality, we calculated the Area Under the Curve (AUC) based on the aforementioned ROC curves. The AUC value provides an intuitive measure of the model's predictive performance: the closer the AUC is to 1, the better the model's performance.

Table 4.2: AUC Values for Different Models and Missing Link Rates

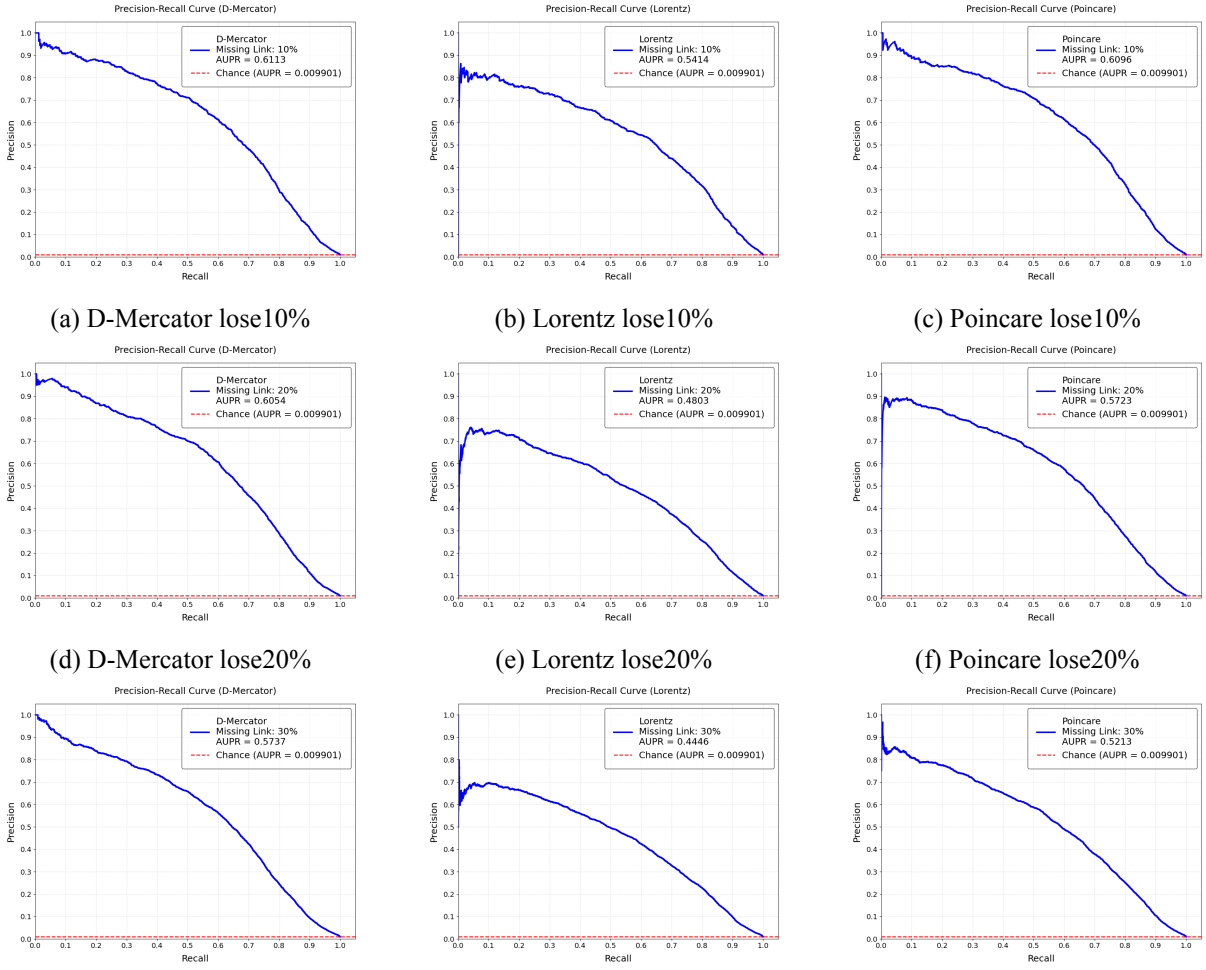
Model	**Lose10%**	**Lose20%**	**Lose30%**
D-Mercator	0.9707	0.9690	0.9665
Poincare	0.9742	0.9702	0.9693
Lorentz	0.9752	0.9702	0.9666

From Table 4.2, it can be observed that the AUC values of all the methods are close to 1, indicating their excellent predictive performance. We find that as the missing link rate increases, the AUC values of all methods decrease, suggesting a slight decline in predictive performance. However, even at a missing link rate 30%, the ROC curves of the three methods remain significantly above the diagonal line, demonstrating their strong robustness to high missing link rates. Across all missing link rates, the AUC values of Lorentz and Poincare are mostly higher than those of D-Mercator, indicating that their predictive performance is slightly better.

The marginally higher AUC of Lorentz could be attributed to its closed-form geodesic formula and efficient Riemannian optimization in hyperbolic space, as demonstrated by Nickel & Kiela (2018) [15]. Their work showed that the Lorentz model's ability to avoid numerical instabilities and precisely compute geodesics leads to more stable embeddings—especially in low dimensions. If this geometric advantage holds for link prediction tasks, it suggests that Lorentz may better preserve hierarchical relationships when global topology is less disrupted (e.g., at low missing rates), though further ablation studies are needed to isolate its impact.

Poincare achieves competitive AUC (0.9741 at 10% missing rate), consistent with prior findings that hyperbolic geometry naturally captures hierarchical relations (Nickel & Kiela, 2017). This might be explained by its conformal distance metric, which induces exponential volume growth—a property theoretically suitable for tree-like structures. If the observed data indeed exhibits latent hierarchies (e.g., clear parent-child relationships at low missing rates), the Poincare model could leverage this geometry to position root nodes near the origin and leaves near the boundary, potentially enhancing link prediction. However, the specific contribution of this geometric property requires further ablation studies.

In contrast, D-Mercator has a slightly lower AUC of 0.9707 at low missing rates. This is primarily because D-Mercator employs a multidimensional hyperbolic space (D+1 dimensions) for modeling, requiring the balancing of more parameters (e.g., hidden degrees and inverse temperature). The multidimensional space necessitates simultaneous optimization of the radial coordinates (popularity) and angular coordinates (similarity) of nodes. At low missing rates, minor noise can disrupt the fine-tuning of local community structures. However, by iteratively adjusting hidden degrees and remapping angular coordinates, D-Mercator partially offsets the impact of data loss. As a result, its AUC decline is the smallest, dropping only 0.43% from a missing link rate of 10% to 30%, the lowest among the three methods.



(a) D-Mercator lose10% (b) Lorentz lose10% (c) Poincare lose10%

(d) D-Mercator lose20% (e) Lorentz lose20% (f) Poincare lose20%

(g) D-Mercator lose30% (h) Lorentz lose30% (i) Poincare lose30%

Figure 4.2: These nine figures represent the Precision-Recall (PR) curves for three hyperbolic embedding methods—D-Mercator, Lorentz, and Poincare—with missing node pairs corresponding to 10%, 20%, and 30% recall rates, respectively.

Next, we present the results of the PR (Precision-Recall) curves and related metrics such as AUPR (Area Under the Precision-Recall Curve). The AS (Autonomous System) dataset used in our experiments contains 27,027 actual links and a total of 10,259 nodes. When forming all possible pairs of nodes, this results in more than 50 million pairs of nodes. If we follow the conventional approach for link prediction by computing and ranking all unconnected node pairs to generate the PR curve, we encounter a severe class imbalance between positive and negative examples. According to the definition of Precision in Equation (3.4), when the number of negative samples vastly exceeds that of positive ones, even a small number of false positives (FP) can significantly inflate the denominator. As a result, even if all true positives (TP) are correctly predicted, the overall Precision value remains low. This, in turn, negatively affects the PR curve and AUPR metrics. Our experimental observations confirm that the AUPR results are indeed poor under such extreme imbalance.

To address this issue, we sample from the negative examples to maintain a reasonable balance between positive and negative samples. Specifically, the following PR curves and AUPR results are obtained based on a 1:100 positive-to-negative sample ratio. Grover and Leskovec (2016) [49] and Zhang and

Chen (2018) [50] also adopted a 1:100 negative sampling ratio in their evaluations of link prediction tasks, demonstrating that this setting can still effectively reflect the predictive performance of models.

Similar to the ROC curves above, the PR curves here also consist of nine plots, showcasing the Precision-Recall (PR) curves of three hyperbolic embedding methods for missing link rates of 10%, 20%, and 30%. Each row corresponds to a missing link rate and contains three plots. The horizontal axis represents Recall, while the vertical axis represents Precision. The closer the curve is to the top-right corner, the better the model's predictive performance. Random guessing corresponds to a horizontal line parallel to the Recall axis, where the Precision is approximately equal to the ratio of the number of connected pairs to the total number of node pairs. In the figure, the "chance" value represents this baseline performance of random guessing; PR curves above the baseline suggest that the model's predictive performance is superior to random guessing.

In Figure 4.2, all models exhibit a general trend: precision gradually decreases as recall increases. This is a common phenomenon in link prediction tasks, as higher recall regions typically involve predictions with lower confidence. The corresponding evaluation metric, AUPR (Area Under the Precision-Recall Curve), consistently shows that the D-Mercator method achieves the highest values across all missing data scenarios, followed by Poincare, and then Lorentz.

Specifically, under the 10% missing rate scenario, the AUPR of D-Mercator reaches 0.6113, higher than 0.6096 for Poincare and 0.5414 for Lorentz. This trend remains consistent under the 20% and 30% missing rates. Moreover, across all three missing ratios (10%, 20%, and 30%), the D-Mercator model demonstrates the smallest decline in precision, indicating that it exhibits greater robustness and consistency in the link prediction task. Even under higher levels of missing data, D-Mercator continues to achieve strong predictive performance.

In contrast, the PR curves of the Lorentz and Poincare models exhibit noticeable fluctuations or sharp drops in their early segments (i.e., low-recall regions) under the 10% and 20% missing link scenarios. Specifically, precision drops rapidly at the very beginning as recall starts to increase, before gradually stabilizing. This phenomenon suggests that the models produce a relatively high false positive rate among the top-ranked predictions, thereby reducing initial precision. A possible explanation is that both Lorentz and Poincare rely on optimization-driven embedding methods, making them more sensitive to missing links—especially in areas where the local network structure changes significantly. This sensitivity can lead to inaccurate estimations of inter-node distances, which in turn affects the ranking of high-confidence prediction samples.

4.4 Rank Correlation And Pearson Coefficient Results

We now investigate how much correlated are the hyperbolic node distances across the three different embedding methods. We now present the results of the embedded rank correlation coefficient (Spearman) and Pearson correlation coefficient generated by the three methods, as well as scatter plots. Initially, after embedding the original graph dataset into a hyperbolic coordinate system using the three methods, the hyperbolic distances of node pairs were directly calculated from the original graph dataset. Subsequently, the distance correlation coefficient was calculated. We observed that the values of distance correlation coefficients were very low, which may be attributed to the inclusion of all node pairs - particularly low-

degree nodes whose coordinate inference tends to be noisier. This indicates that directly comparing the hyperbolic distances between node pairs cannot effectively capture whether there is a linear or nonlinear relationship between embeddings of different methods.

To further investigate whether linear or nonlinear relationships exist, we separately calculated the hyperbolic distances for node pairs with degrees exceeding 6, 10, and 20 (i.e., nodes with more than 6, 10, or 20 neighbors) and then computed the rank correlation coefficients and Pearson correlation coefficients. This step was primarily aimed at filtering out nodes with low degrees, i.e., peripheral nodes, and exploring whether the three methods exhibit common structural characteristics when embedding high-degree core nodes into hyperbolic space.

Table 4.3: Correlation Coefficients for Different Degrees

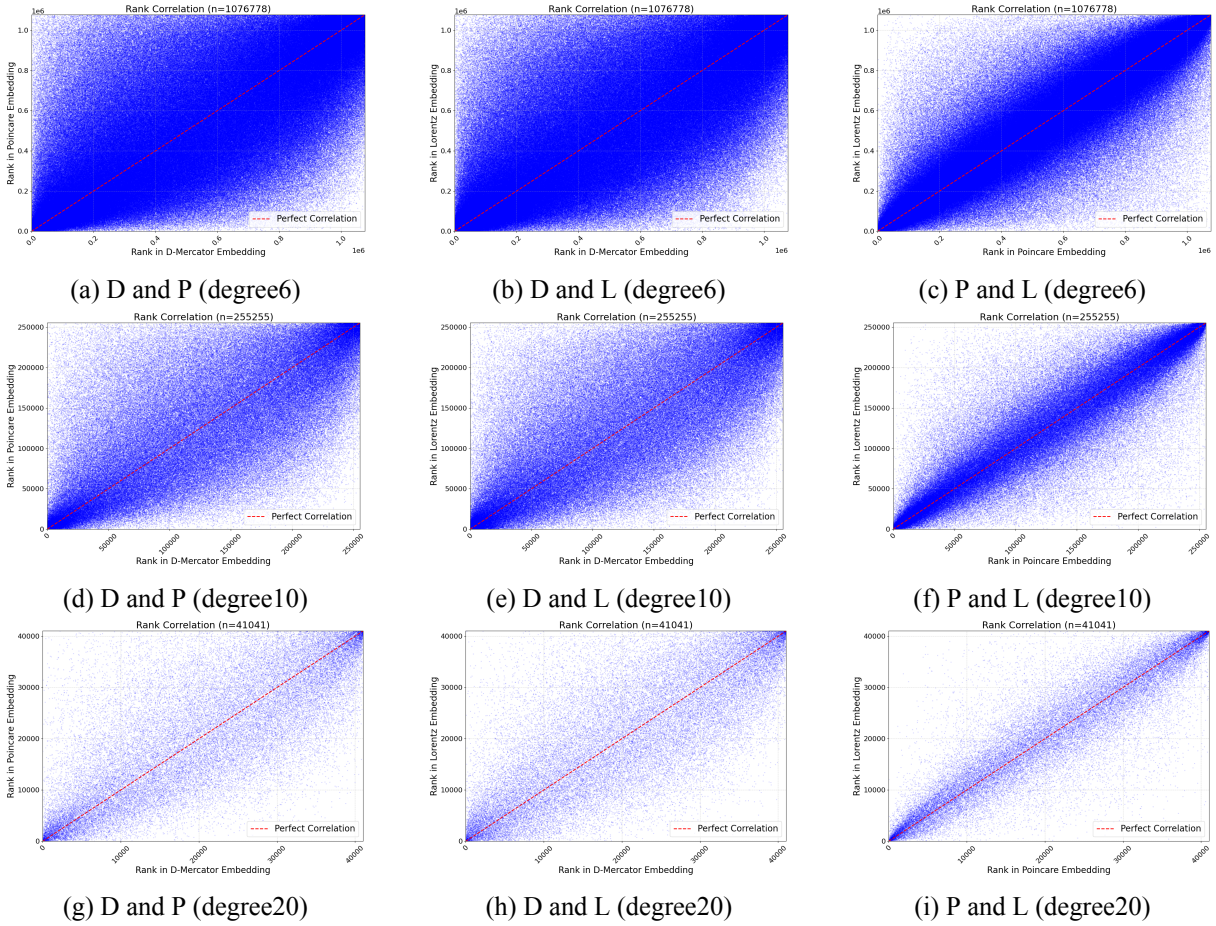
Degree	Model	Spearman (rank coefficient)	Pearson
6	D-Mercator and Poincare	0.6710	0.7114
	D-Mercator and Lorentz	0.6878	0.7196
	Lorentz and Poincare	0.8870	0.8983
10	D-Mercator and Poincare	0.7050	0.7440
	D-Mercator and Lorentz	0.7241	0.7509
	Lorentz and Poincare	0.8962	0.9081
20	D-Mercator and Poincare	0.7622	0.7850
	D-Mercator and Lorentz	0.7729	0.7811
	Lorentz and Poincare	0.9068	0.9152

From the results in Table 4.3, it is evident that the embedding correlation for high-degree nodes is significantly higher than that for low-degree nodes. Specifically, the Pearson coefficient between Lorentz and Poincare for high-degree nodes reaches 0.915, and the Spearman coefficient is 0.9068, indicating a strong consistency in capturing the embeddings of core nodes using these two methods. In contrast, the Pearson coefficient between D-Mercator and Poincare for low-degree nodes is 0.7114, and the Spearman coefficient is 0.6710, showing a noticeably lower correlation. The overall trend reveals that as the node degree increases, the correlation coefficients for all method pairs exhibit an upward trend, suggesting that the embedding results for high-degree nodes tend to converge across different methods, while the embedding results for low-degree nodes show greater variability due to differences in method assumptions.

One possible explanation for the higher correlation among high-degree nodes is their central position in the network topology. These nodes often play a dominant role in shaping hierarchical relationships and global structures. As a result, different embedding methods may converge in their representations of these nodes, leading to more consistent embeddings. In contrast, low-degree nodes, typically residing on the network periphery, are less structurally constrained and may be embedded differently depending on each model's assumptions and optimization strategy.

Additionally, Lorentz and Poincare embeddings consistently show the highest pairwise correlations among all model combinations. This may be attributed to their shared foundation in hyperbolic geometry, which could result in similar interpretations of network structure. Notably, their embedding results are especially close for high-degree nodes, as reflected in the Pearson and Spearman coefficients. Interestingly, D-Mercator also shows relatively high correlations with them, despite relying on different modeling as-

sumptions. These results suggest that the three methods exhibit a certain degree of alignment, particularly when representing structurally prominent nodes.



(a) D and P (degree6) (b) D and L (degree6) (c) P and L (degree6)

(d) D and P (degree10) (e) D and L (degree10) (f) P and L (degree10)

(g) D and P (degree20) (h) D and L (degree20) (i) P and L (degree20)

Figure 4.3: Three rows of pairwise rank scatter plots for degrees 6, 10, and 20 across the three embedding methods.

Due to the different spatial mapping scales of the three hyperbolic embedding methods, even if the same dataset is embedded into the hyperbolic space, the hyperbolic distance between the same node pairs in the dataset cannot be directly compared. So, we calculate the rank correlation coefficient and draw a rank scatter plot, which calculates the relative distance between nodes. Because we can explore whether different hyperbolic embedding methods have correlation in capturing the structural characteristics of the same dataset by obtaining the hyperbolic distance ranking of node pairs in various spaces and comparing the ranking of the same node in three hyperbolic spaces. Figure 4.3 presents the pairwise rank scatter plots for the AS dataset under the three embedding methods, where higher consistency in rank positions indicates greater similarity in structural representation.

Figure 4.3 shows the rank correlation scatter plots of pairwise embedding results for three methods (D-Mercator, Poincare and Lorentz) for node pairs with degrees larger than 6, 10, and 20, respectively. Each row contains three graphs, showing the rank correlation scatter plots of D-Mercator and Poincare, D-Mercator and Lorentz, and Poincare and Lorentz, respectively.

In the scatter plots, the horizontal and vertical axes represent the ranks of node pairs in the two embedding methods, respectively. The diagonal line indicates perfect correlation, meaning that the rankings

of the two methods are completely consistent. We can observe that most points are distributed around the line $y = x$. Among the methods, Lorentz and Poincare exhibit the highest rank correlation across all degree thresholds, especially for high-degree nodes (core nodes), where their embedding results are almost identical. In contrast, D-Mercator shows moderate correlation with the other methods, which gradually improves as the degrees of nodes increase. Overall, the embeddings of high-degree nodes demonstrate strong consistency across different methods, while the embeddings of low-degree nodes (peripheral nodes) are more susceptible to the choice of embedding method.

4.5 Summary

In this chapter, we conducted a detailed analysis and comparison of the performance of three hyperbolic embedding methods (D-Mercator, Poincare, and Lorentz) on autonomous system (AS) datasets through a series of experiments. The experiment covers multiple aspects such as greedy routing, missing link prediction, and rank correlation and Pearson correlation of node embedding. Through these experiments, we have drawn the following main conclusions:

- **Greedy Routing Performance** In the greedy routing task, the Lorentz model demonstrates the best performance, achieving the highest success rate and the lowest average hop count. As discussed in Section 4.2, its relatively strong performance may be attributed to its ability to more effectively represent hierarchical proximity. The Poincare model ranks second, with success rates and hop counts close to those of Lorentz, though slightly lower. Its two-dimensional disk model effectively captures both local and hierarchical structures, but numerical instability near the boundary may lead to some loss in routing efficiency, particularly in high-dimensional or complex networks. The D-Mercator model ranks third, primarily because its design emphasizes global structural optimization, making it less sensitive to local connection patterns. While this approach preserves overall topological consistency, it results in some performance trade-offs in routing tasks that rely heavily on local information.
- **Missing Link Prediction** The results of the missing link prediction task demonstrate that all three hyperbolic embedding methods(D-Mercator, Lorentz, and Poincare)achieve strong overall performance, as indicated by high ROC curves and AUC values across varying missing link rates (10%, 20%, and 30%). Although the AUC scores of all methods are close to 1, Lorentz and Poincare consistently achieve slightly higher AUC values than D-Mercator, suggesting a marginal advantage in preserving predictive signal under moderate link sparsity. Lorentz, in particular, benefits from efficient Riemannian optimization and a closed-form geodesic formula, which may contribute to its stability and performance in low-dimensional settings.

  However, when evaluated using Precision-Recall (PR) curves and AUPR, the performance ranking changes. D-Mercator outperforms the other two methods across all missing rates, achieving the highest AUPR and demonstrating the smallest drop in precision across recall levels. Its PR curves remain more stable, even under high levels of missing data, highlighting its robustness and consistency in predictive tasks. This may be due to its design, which emphasizes global structural preservation through a multidimensional hyperbolic embedding framework.

  In contrast, both Lorentz and Poincare exhibit a steep decline in precision at the early stages of

their PR curves, especially at 10% and 20% missing rates. This suggests a higher false positive rate among top-ranked predictions, potentially caused by increased sensitivity to local structure disruptions in their optimization-driven embeddings. Although they recover and stabilize at higher recall values, the initial drop affects their overall AUPR scores and highlights potential limitations in link prediction when high-confidence estimation is required.

Taken together, these results suggest that while all three models are capable of robust link prediction under missing data conditions, D-Mercator offers better stability and overall ranking performance, particularly in imbalanced scenarios where precision is critical.

- **Rank Correlation and Pearson Correlation** For high-degree nodes, the embedding results of the three methods exhibit high correlation, especially between the Lorentz and Poincare models, indicating strong consistency in capturing the embeddings of core nodes. This is because both Lorentz and Poincare models rely on the inherent properties of hyperbolic geometry to efficiently model hierarchical structures, and they are essentially different parameterized forms of hyperbolic space that can be transformed into each other through differential homeomorphism mapping. The mathematical equivalence leads to geometric consistency in the embedding space. They all sample Riemann optimization methods, so they have convergence similarity in manifold structure.

  In contrast, for low-degree nodes, the correlation between methods is significantly lower. This reflects greater variability in how peripheral nodes are represented, likely due to differences in how each method models local structures. D-Mercator, for instance, optimizes in a multidimensional hyperbolic space and involves simultaneous estimation of radial (popularity) and angular (similarity) coordinates. This global modeling perspective might reduce sensitivity to fine-grained local connectivity patterns. On the other hand, while Lorentz and Poincare also preserve hierarchy, their optimization procedures appear to yield less consistent embeddings for low-degree nodes across methods, potentially due to noise amplification or reduced structural constraints at the network periphery.

  These observations highlight a general trend: as node degree increases, the structural roles of nodes become more pronounced and consistently represented, resulting in higher embedding agreement across different methods. Conversely, low-degree nodes—being more sparsely connected—are more prone to divergence in embeddings, reflecting the influence of method-specific assumptions.

Overall, the overall trend reflected by the above experimental results is that as the degree of nodes increases, the embedding results of the three methods tend to be consistent, indicating that the dominant position of high degree nodes in the network topology makes different methods exhibit high similarity in capturing the embeddings of these nodes. The embedding results of low degree nodes show significant differences due to different methods, mainly because these nodes are located at the edge of the network, with diverse and irregular local connection patterns.

In general, the Lorentz model performs best in greedy routing tasks, making it suitable for scenarios that require efficient routing, such as Internet routing optimization. In complex Internet topologies, the Lorentz model can effectively plan packet routes, reduce hop counts and latency, and improve overall network transmission efficiency. The D-Mercator model demonstrates the best overall performance in missing link prediction, making it more suitable for tasks with high demands on global topology, such

as Internet topology modeling. D-Mercator is capable of accurately capturing the global structure of the Internet, making it ideal for constructing global Internet maps and assisting network engineers in understanding and optimizing large-scale network layouts. The Poincare model exhibits stable performance across multiple tasks. Although it ranks slightly below Lorentz in greedy routing and below D-Mercator in missing link prediction, its overall performance across both tasks is well-balanced. Moreover, it shows a high degree of embedding consistency with Lorentz for high-degree nodes. With its robustness and ability to represent hierarchical structures, the Poincare model is well-suited for complex network analysis tasks that require stable embeddings and effective hierarchical representation.

5 Future Work

Although the current study conducts a comprehensive set of experiments—covering greedy routing, missing link prediction, and correlation analysis—on the Autonomous System (AS) network, relying on a single network type may limit the generalizability of the findings. As a primary direction for future work, it is essential to replicate the same comparative experiments across diverse types of complex networks, to thoroughly evaluate the adaptability and robustness of the three representative embedding methods: D-Mercator, Poincare, and Lorentz. For example, citation networks tend to exhibit strong hierarchical structures and sparse connectivity, collaboration networks often show dense community structures, and infrastructure networks like power grids or transportation systems are typically characterized by low redundancy and high structural constraints. Applying the same evaluation pipeline to such structurally diverse networks would enable a more comprehensive assessment of each model's strengths and limitations, and provide valuable guidance for selecting embedding methods across different domains.

In addition to expanding the evaluation to a broader range of network types, another important future direction is to include more hyperbolic embedding models from both the network science and machine learning communities, to enable a more comprehensive comparison between the two methodological paradigms. The current study focuses on representative models, D-Mercator for network science and Poincare and Lorentz for machine learning. But these do not fully capture the diversity of approaches within each category. For example, models such as HyperMap, H2, and HyperTree represent alternative network science strategies, while deep learning-based methods such as HGCN, HGNN, or Hydra exemplify recent developments in machine-learned hyperbolic embeddings. Including a broader set of models will not only improve the robustness of conclusions, but may also help uncover systematic differences in how each methodological family captures network structure.

In addition to the comparison of static embedding models, another key direction is to explore the potential of hyperbolic embeddings in dynamic graph modeling. Existing hyperbolic embedding methods primarily focus on static networks, whereas real-world network topologies often evolve dynamically over time. For instance, in Autonomous System (AS) networks, nodes and connections frequently change due to routing policy adjustments, hardware failures, or malicious attacks. Traditional static embedding methods struggle to capture these dynamic properties in real-time. Additionally, domains such as social networks (with real-time updates to user relationships) and biological networks (involving dynamic reconstruction of protein interactions) necessitate dynamic modeling of graph data. Exploring the potential of hyperbolic space for processing dynamic graph data thus holds significant practical value. Future research could delve into leveraging hyperbolic neural networks to efficiently model dynamically updated graph data, enabling adaptive embeddings that reflect temporal evolution while preserving hierarchical and geometric properties inherent to hyperbolic representations.

BIBLIOGRAPHY

[1] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *arXiv preprint arXiv:1709.05584*, 2017.

[2] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE transactions on knowledge and data engineering*, vol. 31, no. 5, pp. 833–852, 2018.

[3] B. P. Chamberlain, J. Clough, and M. P. Deisenroth, "Neural embeddings of graphs in hyperbolic space," *arXiv preprint arXiv:1705.10359*, 2017.

[4] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He, "A survey on knowledge graph-based recommender systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 8, pp. 3549–3568, 2020.

[5] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguná, "Hyperbolic geometry of complex networks," *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, vol. 82, no. 3, p. 036106, 2010.

[6] A. Y. Nur and M. E. Tozal, "Identifying critical autonomous systems in the internet," *The Journal of Supercomputing*, vol. 74, no. 10, pp. 4965–4985, 2018.

[7] F. Papadopoulos, D. Krioukov, M. Boguná, and A. Vahdat, "Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces," in *2010 Proceedings IEEE Infocom*. IEEE, 2010, pp. 1–9.

[8] C. Qian and S. S. Lam, "Greedy routing by network distance embedding," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2100–2113, 2015.

[9] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: statistical mechanics and its applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[10] S. Khoshraftar and A. An, "A survey on graph representation learning methods," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 1, pp. 1–55, 2024.

[11] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo, "Ripplenet: Propagating user preferences on the knowledge graph for recommender systems," in *Proceedings of the 27th ACM international conference on information and knowledge management*, 2018, pp. 417–426.

[12] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[13] A. B. Adcock, B. D. Sullivan, and M. W. Mahoney, "Tree-like structure in large social and information networks," in *2013 IEEE 13th international conference on data mining*. IEEE, 2013, pp. 1–10.

[14] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," *Advances in neural information processing systems*, vol. 30, 2017.

[15] ——, "Learning continuous hierarchies in the lorentz model of hyperbolic geometry," in *International conference on machine learning*. PMLR, 2018, pp. 3779–3788.

[16] A. Tifrea, G. Bécigneul, and O.-E. Ganea, "Poincar\'e glove: Hyperbolic word embeddings," *arXiv preprint arXiv:1810.06546*, 2018.

[17] I. Chami, Z. Ying, C. Ré, and J. Leskovec, "Hyperbolic graph convolutional neural networks," *Advances in neural information processing systems*, vol. 32, 2019.

[18] M. Kitsak, I. Voitalov, and D. Krioukov, "Link prediction with hyperbolic geometry," *Physical Review Research*, vol. 2, no. 4, p. 043113, 2020.

[19] R. Jankowski, A. Allard, M. Boguñá, and M. Á. Serrano, "The d-mercator method for the multidimensional hyperbolic embedding of real networks," *Nature Communications*, vol. 14, no. 1, p. 7585, 2023.

[20] B. Wilson and M. Leimeister, "Gradient descent in hyperbolic space," *arXiv preprint arXiv:1805.08207*, 2018.

[21] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," *Advances in neural information processing systems*, vol. 29, 2016.

[22] S. T. Smith, "Optimization techniques on riemannian manifolds," *arXiv preprint arXiv:1407.5965*, 2014.

[23] L. Campbell, *Historical linguistics*. Edinburgh University Press, 2013.

[24] F. Papadopoulos, C. Psomas, and D. Krioukov, "Network mapping by replaying hyperbolic growth," *IEEE/ACM Transactions on Networking*, vol. 23, no. 1, pp. 198–211, 2014.

[25] S. Purcell, "Maximum likelihood estimation," *Accessed 05Jun2015. Available at: http://statgen. iop. kcl. ac. uk/bgim/mle/sslike_3. html*, 2007.

[26] K. M. Harris, "The add health study: Design and accomplishments," 2013.

[27] R. Jankowski, P. Hozhabrierdi, M. Boguñá, and M. Á. Serrano, "Feature-aware ultra-low dimensional reduction of real networks," *npj Complexity*, vol. 1, no. 1, p. 13, 2024.

[28] D. Krioukov, F. Papadopoulos, A. Vahdat, and M. Boguná, "Curvature and temperature of complex networks," *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, vol. 80, no. 3, p. 035101, 2009.

[29] M. E. Newman, "Power laws, pareto distributions and zipf's law," *Contemporary physics*, vol. 46, no. 5, pp. 323–351, 2005.

[30] H. W. Lin and M. Tegmark, "Critical behavior in physics and probabilistic formal languages," *Entropy*, vol. 19, no. 7, p. 299, 2017.

[31] K. Katayama and E. W. Maina, "Indexing method for hierarchical graphs based on relation among interlacing sequences of eigenvalues," *Journal of information processing*, vol. 23, no. 2, pp. 210–220, 2015.

[32] F. Papadopoulos, M. Kitsak, M. Á. Serrano, M. Boguná, and D. Krioukov, "Popularity versus similarity in growing networks," *Nature*, vol. 489, no. 7417, pp. 537–540, 2012.

[33] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[34] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[35] V. Giotsas and S. Zhou, "Inferring internet as relationships based on bgp routing policies," *arXiv preprint arXiv:1106.2417*, 2011.

[36] S. Helgason, *Differential geometry, Lie groups, and symmetric spaces*. Academic press, 1979.

[37] J. M. Lee, *Manifolds and differential geometry*. American Mathematical Society, 2022, vol. 107.

[38] P. Petersen, "Hypersurfaces," *Riemannian Geometry*, pp. 95–109, 2006.

[39] M. Boguna, D. Krioukov, and K. C. Claffy, "Navigability of complex networks," *Nature Physics*, vol. 5, no. 1, pp. 74–80, 2009.

[40] M. Boguná, F. Papadopoulos, and D. Krioukov, "Sustaining the internet with hyperbolic mapping," *Nature communications*, vol. 1, no. 1, p. 62, 2010.

[41] J. M. Kleinberg, "Navigation in a small world," *Nature*, vol. 406, no. 6798, pp. 845–845, 2000.

[42] M. Boguná and D. Krioukov, "Navigating ultrasmall worlds in ultrashort time," *Physical review letters*, vol. 102, no. 5, p. 058701, 2009.

[43] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *Proceedings of the twelfth international conference on Information and knowledge management*, 2003, pp. 556–559.

[44] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[45] C. Spearman, "The proof and measurement of association between two things." 1961.

[46] W. J. Conover, *Practical nonparametric statistics*. john wiley & sons, 1999.

[47] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1-2, pp. 81–93, 1938.

[48] M. G. Kendall and A. Stuart, "Functional and structural relationship," *The advanced theory of statistics*, vol. 2, pp. 399–343, 1979.

[49] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[50] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

APPENDICES

APPENDIX I

Parameter settings

To ensure the reproducibility of the results in this study, we provide the official implementation link of the embedding models used: https://github.com/facebookresearch/poincare-embeddings (Nickel and Kiela, Poincare Embeddings for Learning Hierarchical Representations, 2017). It is worth noting that the Lorentz model is also included in this repository. Specifically, the implementation of the Lorentz manifold can be found in the hype/manifolds directory of the project. This study is based on that version to conduct the hyperbolic space embedding experiments.

For both models, we input the AS-level Internet topology dataset in `.csv` format into the embedding scripts. Minor adjustments were made to the input data format to meet the requirements of each framework, but the core code remained unchanged. All modified hyperparameters used during the embedding stage are provided below; all other parameters were kept at their default values. It is worth noting that, apart from the model type and checkpoint path, most of the parameter settings for the two models are identical. The output of the embedding is a `.pth` file containing node IDs and their corresponding coordinates. After format conversion, the resulting hyperbolic coordinates of each node are directly used for downstream tasks such as greedy routing and correlation analysis, with no additional post-processing required.

Table I.1: Embedding hyperparameters used for the Poincare model

Parameter	**Value**
Learning Rate	0.1
Negative Samples	100
Neg. Multiplier	1.8
Epochs	5000
Burn-in	100
Margin	0.2
Embedding Dimension	3
Batch Size	64
Evaluation Frequency	1 (every epoch)
Learning Rate Type	Constant
Max Norm	100000
Model Type	Distance-based
Manifold	Poincare ball
Dataset	l_lose10.csv
Checkpoint Filename	poin-lose10-5k.pth
Training Threads	2
Number of Processes	2
Fresh Start	Yes (fresh flag)
Sparse Optimization	Enabled (sparse)
Evaluation Task	Reconstruction
Symmetric	Enabled (sym)

The parameters listed in the table define the embedding configuration used throughout the training process. Specifically, the **Learning Rate** determines the step size for optimization updates, while **Negative Samples** and **Neg. Multiplier** control the sampling and weighting of negative examples during contrastive learning. **Epochs** refers to the total number of complete passes over the training data, and **Burn-in** specifies the initial stabilization period. The **Margin** is used in the loss function to separate positive and negative samples. **Embedding Dimension** defines the dimensionality of the hyperbolic space, and **Batch Size** indicates the number of samples processed at once. **Evaluation Frequency** determines how often the model's performance is evaluated during training, while **Learning Rate Type** specifies whether the learning rate remains constant. **Max Norm** constrains the norm of embeddings to prevent numerical instability. **Model Type** and **Manifold** define the structure and geometry of the embedding space, respectively. **Dataset** and **Checkpoint Filename** identify the data and save paths. **Training Threads** and **Number of Processes** control parallelism during data loading and model updates. **Fresh Start** indicates that training begins from scratch, **Sparse Optimization** enables memory-efficient operations, and **Evaluation Task** specifies the task used for validation. Finally, **Symmetric** indicates whether the graph is treated as undirected.

Table I.2: Embedding hyperparameters used for the Lorentz model

Parameter	**Value**
Dimension	3
Learning Rate	0.1
Epochs	5000
Negative Samples	100
Burn-in	100
Data Loading Processes	1
Model	Distance
Manifold	Lorentz
Dataset	l_lose30.csv
Checkpoint Filename	lorentz-lose30-5k.pth
Batch Size	64
Evaluation Frequency	1 (per epoch)
Fresh Start	Yes
Sparse Optimization	Enabled
Training Threads	1
Learning Rate Type	Constant
Negative Sample Multiplier	1.8
Max Norm	100000
Margin	0.2
Symmetric	Yes
Evaluation Task	Reconstruction
GPU	-1 (disabled)