# Tradeoffs Between Embeddings in Different Models of the Hyperbolic Space

**Eric Lou**
erlou@stanford.edu

**Shawn Zhang**
szhang22@stanford.edu

**Ishan Gaur**
ishang@stanford.edu

## Abstract

Hyperbolic embeddings have achieved recent success in capturing hierarchical information (e.g. WordNet); however, these representations in hyperbolic space are complex. Accordingly, there is little theoretical understanding of the tradeoffs in using different models of hyperbolic space and how they are influenced by the properties of the graph they are applied to. In particular, there are four common models that we set out to better understand: the hyperboloid model, the Poincaré disk, the half-plane model, and the Beltrami-Klein model. Through experimentation with various synthesized graphs based on radius, height, and density and also graphs that model real-world data, we compare the running time, loss, distortion, and mean average precision for each of the four models when applied to the graphs. We see that the half-plane model and the Beltrami-Klein model perform similarly in terms of global distortion while only the half-plane model is able to keep local graph structure intact. We also see that the hyperboloid model and the Poincaré disk are optimal for Erdos-Renyi random graphs of density less than 0.75. Lastly, fractals and social network graphs have the least distortion when embedded in the half-plane space.

## 1 Introduction

Recent research has shown that hyperbolic embeddings significantly improve the performance of various tasks that benefit from hierarchical organization (3). One such category of tasks is natural language processing, as the text graphs better fit a curved, hyperbolic space than the standard Euclidean space (2) due to the negative curvature of the manifold which allows greater distances to be compacted into the same Euclidean space. This is also why balanced trees are a very natural embedding in these spaces and serve somewhat as a benchmark for the models in our work. By doing gradient descent on the embedded vectors, we output the Euclidean coordinates of the edges in the hyperbolic embedding. We then are able to calculate the amount of distortion of the data, along with other metrics such as running time, loss, and mean average precision. The four hyperbolic models that we seek to better understand are the hyperboloid, Poincaré ball (referred to as Poincaré), Poincaré half-plane (referred to as half-plane), and Beltrami-Klein (referred to as Klein) models. While these models are isomorphic to each other, our preliminary explorations show that they produce embeddings of different quality. By understanding the reason behind this phenomenon, we hope to produce some foothold for further theoretical work in this area and provide intuition for more effectively deploying these models in practice.

## 2 Related work

### 2.1 Optimization via Exponential Map

In order to translate the gradient, which is calculated in the ambient Euclidian space, for simplicity, it is common practice to use retractions, smooth maps from the ambient space to the manifold. These involve the use of a tangent plane along the surface of the space and projecting or otherwise adjusting the gradient vector onto a geodesic of the embedding. This can be done in ad hoc retractions or more formally as exponential maps. Ad hoc methods (simply referred to as retractions from here on) are generally faster but tend to be close enough to the more accurate exponential maps. Retraction involves scaling and rotating the tangent plane such that the update point lies on the space. Exponential maps incorporate information about the geodesics in the hyperbolic space to ensure that the update point lies on the shortest path from the current point to the optimal point. Research has shown different ways to create exponential maps for the various hyperbolic spaces (8). In our research, we will implement exponential maps for all four models but the Poincaré model as the retraction is generally accurate for Poincaré and the exponential map is expensive to compute.

### 2.2 Poincaré Performance

Previous research has laid the foundation for testing the Poincaré model of hyperbolic space. Efficient algorithms have already been developed to learn the Poincaré embedding given a Euclidean graph (2). It has been shown that for datasets like WordNet, the Poincaré embedding is able to successfully organize words by both similarity and hierarchy (6), outperforming their counterpart Euclidean embeddings. Both word embeddings and sentence embeddings were tested under the Poincaré model; mean average precision was calculated to be 0.857 while Euclidean embeddings only had a mean average precision of 0.162 (3). Others have explored the precision-dimensionality tradeoff on various implementations of the Poincaré model; the same researchers created a PyTorch program for Poincaré embeddings, which we will use as a basis to develop the remaining models (7).

Unlike previous research which compares hyperbolic embeddings with the hyperboloid and Poincaré models to Euclidean and spherical embeddings, we implement the half-plane and Beltrami-Klein models on top of the hyperboloid and Poincaré models. With all four of these models, we are able to compare their performances against each other.

## 3 Dataset and Features

To create our dataset of edge sets, we synthesized 8 knowledge graphs with NetworkX by choosing parameters for tree radius, tree height, and/or graph density. These graphs fall into 4 broad categories: balanced trees, density graphs, social network graphs, and Sierpinski graphs.

**Balanced Trees.** We create 2 trees – a tall tree with a radius of 3 and height of 5 and a wide tree with a radius of 5 and height of 3.

**Density Graphs.** We create 4 Erdos-Renyi random graphs of densities 0.25, 0.5, 0.75, and 1. For a density of 0.25, any pair of nodes has a 0.25 chance of having an edge.

**Social Network graphs.** We use a graph that represents the coappearances from Les-Mis as a model for a small social network graph.

**Sierpinski Graphs.** We create a strongly connected Sierpinski fractal with a depth of 5 as a model for highly-structured, cyclical data.

Samples of these graphs can be seen in the figure below.

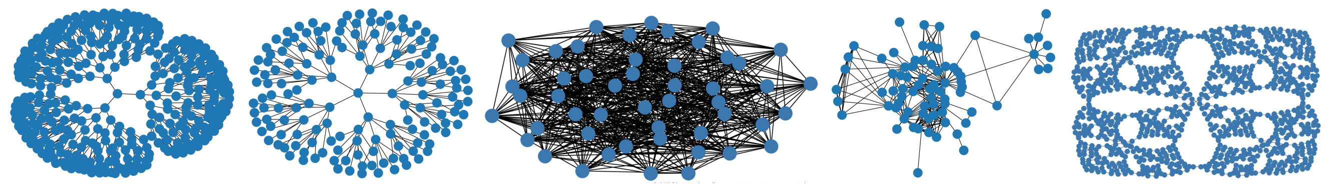

Figure 1: From left to right: (1) balanced tree with radius 3, height 5 (2) balanced tree with radius 5, height 3 (3) Erdos-Renyi 0.5 density graph with 50 nodes (4) Les-Mis coappearances (5) Sierpinski triangle fractal

## 4 Methods

### 4.1 Hyperbolic Spaces

In this paper, the four models of hyperbolic space that we analyze are the hyperboloid model, the Poincaré disk, the half-plane model, and the Beltrami-Klein model. Each model has a domain definition (to map points onto), a distance function (to measure distortion), and an associated Riemannian metric (to update gradients). Precise descriptions for all of these can be found in the Appendix.

### 4.2 Isometries

To project between different models in hyperbolic space, we use an intermediary space $J$ (the Hemisphere model). We start at the Euclidean coordinates, and then map to Hyperboloid (L). From there, we can reach every other space.

$$E \to L, \quad (x_1, \ldots, 1) \mapsto (\sqrt{1 + x_1^2 + \cdots + x_n^2}, x_2, \ldots, x_n, 1)$$
$$J \to H, \quad (x_1, \ldots, x_{n+1}) \mapsto (1, 2x_2/\left(x_1 + 1\right), \ldots, 2x_{n+1}/\left(x_1 + 1\right))$$
$$J \to P, \quad (x_1, \ldots, x_{n+1}) \mapsto (x_1/\left(x_{n+1} + 1\right), \ldots, x_n/\left(x_{n+1} + 1\right), 0)$$

$K \to J, \quad (x_1, \ldots, x_n, 1) \mapsto \left(x_1, \ldots, x_n, \sqrt{1 - x_1^2 - \cdots - x_n^2}\right)$

$L \to J, \quad (x_1, \ldots, x_{n+1}) \mapsto (x_1/x_{n+1}, \ldots, x_n/x_{n+1}, 1/x_{n+1})$

## 5 Experiments/Results/Discussion

### 5.1 Metrics

There are three primary metrics we used for our analysis.

#### 5.1.1 Distortion

Distortion is a global average of the difference between shortest path distances in the graph and the distances on the embedded hyperbolic space. It is given by:

$$D(f) = \frac{1}{\binom{n}{2}} \left( \sum_{u,v \in U : u \neq v} \frac{|d_V(f(u), f(v)) - d_U(u,v)|}{d_U(u,v)} \right)$$

where $n$ is the number of nodes, $V$ is the embedding, $U$ is the graph with points $u, v$ and $f : U \to V$ is the map between the two spaces.

#### 5.1.2 MAP

MAP is mean average precision, which is a local metric, best described as an average proportion of the nearest points of a node which are actually its neighbors in the original graph. Here $f$ is the embedding map, $V$ is the vertex set of the graph, $a$ is the node from $V$ being examined, and $\mathcal{N}$ is the set of neighbors of the point. This neighbor set has points $b_i$ for which $R_{a,b_i}$ which is the set of points in the embedding that are at least as close as $b_i$ to $a$.

$$\text{MAP}(f) = \frac{1}{|V|} \sum_{a \in V} \frac{1}{\deg(a)} \sum_{i=1}^{|\mathcal{N}_a|} \frac{|\mathcal{N}_a \cap R_{a,b_i}|}{|R_{a,b_i}|}$$

#### 5.1.3 Loss

It has been demonstrated that distortion and MAP are not convex in general, so we use a modified notion of the distortion when training. This represents the loss, and it is given by the following:

$$\mathcal{L}(x) = \sum_{1 \leq i \leq j \leq n} \left| \left( \frac{d_{\mathcal{P}}(x_i, x_j)}{d_G(X_i, X_j)} \right)^2 - 1 \right|$$

where $G$ is the original graph, $P$ is the manifold the embedded points are placed in, and $d$ is the respective distance function.

### 5.2 Experiments

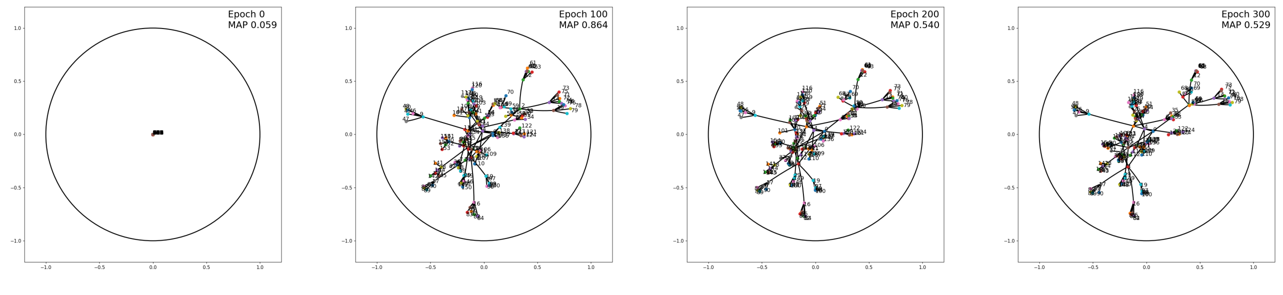

Figure 2: Visualization of a sample learning instance: Hyperboloid for balanced tree (radius 5, height 3).

We first ran preliminary experiments to find promising learning rates and dimensions for each model as they do not have standardized parameters. We created graphs to check for overfitting and underfitting, and continuously readjusted parameters until a stable fit was reached. With our range of learning rates and dimensions, we went through all of the experiments for each graph and model, and we chose the top performing model for each dimension and learning rate combination. One important reason to not pick the two independently is we wanted to see how far you could increase the learning rate (which increases training speed) while preventing failures at the edges due to retractions and preventing convergence errors at different dimensions. This is because, practically speaking, at higher dimensions, which is necessary to produce good embeddings for some graphs, it can take a lot longer to train, depending on the accuracy of the gradient updates. The top performing results are summarized in the tables below, and the transformation of a sample embedding over 300 epochs is shown above in Figure 2.

| Sierpinski: K=4, H=5 | | | | | | |
|---|---|---|---|---|---|---|
| **Model** | **Dim** | **Learning Rt** | **Loss** | **MAP** | **Distortion** | **WC** |
| Hyperboloid | 10 | 5 | 0.308715 | 0.748 | 0.4733 | 38.4042 |
| Poincare | 10 | 5 | 0.308715 | 0.748 | 0.4733 | 38.4042 |
| **Halfplane** | **10** | **5** | **0.306268** | **0.764** | **0.4684** | **38.453** |
| Klein | 10 | 0.1 | 0.583595 | 0.1828 | 0.6895 | 53.2123 |

| Social Graph: Les Miserables | | | | | | |
|---|---|---|---|---|---|---|
| **Model** | **Dim** | **Learning Rt** | **Loss** | **MAP** | **Distortion** | **WC** |
| Hyperboloid | 5 | 5 | 0.013329 | 0.8202 | 0.0869 | 2.1958 |
| Poincare | 5 | 5 | 0.013329 | 0.8202 | 0.0869 | 2.1958 |
| **Halfplane** | **10** | **5** | **0.009316** | **0.9176** | **0.0688** | **2.214** |
| Klein | 10 | 0.1 | 0.028229 | 0.9274 | 0.1339 | 2.8731 |

| Balanced Tree: R=3, H=5 | | | | | | |
|---|---|---|---|---|---|---|
| **Model** | **Dim** | **Learning Rt** | **Loss** | **MAP** | **Distortion** | **WC** |
| Hyperboloid | 10 | 5 | 0.12707 | 0.8811 | 0.2925 | 9.57 |
| Poincare | 10 | 5 | 0.12707 | 0.8811 | 0.2925 | 9.57 |
| **Halfplane** | **10** | **5** | **0.109614** | **0.8967** | **0.2754** | **9.3912** |
| Klein | 10 | 0.1 | 0.310093 | 0.7703 | 0.5353 | 11.8158 |

| Balanced Tree: R=5, H=3 | | | | | | |
|---|---|---|---|---|---|---|
| **Model** | **Dim** | **Learning Rt** | **Loss** | **MAP** | **Distortion** | **WC** |
| Hyperboloid | 10 | 10 | 0.004236 | 0.5888 | 0.0426 | 1.5709 |
| Poincare | 10 | 10 | 0.004236 | 0.5888 | 0.0426 | 1.5709 |
| **Halfplane** | **10** | **5** | **0.004787** | **0.8641** | **0.046** | **1.6317** |
| Klein | 10 | 0.1 | 0.143739 | 0.8915 | 0.3439 | 6.371 |

| Random Graph: 400 Nodes, P=0.25 | | | | | | |
|---|---|---|---|---|---|---|
| **Model** | **Dim** | **Learning Rt** | **Loss** | **MAP** | **Distortion** | **WC** |
| **Hyperboloid** | **5** | **5** | **0.003007** | **0.2579** | **0.0259** | **2.2029** |
| **Poincare** | **5** | **5** | **0.003007** | **0.2579** | **0.0259** | **2.2029** |
| Halfplane | 10 | 5 | 995.505658 | 0.2632 | 31.5376 | 1.2922 |
| Klein | 10 | 0.1 | 0.000832 | 0.2475 | 0.0184 | 1.4653 |

| Random Graph: 400 Nodes, P=0.50 | | | | | | |
|---|---|---|---|---|---|---|
| **Model** | **Dim** | **Learning Rt** | **Loss** | **MAP** | **Distortion** | **WC** |
| **Hyperboloid** | **10** | **5** | **0** | **0.5001** | **0.0001** | **1.0076** |
| Poincare | 5 | 5 | 0.001961 | 0.5049 | 0.0163 | 2.0725 |
| Halfplane | 10 | 5 | 1003.58405 | 0.5076 | 31.6572 | 1.1727 |
| Klein | 10 | 0.1 | 0.000472 | 0.494 | 0.0095 | 1.4156 |

| Random Graph: 400 Nodes, P=0.75 | | | | | | |
|---|---|---|---|---|---|---|
| **Model** | **Dim** | **Learning Rt** | **Loss** | **MAP** | **Distortion** | **WC** |
| **Hyperboloid** | **10** | **5** | **0** | **0.7526** | **0** | **1** |
| **Poincare** | **10** | **5** | **0** | **0.7526** | **0** | **1** |
| Halfplane | 10 | 5 | 1006.02325 | 0.7546 | 31.7221 | 1.1718 |
| Klein | 10 | 0.1 | 0.000239 | 0.7491 | 0.0051 | 1.4001 |

| Random Graph: 400 Nodes, P=1.00 | | | | | | |
|---|---|---|---|---|---|---|
| **Model** | **Dim** | **Learning Rt** | **Loss** | **MAP** | **Distortion** | **WC** |
| Hyperboloid | 10 | 5 | 0.000341 | 1 | 0.0028 | 1.397 |
| Poincare | 10 | 5 | 0.000341 | 1 | 0.0028 | 1.397 |
| **Halfplane** | **10** | **0.1** | **0.000276** | **1** | **0.0013** | **1.4486** |
| **Klein** | **10** | **0.1** | **0.000276** | **1** | **0.0013** | **1.4486** |

Figure 3: Table of best results for each model on all graphs.

## 5.3 Analysis

### 5.3.1 Preliminary Results

As expected, the two balanced trees ($R = 5$, $H = 3$ and $R = 3$, $H = 5$) performed well across the four models. By comparing the performances across the various experiments, it seems that the number of nodes in a graph has a large impact on the quality of the embedding. Moreover, graphs like the Erdos-Renyi random graphs, which are highly interconnected, might get better distortion results, because for a given node placed at the center, on average, a proportion $p$ of the entire vertex set is unit distance from the point. In contrast, if we take a balanced tree like $R = 3$, $H = 5$, two-thirds of the entire vertex set is distance 5 from the center. This leverages the negative curvature of the space well to solve the problem, but is still fundamentally harder to embed.

Moving on to an analysis of specific models, there was an interesting effect where the performances of Half-plane and Klein were very similar, and the performances of Hyperboloid and Poincaré were also similar. However, each pair seemed to perform well whenever the other pair did not. It is not yet clear why this happens, but we will discuss the implications of this here.

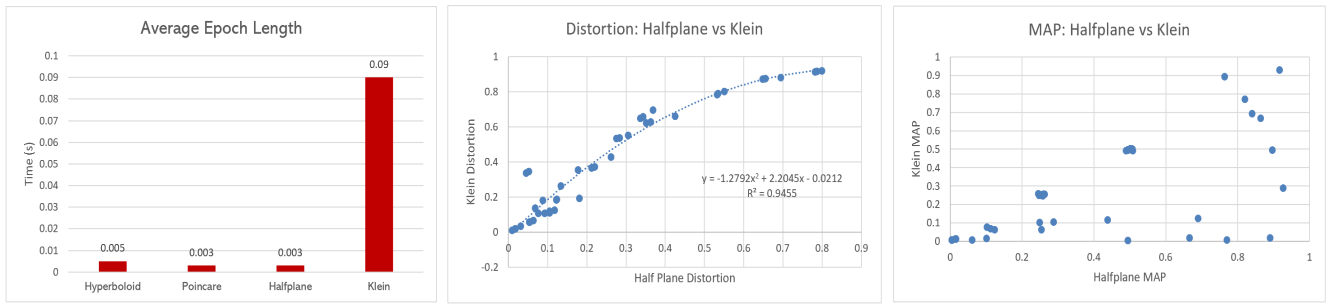

Figure 4: Model Gradient Update Lengths, Distortion and MAP Correlation Graphs: Half-plane and Klein

#### 5.3.2 Half-plane and Klein Tradeoffs

First, the lowest distortions produced by Half-plane and Klein's were correlated strongly across graphs and experiments on different settings, as seen in Figure 4's graph of Half-plane's distortion against Klein's distortion for the same settings and graph. However, even with very similar distortions, the MAP of Klein almost seemed to be bound by the MAP of Half-plane. Accordingly, this seems to be a problem with Klein itself, since no similar behavior was observed between Poincaré and Hyperboloid. Additionally, this unstable MAP for Klein indicates that it is not good at preserving local relationships among points in the graph. From this, we recommend that the use of Klein should be avoided for various flavors of social networks. On the other hand, this also means that Half-plane may be a fitting option for such applications.

Half-plane's weakness, however, seems to be its inability to represent dense graphs like the Erfos-Renyi graphs. This may be because it has an infinitely far edge passing through the origin; so, a random graph, for example, can easily tread too close and send mapped points infinitely far away. Contrary to this idea, we actually see that Half-plane performs well on fully connected graphs, but it is possible that the problem itself was so constrained that gradient updates were naturally small enough that no nodes were placed "infinitely" far away due to lack of precision.

Hyperboloid and Poincaré seem to average well among the four models consistently and do not often display edge errors like the one described above. Additionally, they perform better on the less dense random graphs. However, on all the other graphs, Half-plane performs better.

#### 5.3.3 Varying Distortions Under Equal MAP Values

Related to this, there is a seeming discrepancy between the MAP and distortion of some of the random graph trials where the MAP values are similar but the distortions vary widely. This makes sense because the MAP values will approximately match the probability $p$ (where $p = 0.25, 0.5, 0.75, 1$); this is the proportion of the nearest $k$ nodes any given node is connected to even at random. Due to this, the model may be overfitting, as distortion, which is more closely related to the loss, is being minimized at the cost of any meaningful embedding of local information.

#### 5.3.4 Dimension and Time Tradeoffs

Lastly, our results show that higher dimensions typically performs well, which conforms well with our understanding of higher dimensional spaces having higher capacity to more accurately embed arbitrary information. The only downside is that training becomes more unstable near the edges because approximation errors in the retractions become more important. Also, specifically for the Klein model, training in higher dimensional spaces is more expensive. The rest of the models have exponential maps and retractions that run in linear time with the dimension, but Klein's gradient update has order dimension squared for its time complexity.

## 6 Conclusion/Future Work

### 6.1 Conclusion

Although not commonly used in literature, we found that Half-plane was one of the most effective models, even though it can be hard to train. Klein was a slightly more stable model that often had similar distortions to Half-plane, but its MAP is often quite poor, which almost always makes Half-plane the better choice. On the other hand, if there is not enough compute available to do an in-depth parameter search, Hyperboloid and Poincare might be the better options. They tend to do almost as well as Half-plane but are much easier to train.

### 6.2 Future Work

The first thing we would do is extend our testing of these models on the graphs by putting them through prediction tasks on incomplete edge/node sets where the goal is for the embedding to be able to complete the missing information when asked to do link prediction. Additionally, we would test the data on larger real-world graphs and conduct experiments to find if there are any differences in the precision/accuracy tradeoff between each of our four models. Last, we would compare the performance of these models to spherical and Euclidean embeddings to give further theoretical justifications to our experimental results. This would include targeted work that deals with understanding the implications of a model being conformal or having one or more edges, etc.

## 7 Acknowledgments

We would like to thank Fred Sala and Adva Wolf for their feedback and mentorship.

## Contributions

All authors contributed an equal amount of work.
Eric Lou worked on implementation of the Poincaré, Half-Plane, and Beltrami-Klein model.
Ishan Gaur worked on researching the theory and analyzing the results, as well as implementating the Poincaré model.
Shawn Zhang worked on building the knowledge graphs, coding/running a script for experiments, and implementing the Half-Plane model.

## Code

The code for our four models, the compiled knowledge graphs, the experimental script, and the data files containing training logs can be viewed and downloaded on GitHub at the following link:
`https://github.com/ezl-13/hyperbolics`.

## Appendix

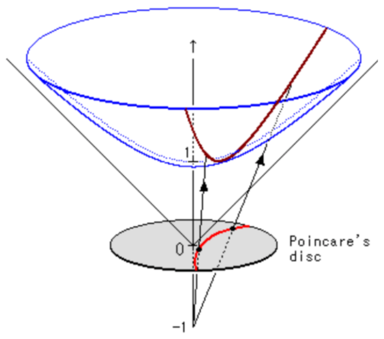

Figure 5: Demonstrating projection from hyperboloid to Poincaré (5).

In this paper, the four models of hyperbolic space that we analyze are the hyperboloid model ($L$), the Poincaré disk ($P$), the half-plane model ($H$), and the Beltrami-Klein model ($K$). Each model has a domain definition (to map points onto), a distance function (to measure distortion), and an associated Riemannian metric (to update gradients). This appendix provides Precise descriptions for all of these below.

### Hyperboloid Model

Domain: $\{(x_1, \ldots, x_n, x_{n+1}) : x_1^2 + \cdots + x_n^2 - x_{n+1}^2 = -1 \text{ and } x_{n+1} > 0\}$
Distance: $d(p, q) = \operatorname{arcosh}(p_0 q_0 - p_1 q_1 - \ldots - p_n q_n)$
Riemannian metric: $ds_L^2 = dx_1^2 + \cdots + dx_n^2 - dx_{n+1}^2$
The hyperboloid model is characterized by embedding nodes in the graph into vectors of the form of the first equation above, visualized as the blue hyperboloid sheet in Figure 5. These nodes obey a distance law, which is the second equation, that is used as a proxy to the number of edges between nodes $p$, $q$ in the embedding.

### Poincaré Disk

Domain: $\left\{(x_1, \ldots, x_n, 0) : x_1^2 + \cdots + x_n^2 < 1\right\}$
Distance: $d(p, q) = \operatorname{arcosh}\left(1 + \frac{2|pq|^2 |r|^2}{(|r|^2 - |op|^2)(|r|^2 - |oq|^2)}\right)$
Riemannian metric: $ds_P^2 = 4\frac{dx_1^2 + \cdots + dx_n^2}{\left(1 - x_1^2 - \cdots - x_n^2\right)^2}$
We can derive the projection onto the Poincaré Disk intuitively from the diagram in Figure 5. The projection lies at the intersection of the line connecting $(0, \ldots, 0, -1)$ to the hyperboloid projection and $x_{n+1} = 0$. The distance function for a Poincaré Disk is a bit more involved, with $r$ representing the radius of the disk, $|op|$ and $|oq|$ representing the Euclidean distance from $p$ and $q$ to the origin of the disk, and $|pq|$ representing the Euclidean distance from $p$ to $q$.

**Half-Plane Model**

Domain: $\{(1, x_2, \ldots, x_{n+1}) : x_{n+1} > 0\}$
Distance: $d(p,q) = \text{arcosh}\left(1 + \frac{(q_1-p_1)^2+(q_2-p_2)^2+\ldots+(q_{n+1}-p_{n+1})^2}{2q_{n+1}p_{n+1}}\right)$
Riemannian metric: $ds_H^2 = \frac{dx_2^2+\cdots+dx_{n+1}^2}{x_{n+1}^2}$

**Beltrami-Klein Model**

Domain: $\left\{(x_1, \ldots, x_n, 1) : x_1^2 + \cdots + x_n^2 < 1\right\}$
Distance: $d(p,q) = \frac{1}{2}\log\frac{|aq||pb|}{|ap||qb|}$
Riemannian metric: $ds_K^2 = \frac{dx_1^2+\cdots+dx_n^2}{\left(1-x_1^2-\cdots-x_n^2\right)} + \frac{(x_1dx_1+\cdots+x_ndx_n)^2}{\left(1-x_1^2-\cdots-x_n^2\right)^2}$

For the distance for Beltrami-Klein, we have a similar notational convention as Poincaré, with $p$, $q$ being the points of interest. Then $a$, $b$ are the intersections of the line joining $p$ and $q$ with the boundary of the ball.

**Calculations for $a$ and $b$:**
Parametric line through $p$ and $q$: $f(c) = c \cdot p + (1-c) \cdot q$.
Solve $||f(c)||_2^2 = 1$ to ensure $a$ and $b$ lie on the boundary of the ball.

$$\begin{aligned}
||f(c)||_2^2 &= ||c \cdot p + (1-c) \cdot q||_2^2 \\
&= c^2 p^T p + 2c(1-c)p^T q + (1-c)^2 q^T q \\
&= c^2 p^T p + (2cp^T q - 2c^2 p^T q) + (q^T q - 2cq^T q + c^2 q^T q) \\
&= c^2 \cdot (p^T p - 2p^T q + q^T q) + c \cdot (2p^T q - 2q^T q) + (q^T q) \\
&= c^2 \cdot ||p-q||_2^2 + c \cdot (2(p-q)^T q) + (q^T q) \\
&= 1
\end{aligned}$$

We can then solve $c^2 \cdot ||p-q||_2^2 + c \cdot (2(p-q)^T q) + (q^T q - 1) = 0$ for $c$ as a quadratic equation. Then, $a = f(c_-)$ and $b = f(c_+)$.

## References

[1] Cannon, J. W., Floyd, W. J., Kenyon, R., & Parry, W. R. (1997). *Hyperbolic Geometry (Vol. 31)*. MSRI Publications.

[2] Chamberlain, B. P., Clough, J., & Deisenroth, M. P. (2017). *Neural Embeddings of Graphs in Hyperbolic Space*. 13th international workshop on mining and learning from graphs held in conjunction with KDD.

[3] Dhingra, B., Shallue, C.J., Norouzi, M., & Dahl, G.E. (2018). *Embedding Text in Hyperbolic Spaces*. arXiv preprint arXiv:1806.04313.

[4] Ganea, O. E., Bécigneul, G., Hofmann, T. (2018). *Hyperbolic entailment cones for learning hierarchical embeddings*. arXiv preprint arXiv:1804.01882.

[5] Kallosh, R. & Linde A. (2015). *Escher in the Sky*. Comptes Rendus Physique 16. 914-927 arXiv:1503.06785.

[6] Nickel, M. & Kiela D. (2017). *Poincaré Embeddings for Learning Hierarchical Representations*. Advances in Neural Information Processing Systems 30.

[7] Sala, F., de Sa, C., Gu, A., & Re, C. (2018). *Representation Tradeoffs for Hyperbolic Embeddings*. International Conference on Machine Learning.

[8] Wilson, B., & Leimeister, M. (2018). *Gradient descent in hyperbolic space*. arXiv preprint arXiv:1805.08207.