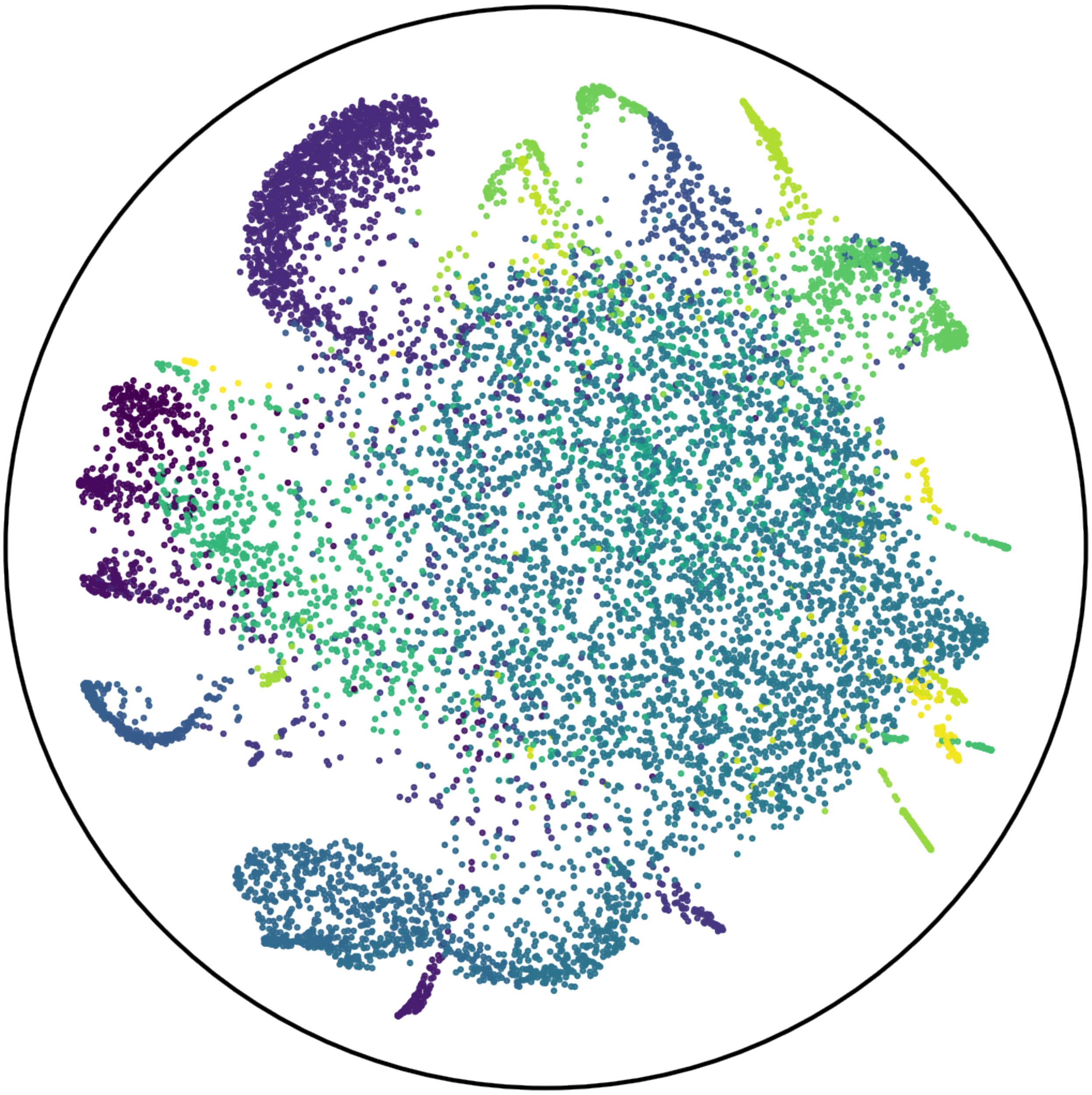
Delft University of Technology

Revisiting Hyperbolic t-SNE

gradient derivation and limitations of the t-distribution

by Haoran Xia



Revisiting Hyperbolic t-SNE

gradient derivation and limitations of the t-distribution

by

Haoran Xia

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday September 5th, 2025.

Student number:	5611024	
Thesis committee:	Prof. Klaus Hildebrandt	TU Delft, Main supervisor
	Prof. Martin Skrodzki	TU Delft, Daily supervisor
	Prof. Jasmijn Baaijens	TU Delft, committee member

TU Delft

Abstract

Dimensionality reduction and visualization methods have become an indispensable tool for the exploration of high-dimensional data. In this area t-SNE has established itself as a primary method; providing a means of visualizing high-dimensional data in a way that preserves local neighbourhood structures. However standard t-SNE embeds data into Euclidean space which struggles to capture properties of data that is network-like, tree-like, or contains hierarchies. As an alternative, the use of Hyperbolic space has been proposed to allow for effective embeddings of such data as connections exist between complex networks and Hyperbolic. In addition, Hyperbolic space has been effectively utilized for representing hierarchical relationships. Given these endeavors, recent works have adapted t-SNE to Hyperbolic space using the Poincaré Disk model resulting in a method that can visually reveal network-like and hierarchical relationships. However, several related works contain an error in their derived gradient. As t-SNE uses the method of gradient descent for optimizing embeddings, this error may lead to incorrect results. In our work we first correct for this error and explore its consequences. One such consequence is that embeddings are strongly pushed outwards in the disk leading to unintelligible results. Since t-SNE uses a t-distribution to model embeddings, we propose the use of a Gaussian distribution instead as it encourages more compact embeddings. Finally we perform experiments comparing each method qualitatively by assessing resulting visualizations, and quantitatively via the PR-metric.

Contents

1 Introduction

Advancements in computing technologies have dramatically increased the ability to collect and store data. Such advancements have resulted in large and complex datasets with many variables describing the data (Also known as high-dimensional data). As a result, dimensionality reduction methods, which reduce the complexity of such datasets by reducing the number of variables (also known as features) needed to describe the data, have been proposed as a way to prevent such analysis from becoming too complex.

Thus in the modern day of working with high-dimensional and complex data, dimensionality reduction has become a common step in the data analysis pipeline. Areas of application for dimensionality reduction have been found in the setting of sports [50], machine learning [39], recommender systems [32] and single-cell analysis [13] just to name a few.

In this thesis we revisit the application of the dimensionality reduction and visualization algorithm t-SNE applied to Hyperbolic Space.

1.0.1. Dimensionality reduction

Effective dimensionality reduction reduces the complexity of the data by lowering the number of variables/features required to represent the data. At first this may seem problematic as we are essentially throwing away information about the data. However, most often the ”essence” of a dataset does not require all this information (i.e. variables/features) as it can be well described by a smaller set of features. This notion is at the core of the manifold hypothesis data which states that data that is originally high-dimensional (i.e. described using a large number of variables) actually lies along a lower-dimensional manifold embedded in the original high-dimensional space. In other words, the data can in actuality be described with fewer variables (lower dimensions).

This implies that an amount of information less than what is present in the original data is relevant for analysis. Thus effective analysis of the original high-dimension data can still be carried out on a simpler description of the dataset obtained via dimensionality reduction methods.

Furthermore, in the case that dimensionality can be reduced to less than three dimensions, visualization of the data is possible. The ability to visually explore data allows one to quickly uncover patterns, relationships, outliers, and other structures from the data using visual intuition [35].

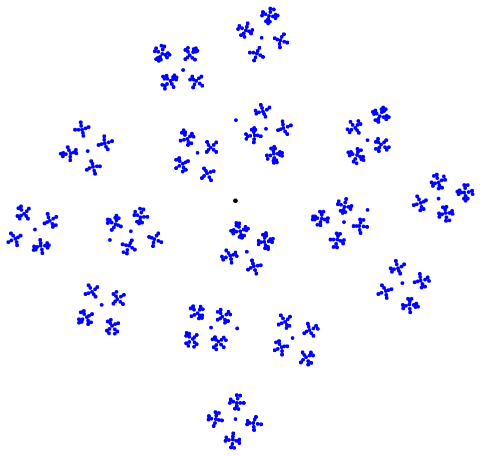
Finally, various kinds of methods exist for dimensionality reduction. These can be broadly classified into linear methods and non-linear ones. We will provide a brief overview of several such methods in section 2.1 however the main focus will be on the dimensionality reduction and visualization algorithm t-SNE [16]. t-SNE is a popular method for dimensionality reduction and visualization due to its ability to preserve local neighbourhoods especially well [51]. Since t-SNE is a core part of this thesis we will describe this method specifically in additional detail in section 3.

1.0.2. Dimensionality reduction in Hyperbolic Space

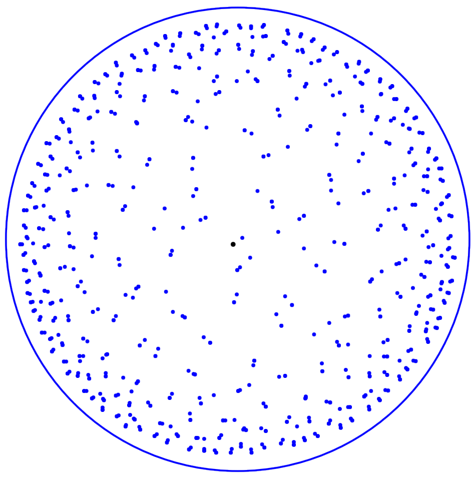
Traditional dimensionality reduction and visualization algorithms (with t-SNE being no exception) have always embedded data into flat/Euclidean space. However, the choice of embedding space (e.g. Euclidean or non-Euclidean) affects the kinds of embeddings that are obtained. In the context of graph embeddings, it has been found that Hyperbolic space is more effective for embedding data representing complex networks [14, 25].

Many real-world graphs such as the internet [21] and social networks [38] contain properties of complex networks (such as strong clustering and hierarchical structure) [28, 41]. The effectivity of Hyperbolic Space for embedding complex networks can be attributed to a connection between complex networks and Hyperbolic Geometry. In [14] a framework was developed to model complex networks using Hyperbolic Spaces. They show how common properties of complex networks arise by assuming an underlying Hyperbolic Geometry to these networks. In addition, data exhibiting tree-like structure (which also falls under the category of graph-like data), can not be embedded in Euclidean Space without distortion [18]. However it has been shown that Hyperbolic Space is able to embed trees well with (arbitrarily) low distortion [31, 30].

Combined with the findings that many real-world networks exhibit tree-like structures [1, 21, 38, 41], Hyperbolic Space thus finds applications in embedding tree and complex network-like data. In this



(a) Standard t-SNE in Euclidean Space embedding of a tree-like dataset. Each cluster represents a node in the tree. The center point is the root of the tree. Notice how in Euclidean Space this structure is not obvious. It is hard to assess for tree-like structures in Euclidean Space.



(b) Hyperbolic embedding (in the Poincaré Disk) of a tree-like dataset. The center black dot is the root of the tree. In the Poincaré Disk the points inhabiting some (approximated and imagined) circle of the same radius (from the origin) represents the points that are at a similar depth level in the tree. Increasing radii correspond to increasing depth. The Poincaré Disk thus allows for a clear way to assess for tree-like structures.

Figure 1.1

thesis we focus on methods for visualizing data in embedded into Hyperbolic Space. Since Hyperbolic Space is effective at capturing hierarchical relationships and for embedding trees (see figure 1.1 for an example).

Various methods have already been proposed [7, 34, 12, 33] that adapt t-SNE [16] to Hyperbolic Space. However in some of these works (specifically in [7, 33]) the proposed methods contain a mistake in the gradient (the Hyperbolic variant of the t-SNE gradient) formulation. Since t-SNE optimizes embeddings via the method of gradient descent, an incorrect gradient likely results in incorrect embeddings with respect to the cost function that we aim to minimize.

As part of our contributions, we will explain where the error occurs in the gradient derivation and how this error likely came to be by investigating the derivation of the original t-SNE gradient. Additionally we shall correct for this error and explore the behaviour of the standard version of Hyperbolic t-SNE (i.e. a direct adaptation of t-SNE to Hyperbolic Space as described in [33]) using the corrected gradient.

We then showcase how the standard version of Hyperbolic t-SNE (using both the corrected gradient and incorrect gradient) produces undesirable results. These results showcase that embeddings produced using Hyperbolic t-SNE are strongly pushed outward in the Poincaré Disk (the object which Hyperbolic t-SNE embeds in) resulting in unintelligible visualizations. We shall see that this boundary-pushing behaviour is a direct consequence of using the t-distribution. To tackle this issue we propose and justify replacing the t-distribution used in (Hyperbolic) t-SNE with a Gaussian distribution which we aptly name **Hyperbolic SNE** (in accordance with the progenitor of t-SNE [16], SNE [9] that uses a Gaussian distribution).

We will then find that Hyperbolic SNE produces better (more intelligible and interpretable) visualizations. Additionally, we also provide a qualitative assessment of the methods via the embedding of artificial tree-like datasets, and a quantitative assessment by comparing the PR-metrics [27] across Hyperbolic SNE and Hyperbolic t-SNE.

In summary the main contributions are:

- A correction to the Hyperbolic t-SNE gradient (as presented in [33, 7], and an analysis of the origin of the incorrect derivation.
- An analysis of the results produced using the correct and incorrect Hyperbolic t-SNE gradient showcasing undesirable results as a consequence of embedding data using Hyperbolic t-SNE.
- An adjustment to Hyperbolic t-SNE by replacing the t-distribution with a Gaussian to combat boundary-pushing behaviour in the resulting embeddings.
- An investigation into the usage of the Gaussian distribution, what it means for the resulting embeddings, and a comparison between results from using the Gaussian and the correct/incorrect Hyperbolic t-SNE gradient (via a qualitative and quantitative form of quality assessment).

2 Related Works

2.1. Dimensionality reduction

Dimensionality reduction techniques are ones that simplify a dataset by reducing the number of variables required to describe the dataset. As alluded to in the introduction, it is a common technique since high-dimensional datasets are hypothesized to lie on lower dimensional manifolds [48]. These methods can be broadly divided into two categories: Linear and non-linear methods.

2.1.1. Linear methods

Linear methods reduce the dimensionality of the dataset by constructing low-dimensional features that are linear combinations of the original high-dimensional features. Features in this context refers to the variables that are used to describe the data. For example, data about a person could be described using variables such as: age, height, and weight. These variables that describe the data are called the features of the data. From this point onward we shall use the term features to refer to these variables. Linear methods then projects the original dataset onto the new set of features. This is in fact where the term ”linear” in its name comes from; data is projected onto a new set of features that are linear combinations of the original features. A popular linear method is PCA [10] which constructs these features by finding a linear combination of the original features that maximizes the variance of the data after projection. However linear methods are limited in their ability to capture complicated relationships and structures.

For example, PCA struggles with datasets such as the ”Swiss Roll” dataset. Since the Swiss Roll dataset is a 2-dimensional manifold in 3-dimensional space (a rolled up 2-dimensional sheet), the variance in the data is thus not aligned with a flat plane. Rather, the variance curls along with the Swiss Roll manifold. Thus the two points that are most different are points at the respective ends of the roll. Effective embeddings of the Swiss Roll should therefore be able to unroll the dataset into a sheet. This however can not be captured by a linear function of the features in the data (i.e. the three axes in the image). Therefore PCA struggles with correctly embedding this dataset.

2.1.2. Non-linear methods

Non-linear methods do not construct new features linearly from the original features. Instead these methods embed data onto a new set of features whose construction is non-linear and depends on the method in question. This allows the method to capture more complicated relationships (as linear methods can only capture linear relationships), and be more flexible in their mapping. An alternative name for these methods is Manifold learning [49] as they aim to project high-dimensional data onto their lower-dimensional manifolds [47, 48]; effectively ”learning” the underlying manifold from the high-dimensional data

We will briefly explain two popular modern non-linear techniques: t-SNE [16] and UMAP [23] as well as an older technique called Multidimensional Scaling (MDS) [15] to give the reader some insight into

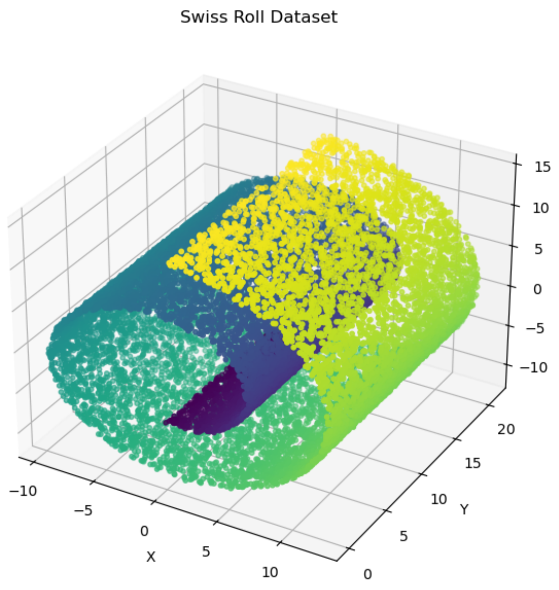


Figure 2.1: Swiss Roll Dataset

the topic of non-linear dimensionality reduction. This brief exposition aims to highlight the core ideas behind non-linear (or manifold learning) algorithms as they all share a core philosophy despite being different in their approach.

Firstly, **MDS** [15] works by first finding the distances between each and every point in the original (high-dimensional) data. This results in what is called a distance matrix. The distances that this matrix encodes denote how close or far away points are from each other. It therefore represents the structure of our data. Points are then embedded (placed) into an embedding (low-dimensional) space such that the embedded points' distance matrix match the high-dimensional one as well as possible. In other words, the goal is to find an low-dimensional embedding (placement of points) whose distances between each other match the high-dimensional distances.

t-SNE [16] starts similarly. It first computes a distance matrix of the high-dimensional data. However after this step the distances are converted into probabilistic values. A relatively high probability corresponds to points that are close to each other, and a low probability corresponds to points that are further away. t-SNE differs from MDS by using probabilities to denote similarities between points instead of distances. How the probabilistic values are computed can determine how much the global structure is considered. For example, if a distribution of the exponential family is used to model the probabilistic similarities, then large distances can be mapped to (near) 0 probabilities. This means that faraway datapoints do not effect each others embeddings much. This differs from MDS where distances are used directly so every points' embedding is affected by every other point. To construct the low-dimensional embedding t-SNE then embeds points such that the low-dimensional probabilities match the high-dimensional ones as best as possible. Further details will be provided in the background chapter: 3.

Finally, **UMAP** [23] works in a similar fashion. UMAP also builds an object analogous to a distance matrix in the high-dimensional space of the data. However the way UMAP measures closeness (or similarity) differs from t-SNE and MDS. UMAP also differs in how it decides where points in the low-dimensional space are to be embedded. The theoretical starting point for UMAP lies in a field called Topological data analysis. UMAP uses fuzzy simplicial sets to build a fuzzy topological structure of the high-dimensional data which is analogous to the distance or similarity matrices used in MDS and t-SNE. This structure is then used to determine how points are to be embedded in the low-dimensional space.

In general non-linear dimensionality reduction techniques first construct a representation of the high-dimensional data in the form of a distance matrix (or something analogous to it). These methods then attempt to embed data in the low-dimensional space that preserves this representation (for example the distance matrix) as best as possible. The above described methods only scratch the surface of

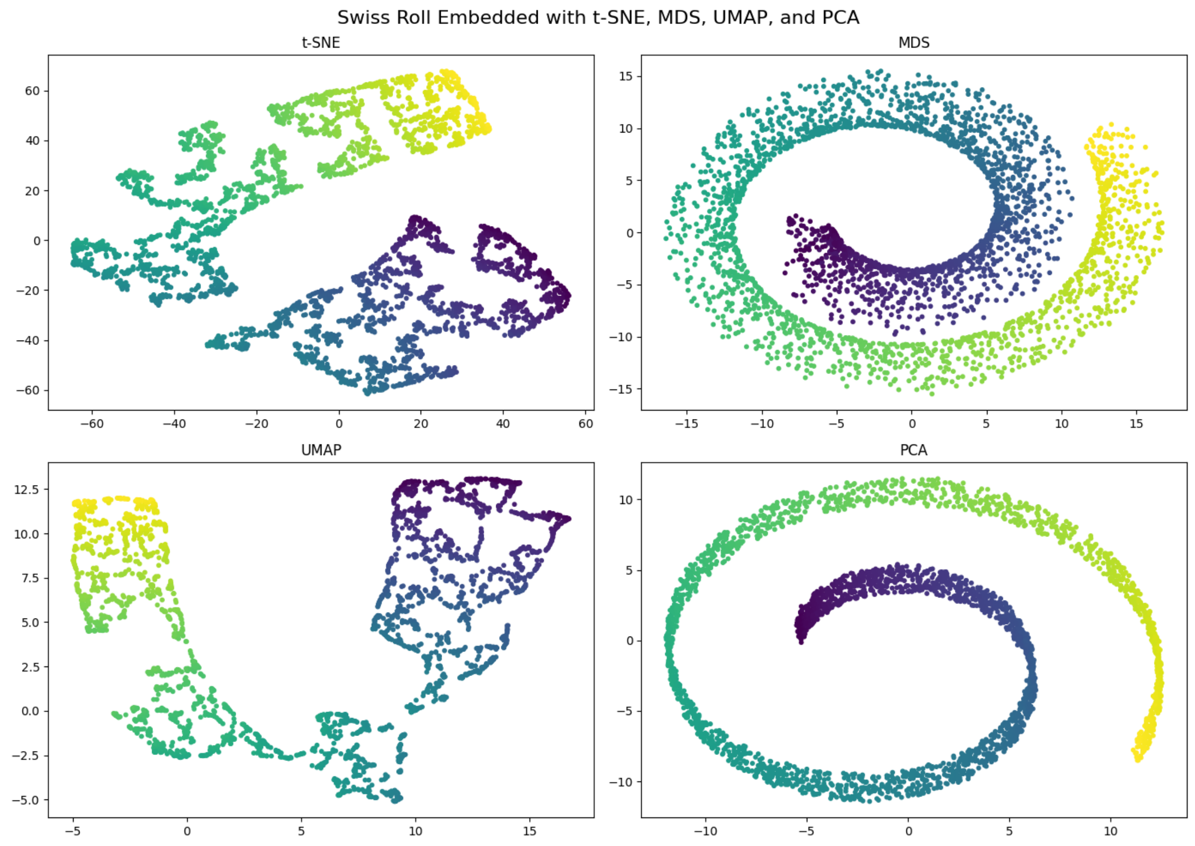


Figure 2.2: t-SNE, MDS, UMAP, PCA embedding of the Swiss Roll dataset. MDS and PCA struggle with ”unrolling” the swirl compared to t-SNE and UMAP. MDS struggles due its use of global distances (similarities). t-SNE and UMAP focus on local structure more, allowing them to unroll the Swiss Roll better.

available non-linear dimensionality reduction algorithms. We direct the reader to other resources [49] [19] for a more detailed look describing the advantages and disadvantages of these methods we refer the reader to a recent survey [51].

2.2. Hyperbolic embeddings

As alluded to in the introduction, data exhibiting network-like structure do not embed well into Euclidean Space. However, a connection between complex networks and Hyperbolic Space exists [14]. This has motivated research into embedding data in Hyperbolic Space.

For example, h-MDS (hyperbolic MDS) [4] is an adaptation of MDS [15] to Hyperbolic Space, and an adaptation of SOM (Self-Organizing Maps) to Hyperbolic Space [26] has also been developed. Finally several papers [7, 12, 34, 33] have proposed a version of t-SNE adjusted to Hyperbolic Space. All these methods seek to utilize the benefits of Hyperbolic Space for embedding data.

In this thesis the focus lies with t-SNE and its adaptation to Hyperbolic Space and not with other dimensionality reduction methods. This is due to the ubiquity in t-SNE’s usage for data analysis [29]. Furthermore, Hyperbolic t-SNE has acceleration structures available that allows for faster experimentation [33]. We therefore build on such previous works.

2.2.1. t-SNE in Hyperbolic Space

Existing methods that adapt t-SNE to the Hyperbolic setting differ from standard t-SNE by changing the probability distributions used, adjusting the cost function, and changing the optimization procedure to be compatible with Hyperbolic Space and the Poincaré Disk. We shall highlight four such prominent methods below:

Firstly, **CO-SNE** [7] adapts t-SNE to the Hyperbolic setting by first interpreting high-dimensional data as being drawn from a Hyperbolic Normal distribution. In the low-dimensional case the probabilities are computed via the Hyperbolic Cauchy distribution. In addition, an extra term is added to the cost function that minimizes the difference between the norms of the high-dimensional data and the low-dimensional embeddings. The motivation behind this is that it encourages the preservation of the global hierarchy present in the high-dimensional data.

Poincaré maps [12] is another adaptation of t-SNE to the Hyperbolic realm. Here the high dimensional probabilities are computed by first approximating the local connectivity of the assumed manifold on which the data lies on. This is done by constructing a KNN graph based on the high-dimensional data and then connecting any disconnect components that may have arisen from the KNN graph construction. Afterwards the global connectivity (high-dimensional probabilities) is found by employing the Relative Forest Accessibility index on the Laplacian resulting from the KNN graph construction described earlier. Low dimensional probabilities are then modeled by Gaussian kernels using Hyperbolic distances and a symmetric version of the KL-divergence is used as the cost function.

h-SNE [34] extends t-SNE by adding a global distance term to the standard (Euclidean) t-SNE cost function which follows a Hyperbolic Metric. Their cost function thus consists of two terms. One being the standard t-SNE cost function, and the second being a Hyperbolic version of the global loss function as described by [52]. However h-SNE differs in the other methods mentioned in this section in using an Euclidean term. It is unclear how h-SNE's embeddings relate to Hyperbolic embeddings as they embed data using traditional t-SNE with an added Hyperbolic term.

Finally, **Accelerating hyperbolic t-SNE** [33] describes a direct adaptation of t-SNE to the Hyperbolic setting by using Hyperbolic distances in the low-dimensional probabilities part of the t-SNE cost function. Additionally, this paper describes an acceleration scheme for Hyperbolic embeddings which will be briefly discussed in the section below.

In this thesis we focus on the method that is obtained when one directly adapts t-SNE to Hyperbolic Space. This is the version laid out in [33]. Our contribution of correcting for the gradient is relevant to [33] and [7] as both use an incorrectly derived gradient. Poincaré maps [12] does not provide an explicit derivation nor an explicit gradient formula so it is unclear how relevant the gradient correction is for them.

Our other contributions pertaining to the analysis of the usage of the t-distribution and the use of a Gaussian over the t-distribution is relevant for all the aforementioned related works. It must be noted however that Poincaré maps [12] also employs a Gaussian distribution for the low-dimensional similarities. However their choice comes unmotivated. In our work we motivate (both empirically and conceptually) why using a t-distribution is not a good choice for Hyperbolic space and how a Gaussian distribution follows as a natural choice considering the limitations of the t-distribution. Therefore part of our contributions is the justification for the use for a Gaussian.

Finally, h-SNE [34] describes a method that differs from the others and we will not consider them much moving forwards. However our contributions can still be relevant towards understanding their methodology more.

2.3. Acceleration techniques

Standard t-SNE has a computational complexity of $O(n^2)$. This makes t-SNE computationally unfeasible in circumstances where datasets (specifically the number of datapoints) are large. The original author of t-SNE proposed an acceleration scheme based on the Barnes-Hut algorithm [22]. The Barnes-Hut acceleration can be used to speed up computations by approximating the gradient instead of computing it exactly. This results in an $O(n \log n)$ implementation of t-SNE [22].

This scheme however does not translate well to the Hyperbolic setting (specifically the Poincaré Disk model of Hyperbolic Space) and therefore an alternative method or adaptation must be sought for. Recent work [33] has been done that improves the computational cost of Hyperbolic t-SNE based on a Polar Quadtree [36] data structure. The usage of the Polar Quadtree allows the Barnes-Hut method to be adapted to Hyperbolic embeddings in the Poincaré Disk. This means that the Hyperbolic version of the t-SNE gradient can also be approximated analogously to the approximation done by Barnes-Hut t-SNE [22]. For the details we refer the reader to [33].

In this thesis we shall make use of the Polar Quadtree-based acceleration scheme [33] to speed up our experiments.

3
Background

In this section we will give a detailed look into t-SNE and provide a conceptual overview of Hyperbolic Space. Finally we discuss how t-SNE can be adapted to produce embeddings in Hyperbolic Space.

3.1. t-SNE

t-SNE [16] is a popular [29] non-linear dimensionality reduction method that is capable of visualizing high-dimensional data by mapping data into at most 3-dimensions while aiming to preserve local neighbourhood structure. Local neighbourhood structure is preserved by mapping points in such a way such that nearby points in high-dimensional space remain nearby in low-dimensional space. Formally t-SNE takes as input high-dimensional data $\mathbf{x}_i \in \mathbb{R}^n$ and produces a low-dimensional embedding $\mathbf{y}_i \in \mathbb{R}^m$ with $m < n$ (usually $m = 2$) of the data.

In brief, t-SNE starts by computing a similarity matrix P (with entries p_{ij}) for the high-dimensional data $\mathbf{x}_i$. It then attempts to embed data into a low-dimensional space such that the low-dimensional embeddings $\mathbf{y}_i$ produce a (low-dimensional) similarity matrix Q (with entries q_{ij}) that is similar to P. Recall from section 2.1.2 that the similarity matrices encode the structure of our data. Therefore producing embeddings with a similarity matrix similar to the high-dimensional case can be intuitively thought of as obtaining an embedding with similar structure as the original data. This is desirable as it allows us to visually investigate the structures. Finally, How good an embedding is is measured via the KL-divergence between the two distributions (the two similarity matrices P and Q). The KL-divergence allows us to quantify how similar two distributions are.

In the following sections we will go into each of these steps in detail. Additionally, some time will also be spent to understand t-SNE on a conceptual level. The motivation behind the t-distribution will be explored as well as how local neighbourhood structure are preserved in t-SNE. Last but not least, a physical interpretation of the t-SNE gradient will be made which will shed some light on the embedding procedure itself (i.e. the process of iteratively computing the embedding of the data until convergence).

3.1.1. The inner workings of t-SNE

We first go through how the similarities P and Q in each respective dimension (the high-dimensional data $\mathbf{x}_i$ and the low-dimensional embeddings $\mathbf{y}_i$) are computed. We then look at the cost function C and finally the optimization procedure.

High-dimensional similarities

t-SNE starts by computing the high-dimensional probabilistic similarities P. Each entry p_{ij} in P is defined as follows:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n} \tag{3.1}$$

Where the conditional probabilities are computed as:

$$p_{j|i} = \frac{\exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-||\mathbf{x}_i - \mathbf{x}_k||^2/2\sigma_i^2)} \tag{3.2}$$

Here $p_{i|i}$ is defined to be 0. Furthermore, a Gaussian distribution is used as the distribution of choice to model similarities between points as can be seen from the $\exp(-||\mathbf{x}_i - \mathbf{x}_j||/2\sigma_i^2))$ term.
Conceptually speaking, a Gaussian distribution P_i with variance σ_i^2 is placed on top of each high-dimensional data-point $\mathbf{x}_i$. Similarities between $\mathbf{x}_i$ and its neighbours are then computed via this Gaussian, normalized via equation 3.2, and symmetrized via equation 3.1. Symmetrization provides some benefits for how outliers in the dataset are represented in P. For more details please see [16].

The benefit of using a probabilistic similarity-based approach over a purely distance-based (e.g. distance matrix such as in MDS [15]) approach is that we can focus less on global structures allowing for more flexibility in our representation of the data. This aligns with what t-SNE excels at and is designed for (i.e. maintaining local structure) as datapoints that are nearby will have an appreciable similarity (probability of being neighbours), while distant datapoints will have (close to) 0 probability. This means that distant points do not affect the embeddings of each other much. In a purely distance-based approach, every point affects the positioning of every other points when we attempt to position points such that the similarity matrices match. This leads to less flexibility in the embeddings as there are more constraints on what an optimal embedding is (what embeddings make the similarity matrices match).

Choosing the variance σ_i^2

Because a Gaussian distribution is used for the high-dimensional similarities, a variance σ_i^2 hyperparameter must be manually decided on.

In t-SNE, the value of σ_i^2 is set to match a user-defined perplexity value $Perp(P_i)$. One singular variance σ^2 is chosen for every Gaussian that sits on top of the high-dimensional data $\mathbf{x}$. (i.e. for every P_i that requires a σ_i^2 to be set, we choose one σ^2 based on the perplexity and set every $\sigma_i^2 = \sigma^2$. Note that the subscript i refers to the induced distribution P_i and not singular data-points $\mathbf{x}_i$). This induces a probability distribution P_i whose entries are computed according to equations 3.2 and 3.1. We then compute $2^{H(P_i)}$ which is equal to a user-defined perplexity value $Perp(P_i) = 2^{H(P_i)}$. Where $H(P_i)$ is the Shannon Entropy measured in bits:

$$H(P_i) = -\sum_j p_{j|i}\log_2 p_{j|i} \tag{3.3}$$

The value of σ_i^2 can be searched for by employing a binary search method (see [16]) that matches the Perplexity $Perp(P_i)$.

Since the high-dimensional probabilities P do not change during the embedding process, the above computations only need to happen once.

Low-dimensional similarities

In the (low-dimensional) embedding, corresponding probabilistic similarities are computed as follows:

$$q_{ij} = \frac{(1+||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}}{\sum_{k\neq i}(1+||\mathbf{y}_i - \mathbf{y}_k||^2)^{-1}} \tag{3.4}$$

The similarities over the embeddings use a t-distribution (note the $(1+||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}$ term) instead of a Gaussian distribution as employed in the high-dimensional similarities. The reasoning behind this design choice, as well as its consequences will be discussed in section 3.1.3.

Cost Function

Finally, the KL-divergence is used to measure how good the embeddings are:

$$C = KL(P||Q) = \sum_i\sum_j p_{ij}\log\frac{p_{ij}}{q_{ij}} \tag{3.5}$$

The cost function is minimal when $p_{ij} \approx q_{ij}$. Therefore the KL-divergence measures how different two distributions are. If the two distributions are the same we obtain a cost of 0.

For t-SNE this means that the optimal embedding requires points in low-dimensional space $\mathbf{y}_i$ to be positioned in a way such that the low-dimensional similarities Q (which are computed over the low-dimensional embeddings $\mathbf{y}_i$) match the high-dimensional similarities P as best as possible. This is the measure of goodness that t-SNE employs.

Optimization

To get to a good solution (i.e. a good embedding of the dataset) the positions of the embedding points $\mathbf{y}_i$ are adjusted in a way that minimizes the cost function. This procedure responsible for these adjustments is the method of gradient descent [45]. Gradient descent is an optimization method that can find local optima (minima) of functions by starting with a random solution (in this case $\mathbf{y}_i^t$ for $t = 0$) and iteratively updating that solution using the gradient of the function with respect to that solution: $\frac{\delta C}{\delta \mathbf{y}_i^t}$. This procedure minimizes functions because the gradient of a function is the direction in which the function increases the most. Moving in the opposite direction (descending along the gradient), thus allows us to step into the direction of most decrease allowing us to eventually converge to a (local) minima.

The computation of the gradient descent steps uses the expression below (where η is the learning rate):

$$\mathbf{y}_i^{t+1} \leftarrow \mathbf{y}_i^t \ - \eta \frac{\delta C}{\delta \mathbf{y}_i^t} \tag{3.6}$$

The new embeddings $\mathbf{y}_i^{t+1}$ are the embeddings of the previous iteration $\mathbf{y}_i^t$ moved slightly in some direction determined by the gradient of the cost function $\frac{\delta C}{\delta \mathbf{y}_i^t}$. This update step is then executed iteratively until some convergence criteria or a maximum number of iterations steps is reached.

Finally, in standard t-SNE practice, the optimization procedure scales a part of the gradient (the positive forces in the gradient, see the next section for what this means) by some constant for some initial predetermined amount of iterations. This is to encourage early cluster forming as it encourages similar points to attract during the initial optimization stage. This process is called early exaggeration (see the original t-SNE paper [16]).

Interpretation

t-SNE is optimized via gradient descent using the gradient of the cost function described above. This gradient takes the form (for a derivation please see the t-SNE appendix [16]:

$$\frac{\delta C}{\delta \mathbf{y}_i} = 4 \sum_j (p_{ij} - q_{ij})(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}(\mathbf{y}_i - \mathbf{y}_j) \tag{3.7}$$

To get a better sense of the behaviour of the gradient, i.e. how the embedding procedure proceeds since the embedding happens via gradient descent; a physical analogy can be made.

Physically, the gradient (equation 3.7) can be interpreted as the forces resulting from a set of springs acting between each pair of points $\mathbf{y}_i$ and $\mathbf{y}_j$. This is because this equation takes on a similar form as an equation describing a mechanical system consisting of a spring. In our gradient, we can interpret it as there existing a spring between every two points $\mathbf{x}_i, \mathbf{x}_j$ where $i \neq j$). These springs exert a force along the direction ($\mathbf{y}_i - \mathbf{y}_j$), i.e. between each other. If $(p_{ij} - q_{ij}) > 0$ meaning $p_{ij} > q_{ij}$, then this means that the low-dimensional embeddings $\mathbf{y}_i$ and $\mathbf{y}_j$ are to close the distance to each other (since $p_{ij} \approx q_{ij}$ constitutes an optimal embedding). In other words, $\mathbf{y}_i, \mathbf{y}_j$ are pulled towards each other. This denotes a an attractive force or a negative spring force. Conversely, if $p_{ij} < q_{ij}$, the embeddings are to take a larger distance from each other.
Therefore $(p_{ij} - q_{ij})$ determines whether the points $\mathbf{y}_i$ and $\mathbf{y}_j$ are to experience a repulsive (negative) or a attractive (positive) force due to the spring that can be imagined to be between them. Finally $(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}$ modulates the strength of the force.
We can make the above analogy even more explicit by rewriting the gradient into an expression where

the repulsive and attractive forces are separated into their own distinct terms:

$$\frac{\delta C}{\delta \mathbf{y}_i} = 4\left(\sum_j p_{ij}q_{ij}Z(\mathbf{y}_i - \mathbf{y}_j) - \sum_j q_{ij}^2 Z(\mathbf{y}_i - \mathbf{y}_j)\right) \tag{3.8}$$

Where $Z = \sum_j (1+||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}$, $\sum_j p_{ij}q_{ij}Z(\mathbf{y}_i - \mathbf{y}_j)$ represent the attractive forces, and $\sum_j q_{ij}^2 Z(\mathbf{y}_i - \mathbf{y}_j)$ the repulsive forces.

At each gradient descent iteration, these spring forces are computed (i.e. the gradient is computed), and the embedded points are updated according to this gradient (the spring forces). Thus the gradient descent procedure iteratively displaces the embeddings until an embedding with a minimum energy (measured by the KL divergence) is reached. This is directly analogous to a physical system of springs moving towards a stable configuration from some starting/initial configuration.

3.1.2. Capturing local neighbourhood structure

t-SNE by design focuses on capturing the local neighbourhoods of points. How t-SNE achieves this is by allowing neighbouring points to influence each other much more strongly than distant points.

In t-SNE points that are distant from each other have a probabilistic similarity that is negligible (i.e. a very small p_{ij}) since the p_{ij} depends on the distance between $\mathbf{x}_i, \mathbf{x}_j$. Employing a Gaussian distribution (which has exponentially decaying tails) means that relatively far away points have a probabilistic similarity that is very small or effectively 0. These points do not affect the cost function 3.5 since terms with very small p_{ij} are effectively 0 thus contributing little to the sum. Local relationships are therefore emphasized as they contribute a larger amount to the cost function, while global relationships (i.e. relationships between distant points) matter less.

We can deduce from the gradient formula in equation 3.7 what this means for the embeddings themselves. Points that are distant from each other have a high-dimensional similarity that is small and thus negligible (i.e. $p_{ij} \approx 0$). The low-dimensional position of the embeddings therefore do not matter much as long as $p_{ij} \approx q_{ij} \approx 0$. This means that their corresponding embeddings $\mathbf{y}_i$ and $\mathbf{y}_j$ can be positioned anywhere where distances are large as this results in a similarity of $q_{ij} \approx 0$.
On the other hand, for any relatively large p such that $p_{ij} \approx q_{ij} \approx p$ (which happens for neighbouring points) the positions of the low-dimensional embeddings are optimally placed in a certain region that results in these probabilities. So the relationship (distance) in the embedding between $\mathbf{y}_i$ and $\mathbf{y}_j$ matter much more in this scenario. $p_{ij} \approx q_{ij} \approx p$ therefore ensuring that local neighbourhood relationships are preserved.

In summary, t-SNE puts most optimization pressure on keeping (high-dimensional) neighbourhoods together in the low-dimensional embedding. Distant data-points can be positioned across a much larger range of positions as compared to similar data-points (neighbours).

3.1.3. Crowding problem

A defining characteristic of t-SNE is the usage of a t-distribution to model the low-dimensional similarities. This specific feature aims to tackle something called the ”Crowding problem” present in the progenitor of t-SNE, SNE [9]. SNE, which used a Gaussian to model low-dimensional embedding points, struggled with the problem where embedded points tended to crowd together strongly which lead to cluttered visualizations. t-SNE addresses this issue by using a distribution with larger tails for the low-dimensional similarities resulting in the use of a t-distribution. The Gaussian distribution has exponentially decaying tails whereas the t-distribution decays at a slower rate. Since distances are converted to probabilities, the use of a distribution with larger tails means that larger distances in the embedding still correspond to appreciable probabilities. Using a Gaussian may result in probabilities of close to 0 (if not 0) for large distances. So by using the t-distribution larger distances in the low-dimensional embedding are required before $p_{ij} \approx q_{ij}$ (Before the two distributions match up which corresponds to a lower cost function value). As a result, embeddings become more spread out thus alleviating the crowding problem. For more detail please see the t-SNE paper [16].

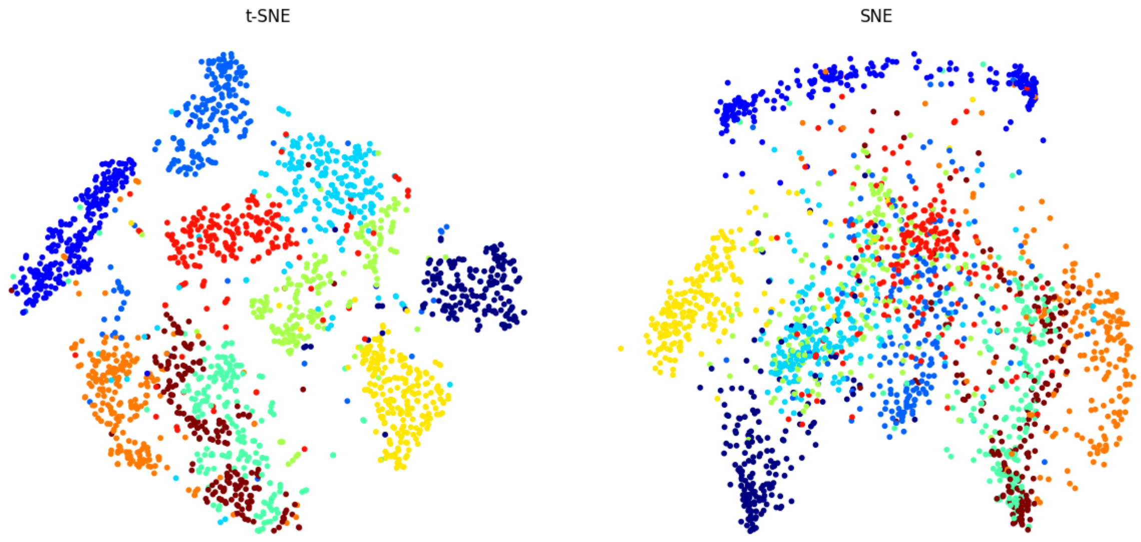


Figure 3.1: A comparison between t-SNE (left) and SNE (right) embeddings of a subsample of the MNIST dataset. Note how the SNE embeddings appear more crowded together resulting in a messier visualization.

3.2. Hyperbolic Space

The goal of this thesis is to use Hyperbolic Space for embedding data. It is therefore of importance to have an understanding of what Hyperbolic Space is, how it can be modeled, and how it can be used. In this section we shall provide a brief and mostly conceptual overview.

3.2.1. Why Hyperbolic Space?

In the introduction we have hinted at the benefits of embedding data in Hyperbolic Space. In this section we will explain these further.

Firstly, recall from the introduction that many kinds of data which we may wish to visually explore contain complex network-like or tree-like structures. In [14] the authors established a connection between complex networks and Hyperbolic geometry. This connection hinges on the fact that random graphs generated in Hyperbolic space (see the [14] for more details) exhibit complex network-like properties such as power-law degree distributions and strong clustering. Furthermore, the authors also showed that the converse holds; i.e. that such networks have an underlying Hyperbolic geometry. Therefore it is natural to model such networks as graphs in Hyperbolic Space. And thus Hyperbolic Space provides a natural setting for the embedding of such data.
The exact details of their argument are beyond this thesis, however we can gain some intuition behind this by examining two ideas in Hyperbolic space. Namely, how space expands exponentially and how distances between points in Hyperbolic space (and thus also the Poincaré Disk) behaves. Firstly, Hyperbolic space has the property that the amount of space grows exponentially as we move away from the origin. One way to show this is by computing the area (or volume) of objects such as a circle in both Hyperbolic and Euclidean space. One can show that the area of a circle in Hyperbolic space scales exponentially with its radius while the area of an (2-dimensional) Euclidean circle only grows quadratically (in general for n-dimensions, Euclidean volume grows polynomially and not exponentially). This is related to the distortion (in more technical terms the metric tensor) involved in Hyperbolic Space (explained further in section 3.2.4). This property is useful when we wish to embed objects that grow exponentially in size such as trees and networks. Euclidean Space would not be able to accommodate for the exponential growth of such objects [11, 25].

A related phenomena is that the pairwise distance $d^{\mathcal{H}}(\mathbf{x},\mathbf{y})$ between two points $\mathbf{x},\mathbf{y} \in \mathcal{D}$ on the Poincaré Disk $\mathcal{D}$ approaches the sum of the distances $d^{\mathcal{H}}(\mathbf{x},\mathbf{y}) \to d^{\mathcal{H}}(\mathbf{x},O) + d^{\mathcal{H}}(\mathbf{y},O)$ of each point and the origin O (see figure 3.2, for more details see [30]). This mimics how distances between nodes in trees are determined as the pairwise distance between any two nodes in a tree must go through the closest parent shared by each of the nodes involved. A consequence of this is that we can interpret points

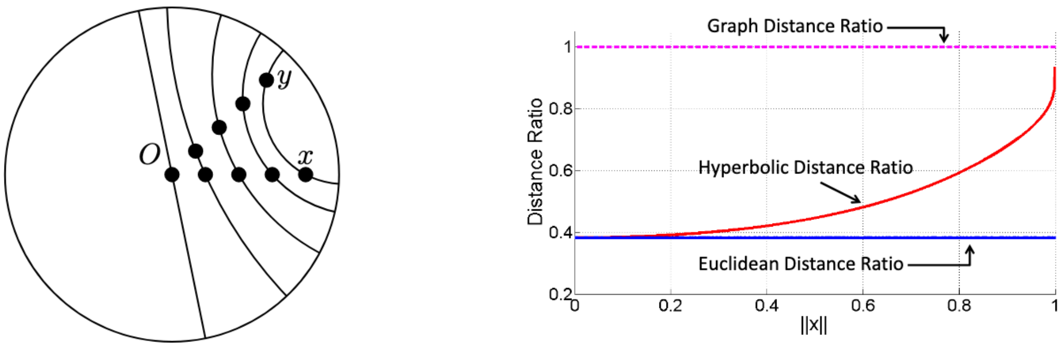


Figure 3.2: On the left we see points on the Poincaré Disk lying on geodesics (straight lines) in the disk. These points are positioned to reflect a tree-like structure. On the right the ratios of distances between two points x and y on the Poincaré Disk are plotted. The Euclidean distance ratio is $\frac{d_E(x,y)}{d_E(x,0)+d_E(y,0)}$ (where $d_E(x,y)$ is the Euclidean distance between two points) and the Hyperbolic distance ratio is $\frac{d_H(x,y)}{d_H(x,0)+d_H(y,0)}$ (where $d_H(x,y)$ is the Hyperbolic distance between two points). The Hyperbolic distance ratio approaches 1 as the points on the disk (x, y in the above example) approach the border of the disk. The fact that the ratio approaches 1 means $d_H(x,y) \to d_H(x,0) + d_H(y,0)$ which reflects how distances behave for trees. (For more details please see the source: [30])

lying on an (imaginary) circle of the same radius r to represent points existing on a similar depth in a tree-like structure.

We thus see how Hyperbolic Space (and the Poincaré Disk) is useful in embedding network and tree-like data.

3.2.2. What is Hyperbolic Space?

Hyperbolic Space refers to a geometrical space where every point in space has constant negative sectional curvature. For our purposes we will be dealing with the Hyperbolic Plane (2-dimensions). In 2-dimensions the concept of sectional curvature is called Gaussian curvature. Any further references to Hyperbolic Space will thus implicitly refer to the Hyperbolic Plane. The Hyperbolic Plane is thus a 2-dimensional surface where each point has constant negative Gaussian curvature.

Gaussian Curvature

Gaussian curvature [43] is a concept that describes how a surface curves at a point. Specifically, how a surface curves in two perpendicular directions. If we imagine ourselves standing on a point on some surface then we can decide to walk into two perpendicular directions (e.g. horizontal/vertical or north/east). We can then measure the curvature of each of these directions. For every possible pair of perpendicular directions, there is a pair where one direction has maximal curvature (i.e. curves the most ”up” out of any direction, often denoted by a positive sign), and one has minimal curvature (i.e. curves the most ”down” out of any direction, often denoted by a negative sign). We shall call these curvatures κ_1, κ_2. The Gaussian curvature is then the product of these two: $K = \kappa_1 \kappa_2$.

If both directions curve in the same direction (e.g. both curve up or both curve down), then K is always positive and we have a positively curved point. If $K = 0$ then there is no curvature and we have a ”flat” point (like how flat space has no curvature). Finally, if $K < 0$ then one of the curvatures κ_1, κ_2 must differ from the other (since they must have different signs otherwise the signs cancel out and we get $K > 0$). Such points are called saddle points (see figure 3.3) for their saddle-like appearance.

We are now ready to unpack the definition of Hyperbolic Space above. Hyperbolic Space is a geometrical space where each point is a saddle point as the principal curvatures κ_1, κ_2 must differ in sign for K to be negative.

Furthermore, using the above definitions for curvature, we can obtain three kinds of geometric spaces (or surfaces): Euclidean (constant zero curvature, $K = 0$), Spherical (constant positive curvature, $K > 0$), and Hyperbolic (constant negative curvature $K < 0$) space.

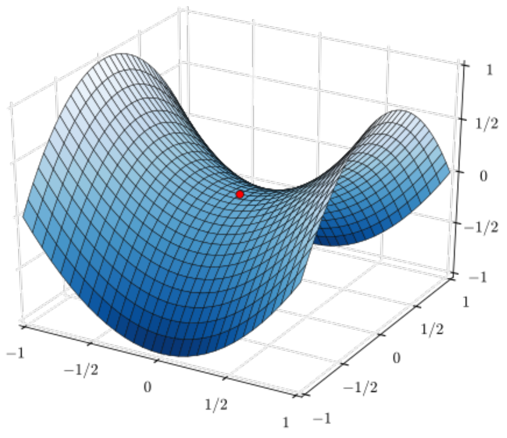


Figure 3.3: A saddle point. Note how we have two opposing types of curvature which is the defining property of a saddle point. (source: Nicoguaro, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons)

As a consequence of the curvature, geometry in each of these spaces appears different. For example, in Euclidean space the sum of angles of a triangle is exactly $180°$ while in Spherical space it is greater than $180°$ and in Hyperbolic space it is less than $180°$.

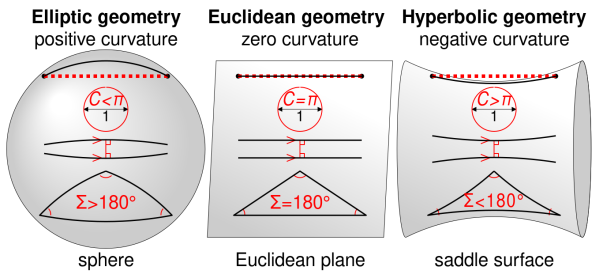


Figure 3.4: Comparison of the three geometries. A curvature of $K > 0$ refers to spherical geometry. $K = 0$ refers to standard (flat) Euclidean geometry. Finally $K < 0$ refers to Hyperbolic geometry (source: Cmglee - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=94781281)

3.2.3. Modeling Hyperbolic Geometry

To be able to make use of Hyperbolic Geometry for practical purposes we must establish a workable model of Hyperbolic Geometry first. In the case of Euclidean geometry this model is the Euclidean plane (or Euclidean space for dimensions greater than 2). We must analogously construct such a model of Hyperbolic Geometry.

By definition every point in Hyperbolic Space is a saddle point. Unfortunately this means that we cannot fully embed a Hyperbolic Plane (2D) into (3D) Euclidean space [8] like we can with a Positively curved surface (such as a Sphere). Since we can only visualize things up to 3-dimensional space any (visualizable) model of Hyperbolic Space will contain some form of distortion. Therefore Hyperbolic Geometry is difficult to conceptualize fully since we can not go beyond 3-dimensions in our visualizations.

Nonetheless, several models of Hyperbolic Geometry have been introduced. In this thesis we will focus on one such model namely the Poincaré Disk model (discussed in the following section) Other models of Hyperbolic Geometry also exist and may also be used in the context of embedding data. However much prior work has been in the Poincaré Disk model of Hyperbolic Space [25, 12, 34, 33, 7] which motivates our sole focus on the Poincaré Disk.

3.2.4. Poincaré Disk Model

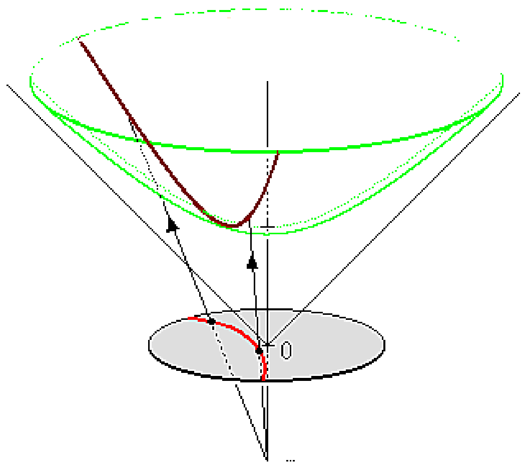
The Poincaré Disk is a model of 2-dimesional Hyperbolic Space embedded inside Euclidean Space.

It can be constructed via the projection of another model of Hyperbolic Geometry called the Hyperboloid model onto a disk of radius 1 (the unit disk) [46] (see fig 3.5a). We will not go into why the Hyperboloid model is a correct model for Hyperbolic Space. For more details please see [46]

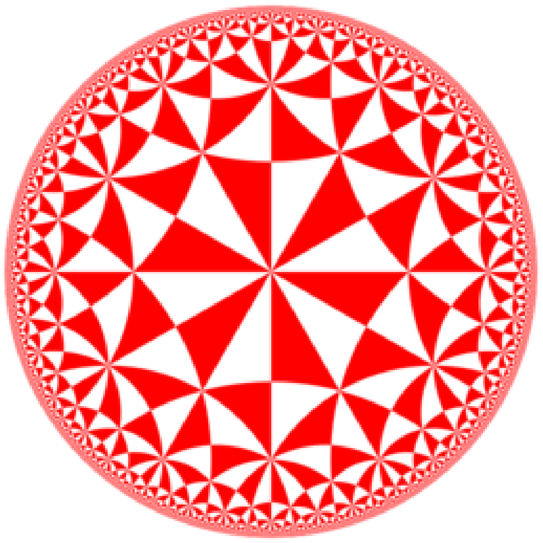
This construction allows us to fully describe Hyperbolic Space using all the points inside the unit disk (excluding the border). Mathematically speaking the coordinates are drawn from the set $\mathcal{D} = \{(y_1, y_2) \mid y_1^2 + y_2^2 < 1\}$. The Poincaré Disk $\mathcal{D}$ is thus a model of 2-dimensional Hyperbolic Geometry embedded in 2-dimensional Euclidean Space $\mathbb{R}^2$. Furthermore, it models all of (infinite) Hyperbolic Space in the unit disk (a finite model). As a result, some distortion must be happening. Specifically, as the border of the disk is approached, seemingly small regions of space represent large amounts of actual Hyperbolic Space (see fig:3.5b).

This means that there is a mismatch between the amount of space we see (i.e. we see Hyperbolic Space through our (Euclidean) model, the Poincaré Disk, which is a distorted view of true Hyperbolic Space), and the actual amount of space there is (the actual underlying amounts of Hyperbolic Space that the model describes). This is exactly the distortion introduced by the Poincaré Disk. We can describe this distortion mathematically using the concept of the Metric Tensor discussed later (section 3.2.4).

Another difference with respect to Euclidean Space is that due to constant negative curvature, straight lines (shortest paths in Euclidean space) do not appear straight in (this model of) Hyperbolic Space. The generalization of a straight line is called **a geodesic**. Figure 3.5a showcases an example of a geodesic.



(a) Construction of the Poincaré Disk as a projection of the Hyperboloid model (above, green) onto the Unit Disk (below, gray). The Hyperboloid Model extends infinitely far upwards. The red curve represents a geodesic on the Poincaré Disk. The brown curve is a geodesic on the Hyperboloid Model. (source: By Selfstudier - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=18419030)



(b) Tiling of Hyperbolic Space using the Poincaré Disk model. Every tile is the same size even though the tilings nearing the borders appear increasingly smaller. An example of the distortion at play. (source: Tom Ruen and Roice Nelson, Public domain, via Wikimedia Commons)

Figure 3.5

Overview of the model

The technical description of the Poincaré Disk is that it is a Riemannian Manifold. The concept of Riemannian Manifolds generalizes the idea of continuous and smooth space. It is a Smooth (Differentiable) Manifold equipped with a (Riemannian) Metric Tensor.

A smooth manifold is the formal mathematical definition of a smooth and connected space. It refers to the set of points that makes up the space/structure. In our case this is the Poincaré Disk. In these sections we will only introduce concepts necessary for understanding our work. This means that only the metric tensor will be explained in some detail. We refer the reader to other resources on differentiable geometry for a more in-depth look [6].

The metric tensor refers to an object that tells you how to measure lengths/distances (and as a consequence other geometrical notions) on this manifold. In other words, it is a measuring stick. We work with the Poincaré Disk model via $\mathbb{R}^2$ since we have an embedding of Hyperbolic Space into Euclidean Space. This means that any distances we measure through our model are also in Euclidean amounts and we must incorporate how much distortion is involved in these measurements to get the true amounts of distance corresponding as per Hyperbolic Space which is the actual type of space that underlies our model.

To make this distortion quantifiable we use the metric tensor. The metric tensor relates distances/units of space that we observe as an outsider looking at the disk (through our model which is embedded in $\mathbb{R}^2$ to the actual magnitudes of these distances/units in Hyperbolic Space (the geometrical space underlying the model which the model describes). For example, figure 3.5b showcases a tiling of the Poincaré Disk. Every tile is the same size. However our model distorts tiles that approach the border to appear smaller in size. If we were to calculate the areas of different tiles (which is based on our standard Euclidean notions of space as reference), these tiles would come out in differing sizes. However, we can use the metric tensor to correct for this distortion. It allows us to rescale distances/units of space during such calculations such that in our end result all the areas of the tiles come out the same size. We will see an example of this later on in section 3.2.4. To put it another way, distances on the disk (that we measure/observe through the model) can be rescaled using the metric tensor to their actual magnitudes in Hyperbolic Space.

To summarize, our model of Hyperbolic Geometry, the Poincaré Disk is a space consisting of a set of points (in technical terms a smooth manifold), equipped with a metric tensor (a measuring stick) for calculating the actual lengths of things. In other words, it is a Riemannian Manifold.

Metric Tensor

We have already established that the Metric Tensor allows us to formally describe (since it acts as a measuring stick) the distortion of space. In this section we will make this statement more precise.

To start we can gain some intuition about the metric tensor by considering the metric tensor on a well-understood Riemannian Manifold; namely the Euclidean plane. We shall see how the metric tensor is used to find distances in $\mathbb{R}^n$. We then generalize this result to the Poincaré Disk $\mathcal{D}$.

Metric Tensor in $\mathbb{R}^n$

In Euclidean Space $\mathbb{R}^n$(and by extension the Euclidean plane) the Metric Tensor is defined to be the identity matrix $g_E = I$ for every point in the space. One way to interpret this is that there is no distortion happening. To calculate geometrical concepts such as lengths and distances, the dot product (or more generally the inner product) is used.

A point in Euclidean space $\mathbb{R}^n$ can be represented as a vector $\mathbf{v} \in \mathbb{R}^n$. The dot product is then defined as:

$$\langle \mathbf{v}, \mathbf{v} \rangle \ = \ \mathbf{v} \cdot g_E \cdot \mathbf{v} = \mathbf{v} \cdot I \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{v} = ||\mathbf{v}||^2 \tag{3.9}$$

Which can be understood as a generalization of the Pythagorean Theorem.

The squared magnitude of vector $\mathbf{v}$, which is used in the computation of distances and other geometrical concepts, is obtained as a result of this operation. We can also observe that $g_E = I$ does not alter lengths in any way. Since the identity matrix is $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, when we compute the dot product as above, we simply get $\mathbf{v} \cdot \mathbf{v}$. In the Euclidean Plane we do not expect any lengths to be scaled. If, for example we would have some $c \neq 1$ along the diagonal instead, then the above dot product would instead be evaluated as $c \cdot \mathbf{v} \cdot \mathbf{v}$. This means that the result and therefore lengths are scaled.

Finally, distances between two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ can then be computed by calculating the magnitude of their difference. In other words, finding the magnitude of $\mathbf{u} - \mathbf{v}$ using the above definition.

Metric Tensor in $\mathcal{D}$

For the Poincaré Disk[1] $\mathcal{D}$ we have the following Metric Tensor (For a derivation please see [11]):

$$g_{\mathcal{D}} = \begin{bmatrix} \frac{4}{(1-||\mathbf{y}||^2)^2} & 0 \\ 0 & \frac{4}{(1-||\mathbf{y}||^2)^2} \end{bmatrix} = \frac{4}{(1-||\mathbf{y}||^2)^2} \cdot g_E \tag{3.10}$$

Where $\mathbf{y} \in \mathcal{D}$

Note that g_D depends on the point $\mathbf{y}$, meaning that for different points $\mathbf{y}$ on the disk, we have a different value for the metric. We can see that as we approach the border of the disk (i.e. as $||\mathbf{y}|| \to 1$), the diagonal entries become larger and larger, scaling things more and more. This is in line with our intuition. As we approach the border of the Poincaré Disk, space becomes increasingly distorted (squished). The distance between small differences in coordinate points near the boundary thus represents a large amount of Hyperbolic distance. The metric tensor $g_{\mathcal{D}}$ thus gives us the means to convert from distances on the disk to their actual Hyperbolic distances. We will illustrate how this metric tensor is used to obtain the Hyperbolic distance between two points on the Poincaré Disk.

Hyperbolic distance

To find a formula for the distance between two points on the Poincaré Disk, we must first find the geodesic between these two points. The geodesic can conceptually be thought of as the straightest path between two points. We then compute its arclength to obtain the distance. The arclength is computed by summing up (integrating over) the magnitudes of the tangent vectors of the curve, for every (infinitessimal) segment of the curve.

Tangent vector

Any continuous curve on some space has a tangent vector at every point along the curve. This tangent vector (of a curve) at some point (on the Poincaré Disk) can be thought of as how much distance one must travel if we were to move an infinitesimal step in the direction specified by the tangent vector. For example, if we were to follow some curve defined in the Poincaré Disk then we can step along the curve in very small (infinitesimal) steps. Each step must be done with some direction and magnitude and this direction and magnitude is defined by the tangent vectors at every step. The sum of all these steps then traces out the curve. For a more formal treatment of this please see [6, 11]. The main idea is that we can break down any curve into a sum of steps where each step has a direction and magnitude following some tangent vector[2]. Furthermore, we note that the tangent vectors are intrinsic to the model itself (in our case the Poincaré Disk). For example, the amount of distance covered by stepping into some direction of the tangent vector at some point on the disk is in terms of Hyperbolic spatial amounts of distance. So near the border of the disk, a large tangent vector (in Hyperbolic Space) corresponds to a small amount of distance on the Poincaré Disk (which is our model of Hyperbolic Space embedded in $\mathbb{R}^2$).

Euclidean distance computation using geodesics

In Euclidean Space the geodesic is simply a straight line. Thus the tangent vectors simply point along this line (they all point in the same direction with the same magnitude). Furthermore, the Euclidean metric is the identity matrix $g_E = I$ for every point thus no adjustments are required. This simplifies things and allows us to obtain the expression in equation 3.9. The arclength of an Euclidean geodesic is therefore exactly the magnitude of the difference of two position vectors $\mathbf{a}, \mathbf{b}$ which can be computed via the inner product (or the Pythagorean theorem).

One could arrive at the same expression in equation 3.9 by summing over the magnitudes of the tangent vectors of the parameterized (straight) path between two points $\mathbf{u}, \mathbf{v}$ (for infinitessimal stepsizes

[1]The Poincaré Disk is specified using 2 coordinates $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ and thus $||\mathbf{y}||^2 = y_1^2 + y_2^2$. Also recall that the coordinates of the Poincaré Disk is the set: $\{(y_1, y_2) \mid y_1^2 + y_2^2 < 1\}$

[2]To be more precise, our tangent vectors live in something called the tangent space (at some point $\mathbf{y}$). Every point on a Riemannian Manifold has a tangent space. Vectors in the tangent space behave just like vectors in Euclidean space. Furthermore, just like in Euclidean space, these tangent vectors can be written in terms of their (tangent space) basis vectors. Since this tangent space resembles Euclidean space, we can use the same inner product as described in equation 3.9. For more details please see more formal treatments of these subject such as [6].

this sum becomes an integral). The result would be the same. We can conceptualize this process as stepping in fixed increments (infinitessimal steps) along the geodesic (i.e. steps of $d\lambda$), each infinitessimal step corresponds to an amount of distance covered equal to the magnitude of the tangent vector at that point. Summing all these up between two points results in the total amount of distance covered (i.e. the arclength). We will illustrate this method for geodesics in the Poincaré Disk.

To remain within the scope of this thesis, we will not provide a derivation for geodesics on the Poincaré Disk. We refer the reader to several sources [44, 6, 11, 5] for more detail. We will only outline the process of computing the Hyperbolic distance between two points assuming the geodesic between these two points is known.

Hyperbolic distance computation

We start with the assumption that we have a (parametric) expression describing a geodesic on the Poincaré Disk between two points $\gamma_1, \gamma_2 \in \mathcal{D}$. We will call this expression $\gamma(\lambda)$ where $\lambda \in [a, b]$:

$$\gamma(\lambda) = (y_1(\lambda), y_2(\lambda)) \tag{3.11}$$

The arc length can then be computed using the arc length formula for curves on curved spaces [40, 42]:

$$arclength = \int_{\gamma_1}^{\gamma_2} ||\dot{\gamma}(\lambda)||\ d\lambda = \int_{\gamma_1}^{\gamma_2} \sqrt{g(\frac{d\gamma}{d\lambda}, \frac{d\gamma}{d\lambda})}\ d\lambda \tag{3.12}$$

Where g_D is the metric tensor on the Poincaré Disk and $\gamma_1 = \gamma(a)$, $\gamma_2 = \gamma(b)$ correspond to the start and the end points of the geodesic. In other words, they correspond to the two points $\mathbf{a}, \mathbf{b} \in \mathcal{D}$ whom we wish to compute their in-between distance for.

$||\dot{\gamma}(\lambda)||$ represents the magnitude of the tangent vector (of our geodesic $\gamma(\lambda)$) at some point λ on the geodesic. $d\lambda$ represents an infinitesimal step. Their product corresponds to the amount of distance covered in an infinitesimal step $d\lambda$, i.e. the infinitesimal displacement. Even though the incremental (infinitesimal steps) $d\lambda$ are uniform, the distances covered with each step may differ based on the magnitude $||\dot{\gamma}(\lambda)||$ as this is dependent on the tangent vector $\dot{\gamma}(\lambda)$ and the metric tensor.

In our case $\frac{d\gamma}{d\lambda} = [\frac{dy_1}{d\lambda}, \frac{dy_2}{d\lambda}]$ is the tangent vector. We know from equation 3.9 how the metric tensor acts on two vectors thus the squared magnitude is: $||\dot{\gamma}(\lambda)||^2 = g_D(\frac{d\gamma}{d\lambda}, \frac{d\gamma}{d\lambda}) = [\frac{dy_1}{d\lambda}, \frac{dy_2}{d\lambda}] \cdot g_D \cdot [\frac{dy_1}{d\lambda}, \frac{dy_2}{d\lambda}]^T$. We can see here that the metric g_D (see 3.10) enters and adjusts the squared magnitude. If we then take the square root and integrate over our path we arrive at the arclength described by equation 3.12.

Evaluating expression 3.12 for two points $\gamma_1 = \mathbf{a}, \gamma_2 = \mathbf{b}$ allows us to obtain the formula for the Hyperbolic Distance (in the Poincaré Disk model) between these points:

$$d(\mathbf{a}, \mathbf{b}) = \cosh^{-1}(1 + 2\frac{||\mathbf{a} - \mathbf{b}||^2}{(1 - ||\mathbf{a}||^2)(1 - ||\mathbf{b}||^2)}) \tag{3.13}$$

As a sanity check, we can use this formula to find the Hyperbolic distance between points $(0.999, 0)$ and $(0, 0)$:

$$d^{\mathcal{H}}(\begin{bmatrix} 0.999 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}) \approx 7.6004 \tag{3.14}$$

Which is larger than the Euclidean distance between these two points.

In other words, the straightest path between $(0.999, 0)$ and $(0, 0)$ on the Poincaré Disk is along the geodesic between these two points. The distance is then the arclength of this geodesic which uses the Metric Tensor for its computation.

3.3. Hyperbolic Neighbour embeddings

To embed data into Hyperbolic Space using the Poincaré Disk we must adapt t-SNE to work on the Poincaré Disk.

Our low-dimensional embeddings now reside in the Poincaré Disk, with the probability distribution used

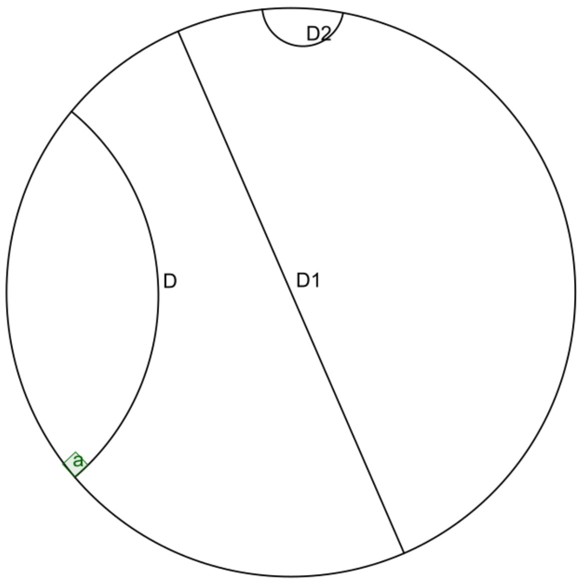


Figure 3.6: Example of geodesics on the Poincaré Disk. Note how (except for the center line), the geodesics appear curved. Furthermore, the length of infinitesimal steps along the geodesics differ from point to point. We thus can not capture the path length (or distance) between two points in the Euclidean way. (source: Jean-Christophe BENOIST, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons)

to model these embeddings adapted to use the Hyperbolic distance. This introduces changes in the low-dimensional similarities and the cost function.
Furthermore, the optimization process employed by t-SNE (originally in Euclidean Space) can not be directly used. Gradient descent steps must take into account the distortion of the Poincaré Disk. In addition, the gradient descent steps must follow the geodesics as those represent straight paths.

3.3.1. Hyperbolic t-SNE

Hyperbolic t-SNE is in its simplest form the adaptation of t-SNE to Hyperbolic Space. In this setting low dimensional embeddings are now computed with respect to the Poincaré Disk.

In its most basic form as described in [33] and in [34, 7, 12] (although the latter three contain some slight modifications but the essence remains teh same), only the low dimensional probabilities (represented by the matrix $Q^{\mathcal{H}}$ with entries $q_{ij}^{\mathcal{H}}$) differ from standard t-SNE due to the low-dimensional embedding $\mathbf{y}_i$ residing on the Poincaré Disk. High dimensional probabilities are unadjusted and appear the same as in t-SNE (see equation 3.1) since they are assumed to exist in Euclidean Space.

The low dimensional probabilities are defined using the Hyperbolic distance $d_{ij}^{\mathcal{H}} = d^{\mathcal{H}}(\mathbf{y}_i, \mathbf{y}_j)$ where $\mathbf{y}_i, \mathbf{y}_j$ are points on the Poincaré Disk (see equation 3.13):

$$q_{ij}^{\mathcal{H}} = \frac{(1 + (d_{ij}^{\mathcal{H}})^2)^{-1}}{\sum_{l \neq k}(1 + (d_{kl}^{\mathcal{H}})^2)^{-1}} \tag{3.15}$$

This means that the cost function takes the following form (note the $Q^{\mathcal{H}}$ and $q_{ij}^{\mathcal{H}}$):

$$C^{\mathcal{H}} = KL(P||Q^{\mathcal{H}}) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}^{\mathcal{H}}} \tag{3.16}$$

3.3.2. Gradient descent on the Poincaré Disk

In gradient descent we compute the direction of steepest increase (of the cost function), and update into the opposite direction as we aim to minimize our cost function.
If we were to compute this naïvely for the Poincaré Disk we get the gradient $\nabla \mathcal{L} = \frac{\partial \mathcal{L}}{\partial \mathbf{y}}$ which is a vector in the Tangent Space and tells us in what direction (and with what magnitude) to move in.
Simply stepping in that direction with its corresponding magnitude (i.e. using standard gradient descent)

does not take into account the amount of spatial distortion in the Poincaré Disk, nor the correct update path as straight lines are (curved) geodesics in the disk. Movement along a direction (defined by some vector) equals movement along the geodesic pointing in the direction of that vector, with the amount of movement adjusted according to the metric tensor (see figure 3.7). In the following sections we will describe how to correctly go about this.

Inverse Metric Tensor

The first issue is tackled by scaling the Euclidean gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{y}}$ correctly. This is achieved via the application of the inverse Metric Tensor.

In section 3.2.4 we described how the metric tensor allows us to convert distances on the disk (i.e. amounts of Euclidean distance on the disk as seen from the outside) to their actual Hyperbolic distance (the corresponding actual Hyperbolic amounts of distance in Hyperbolic Space).
In the current situation however we are faced with the inverse problem. We have the gradient (a vector in the Tangent Space) $\frac{\partial \mathcal{L}}{\partial \mathbf{y}}$ which describes the amount (magnitude) and direction (the direction will be addressed in the next section) to step in. The magnitude of this step however is in terms of Hyperbolic distance amounts as we are computing a derivative with respect to Hyperbolic coordinates $\mathbf{y}$. In other words, corresponding to what we described in section 3.2.4, this magnitude is the amount of distance someone has to travel if they were to exist in Hyperbolic Space, in the Poincaré Disk. However, on the numerical side (i.e. how gradient descent is applied) we do not work inside Hyperbolic Space. All the computations happen in standard Euclidean space[3] since our model (Poincaré Disk) is embedded in $\mathbb{R}^2$. This means that we must convert the Hyperbolic distances (distances inside the disk/inside Hyperbolic Space) to Euclidean distances. This is the inverse problem described in section 3.2.4 and thus the inverse of what the metric tensor deals with. Since the metric tensor translates distances on the disk to the corresponding actual Hyperbolic distance, the inverse metric thus translates Hyperbolic distances to the corresponding distance on the disk (on the model of the Poincaré Disk which is embedded in $\mathbb{R}^2$). This means that the correct magnitude is rescaled by the inverse metric tensor and we get the following expression:

$$\nabla_{\mathbf{y}_i} \mathcal{L} = g_{\mathcal{D}}^{-1} \frac{\partial \mathcal{L}}{\partial \mathbf{y}_i} \tag{3.17}$$

This formula gives us the magnitude and direction of the vector $\frac{\partial \mathcal{L}}{\partial \mathbf{y}_i}$ with respect to our model of Hyperbolic Space (The Poincaré Disk $\mathcal{D}$)embedded in $\mathbb{R}^2$. For example, a large $\frac{\partial \mathcal{L}}{\partial \mathbf{y}_i}$ near the border of the disk in Hyperbolic Space will be scaled down by $g_{\mathcal{D}}^{-1}$ so that a step in the direction and magnitude specified by $\frac{\partial \mathcal{L}}{\partial \mathbf{y}_i}$ does not go out of bounds of the Poincaré Disk $\mathcal{D}$ as embedded in $\mathbb{R}^2$.

The exponential map

We have obtained the correct magnitude and direction of the gradient (the update step). We must now determine the correct update path. Specifically, we must update/travel along the geodesic pointing in the direction and with the magnitude of $\nabla_{\mathbf{y}_i} \mathcal{L}$. In other words, we must update along the geodesic in the direction as specified by the gradient, by an amount as specified by that gradient.

This idea is formalized in terms of the exponential map $\mathbf{y}^{t+1} = \exp_{\mathbf{y}^t}(\mathbf{v})$. Here $\mathbf{y}^{t+1}$ is the new position vector (our destination point) and $\mathbf{y}^t$ the position vector we travel from. Furthermore $\mathbf{v} \in \mathbb{R}^2$ is a vector representing the direction we wish to travel in[4]. Therefore the exponential map is a function that takes as input our starting position $\mathbf{y}^t$, and a vector whose direction (and amount as determined by that vectors magnitude) we wish to travel in $\mathbf{v}$ from that starting position $\mathbf{y}^t$, and maps it to the corresponding destination point on the manifold $\mathbf{y}^{t+1}$. The exponential maps thus ensures we travel along the geodesic pointing in the direction of the specified vector which in our case is $\mathbf{v} = \nabla_{\mathbf{y}_i} \mathcal{L}$. However since we are minimizing a function via performing gradient descent, we take the opposite of the direction of steepest increase (adjusted by a learning rate η), i.e. we move in the direction of steepest decrease $-\mathbf{v} = -\nabla_{\mathbf{y}_i} \mathcal{L}$ adjusted by η. Thus the actual exponential map applicable to our

[3]In technical terms this is called the embedding space or ambient space. The Euclidean Space in which we have embedded our model of Hyperbolic Space (the Poincaré Disk). Not to be confused with the embedding related to how the datapoints are embedded in the Poincaré Disk

[4]To be precise, $\mathbf{v}$ lives in the Tangent Space at some point p, $\mathbf{v} \in T_p$ of our manifold (for more details please see [6, 11, 2])

case is:

$$\mathbf{y}^{t+1} = \exp_{\mathbf{y}^t}(-\eta\mathbf{v}) = \exp_{\mathbf{y}^t}(-\eta\nabla_{\mathbf{y}_i}\mathcal{L}) \tag{3.18}$$

Therefore equation 3.18 formally captures the gradient descent update step on our Riemannian Manifold (Poincaré Disk).

For the Poincaré Disk the exponential map for a vector $\mathbf{v}$ departing from point $\mathbf{y}_i$ is:

$$\exp_{\mathbf{y}_i}(\mathbf{v}) = \mathbf{y}_i \oplus (\tanh(\frac{\lambda_{\mathbf{y}_i}\mathbf{v}}{2})\frac{\mathbf{v}}{||\mathbf{v}||}) \tag{3.19}$$

Where $\lambda_{\mathbf{y}_i} = \frac{2}{1-||\mathbf{y}_i||^2}$ and $\oplus$ is defined to be the Mobiüs addition operator:

$$\mathbf{y}_i \oplus \mathbf{y}_j = \frac{(1+2\langle\mathbf{y}_i,\mathbf{y}_j\rangle + ||\mathbf{y}_j||^2)\mathbf{y}_i + (1-||\mathbf{y}_i||^2)\mathbf{y}_j}{1+2\langle\mathbf{y}_i,\mathbf{y}_j\rangle + ||\mathbf{y}_i||^2||\mathbf{y}_j||^2} \tag{3.20}$$

Please see [5] for further details regarding the above equations.

Putting everything together we have the following gradient descent update rule for points on the Poincaré Disk:

$$\mathbf{y}_i^{t+1} = \exp_{\mathbf{y}_i^t}(-\eta\, g_{\mathcal{D}}^{-1}\frac{\partial\mathcal{L}}{\partial\mathbf{y}_i}) = \exp_{\mathbf{y}_i^t}(-\eta\nabla_{\mathbf{y}_i}\mathcal{L}) \tag{3.21}$$

The generalization of gradient descent to non-Euclidean spaces is called Gradient Descent on Riemannian Manifolds [3, 2].

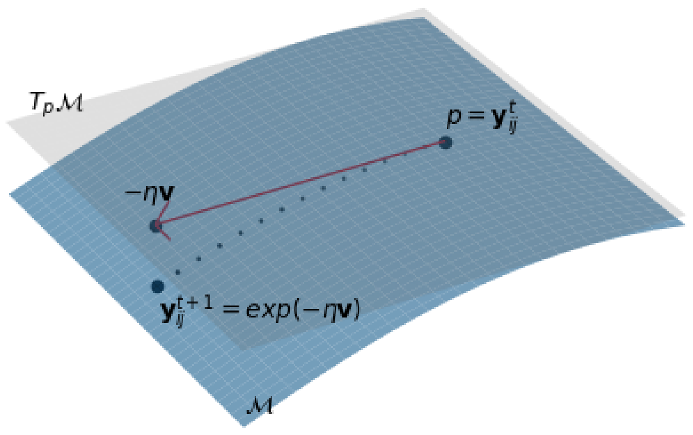


Figure 3.7: Visualization of the exponential map operation. $-\eta\mathbf{v}$ is the direction we wish to travel in on our manifold $\mathcal{M}$ (The blue surface) from point $p = \mathbf{y}_{ij}^t$. This vector $-\eta\mathbf{v}$ (represented by the red arrow) lives in the Tangent Space $T_p\mathcal{M}$ (The flat gray sheet). The exponential map correctly maps to the corresponding point $\mathbf{y}_{ij}^{t+1} = \exp(-\eta\mathbf{v})$ on the manifold $\mathcal{M}$ that would have been the result of traveling in the direction (and with the magnitude of) vector $\eta\mathbf{v}$. The corresponding path taken on the manifold is outlined by the small black dots.

4 Methods

In this chapter we discuss our contributions. Each piece of contribution (each section in this chapter) has a corresponding experiment in section 5.

We first discuss the occurrence of a mistake in the gradient of Hyperbolic t-SNE related methods in section 4.1, and provide a derivation of the correct gradient. In section 4.2 we turn our attention to how as a consequence of the t-distribution, the Poincaré Disk produces unintelligible visualizations as points are embedded near the boundary. Furthermore, we will see that the correct Hyperbolic t-SNE gradient exacerbates this problem. As a way to remedy this we propose the use of a Gaussian distribution for the low-dimensional embeddings which is described in section 4.3. Finally, in section 4.4 two ways of assessing the quality of the embeddings are discussed, providing a means to compare the embeddings of each method. This is done qualitatively by inspecting the embeddings of artificial tree-like data and quantitatively via the PR-metric.

4.1. Gradient Correction

Our first contribution is a correction to the gradient that adapts t-SNE to Hyperbolic Space. Specifically, in CO-SNE [7] and in the work of [33], a mistake can be observed in their gradient derivation. Furthermore, even though the paper describes the optimization procedure correctly, the implementation of [33] does not use the inverse metric tensor in their optimization (see section 3.3.2) resulting in an incorrect optimization procedure being employed. In the corresponding experiment (section 5.1) we take note of both the incorrect gradient and the omission of the inverse metric tensor.

As for h-SNE [34] , their method does not follow a direct adaptation of t-SNE to the Hyperbolic setting. In h-SNE, standard Euclidean t-SNE is used where the cost function is adapted with an extra term that resembles Hyperbolic t-SNE. h-SNE thus incorporates both standard Euclidean t-SNE and Hyperbolic t-SNE in one method. The gradient derivation they provide (specifically for the Hyperbolic t-SNE term as their gradient is basically the sum of Euclidean t-SNE gradient with the Hyperbolic t-SNE gradient) does however contain the correction factor that we will discuss in this section. Because their methodology differs from a standard adaptation of t-SNE to the Hyperbolic setting, we do not incorporate h-SNE in our discussions.

Finally Poincaré Maps [12] does not showcase an explicit gradient formula or derivation therefore no definitive statements can be made there.

In the following subsections we first address how the error in CO-SNE [7] and [33] likely came to be by tracing this back to the original Euclidean t-SNE gradient derivation. We then provide a proper derivation and take some steps into understanding the corrected gradient qualitatively.

4.1.1. Correct gradient formulation

Firstly, the incorrect gradient takes the following form:

$$\frac{\delta C_{incorrect}}{\delta \mathbf{y_i}} = 4\sum_{j\neq i}(p_{ij} - q^{\mathcal{H}}_{ij})(1 + {d^{\mathcal{H}}_{ij}}^2)^{-1}\frac{\delta d^{\mathcal{H}}_{ij}}{\delta \mathbf{y}_i} \tag{4.1}$$

This is the gradient used in [33]. CO-SNE [7] has an analogous gradient where $d^{\mathcal{H}}_{ij}$ is replaced by $d_{\mathbb{B}^n}(\mathbf{y}_i, \mathbf{y}_j)$ since a slightly different distribution is used. Nonetheless, in both derivations the same mistake is present[1].

The correct formulation of the Hyperbolic t-SNE gradient looks as follows (note the extra $d^{\mathcal{H}}_{ij}$ term. See equation 3.13):

$$\frac{\delta C_{correct}}{\delta \mathbf{y}_i} = 4\sum_{j\neq i}(p_{ij} - q^{\mathcal{H}}_{ij})(1 + {d^{\mathcal{H}}_{ij}}^2)^{-1} \cdot d^{\mathcal{H}}_{ij} \cdot \frac{\delta d^{\mathcal{H}}_{ij}}{\delta \mathbf{y}_i} \tag{4.2}$$

The full derivation can be found in Appendix A.

4.1.2. Origin of the mistake

We suspect that the gradient mistake originates from attempting to follow the gradient derivation of the original t-SNE paper.

Mistake in original t-SNE gradient derivation

If we follow the original t-SNE gradient derivation (as presented in their paper [16]) then we can notice that it contains mistakes[2]. t-SNE splits up the derivation of the gradient $\frac{\delta C}{\delta \mathbf{y}_i}$ up into two main terms of $\frac{\delta C}{\delta d_{ij}}$ and $\frac{\delta d_{ij}}{\delta \mathbf{y}_i}$. The exact expression is: $\frac{\delta C}{\delta \mathbf{y}_i} = \sum_j \frac{\delta C}{\delta d_{ij}} \frac{\delta d_{ij}}{\delta \mathbf{y}_i}$. t-SNE then proceeds to derive each of the two terms independently before recombining them into the final expression.

Firstly, t-SNE incorrectly arrives at $\frac{\delta d_{ij}}{\delta \mathbf{y}_i} = (\mathbf{y}_i - \mathbf{y}_j)$ instead of the correct result of $\frac{\delta d_{ij}}{\delta \mathbf{y}_i} = \frac{\mathbf{y}_i - \mathbf{y}_j}{||\mathbf{y}_i - \mathbf{y}_j||}$. Therefore in the original t-SNE derivation a factor of $\frac{1}{||\mathbf{y}_i - \mathbf{y}_j||}$ is missing. Then, in the derivation of the $\frac{\delta C}{\delta d_{ij}}$ term, t-SNE misses an extra $d_{ij} = ||\mathbf{y}_i - \mathbf{y}_j||$ term. This term enters because d_{ij} appears in the t-distribution in the form of $(1 + d^2_{ij})^{-1}$. During the t-SNE derivation of $\frac{\delta C}{\delta d_{ij}}$, an intermediate step of $\frac{\delta(1+d^2_{ij})^{-1}}{d_{ij}}$ is taken which results in $-2(1 + d^2_{ij})^{-2}d_{ij}$ by the chain rule. However, only $-2(1 + d^2_{ij})^{-2}$ appears in their derivation with d_{ij} missing. We thus conclude that the chain rule has not been correctly applied.

Finally, when we recombine these two terms, we are left dealing with a distance term $d_{ij} = ||\mathbf{y}_i - \mathbf{y}_j||$ and $\frac{\mathbf{y}_i - \mathbf{y}_j}{||\mathbf{y}_i - \mathbf{y}_j||}$. Since $||\mathbf{y}_i - \mathbf{y}_j||$ is present in both terms we can cancel them out. As a result only the $(\mathbf{y}_i - \mathbf{y}_j)$ is left over which *is* present in the original t-SNE gradient derivation. Since the rest of the derivation happens without error, these mistakes in the original t-SNE derivation do not affect the correctness of the final gradient they derive.

However this coincidence does not translate to the Hyperbolic setting as we do not have an analogous cancellation of $d^{\mathcal{H}}_{ij}$ with $\frac{\delta d^{\mathcal{H}}_{ij}}{\delta \mathbf{y}_i}$ (i.e. the Hyperbolic variants of the terms discussed).

Mistake in Hyperbolic t-SNE derivations

The mistakes in the derivations of [7, 33] have a similar flavour. In the Hyperbolic variant of the derivations, the derivative is now taken with respect to the Hyperbolic distance function instead of the Euclidean (d_{ij} in the original t-SNE derivation) distance. Both [7, 33] employ a distribution (t-distribution in [33], Cauchy distribution in [7]) containing a squared distance term (${d^{\mathcal{H}}_{ij}}^2$ in [33] and $d_{\mathbb{B}^n}(\mathbf{y}_i, \mathbf{y}_j)^2$ in [7]). In both derivations, the gradient derivation is split up into two components (analogous to the above explanation and the derivation in t-SNE). And thus in both derivations, the same mistake as made by the original t-SNE derivation is made where the chain rule is incorrectly applied where the squared distance term being incorrectly considered. This leads to a mistake where the distance factor ($d^{\mathcal{H}}_{ij}$ in [33] and $d_{\mathbb{B}^n}(\mathbf{y}_i, \mathbf{y}_j)$ in [7]) is omitted in both derivations.

[1] In CO-SNE [7] an extra term $d_{\mathbb{B}^n}(\mathbf{y}_i, \mathbf{y}_j)$ is missing whereas in [33] the missing term is $d^{\mathcal{H}}_{ij}$

[2] See https://stats.stackexchange.com/questions/301276/calculating-t-sne-gradient-a-mistake-in-the-original-t-sne-paper

4.1.3. Importance of correction

Use of the incorrect gradient means that the target cost function (equation 3.16) is not being properly optimized as the gradient used does not correspond to the gradient of the actual cost function. This also means that the embeddings produced as a consequence of using the incorrect gradient can not be guaranteed to be the expected results since the optimization procedure proceeds incorrectly.

However, both gradient formulations are very similar in mathematical form. This could mean that despite the incorrect gradient not corresponding to the cost function, the cost function itself may nonetheless be somewhat minimized.

To experimentally confirm that the usage of the correct gradient results in more optimal embeddings, we will compare cost function values between the optimization procedure of the correct gradient and the incorrect gradient (with respect to the same cost function described in equation3.16). This will be our first experiment (section 5.1). Our goal is thus to determine that the correct gradient formulation results in a more optimal embedding by comparing cost function values.

4.2. Poincare Disk limitations

The Poincaré Disk imposes some limitations on what it means for embeddings to remain visually informative. In this section we will explore what this limitation is. We also will see that the t-distribution is a major culprit when it comes to producing uninformative visualizations.

4.2.1. Poincaré Disk problems

Recall from section 3.2 that the Poincaré Disk distorts space. Even though the Poincare Disk holds all of (infinite) Hyperbolic Space, the vast majority of space that is useful for visualization in the disk (e.g. the region bounded by $\{(y_1, y_2)|y_1^2 + y_2^2 < 0.99\}$[3]) only corresponds to a very small amount of all of Hyperbolic Space. Thus most of the actual Hyperbolic Space lives around the boundary of the disk which, due to distortion, is not clearly visible to us. As we approach the boundary, space is increasingly distorted. Specifically, space becomes increasingly dense and increasingly small amounts of disk space corresponds to large amounts of Hyperbolic Space. For example, in figure 3.5b we can see how objects near the boundary of the disk appear increasingly small. Therefore if we were to embed points there, similar distortion will be present in our embeddings. This leads to hard-to-interpret visualizations.

Embeddings that correspond to useful visualizations should inhabit the majority of visible space (which is a tiny amount of actual Hyperbolic Space). In practice this corresponds to regions of the Poincaré Disk where there is relatively little distortion (e.g. $\{(y_1, y_2)|y_1^2 + y_2^2 < 0.99\}$). If the majority of the embeddings do not occupy this space (and instead are pushed to or occupy the boundary) then those resulting embeddings may not be visually informative as things become heavily distorted. Hyperbolic structures (such as hierarchy, tree-like structures) are hard to observe for points that are visually on top of each other (densely packed due to the distortion of the Poincaré Disk) despite these points having possibly large distances between them. Such points may appear as a dense cluster visually whereas mathematically they do not have to be.

In section 5.2 we will see several examples of this. Furthermore, from this point onward we will refer to the region in the Poincaré Disk where visualizations remain informative (e.g. the region bounded by $\{(y_1, y_2)|y_1^2 + y_2^2 < 0.99\}$ as described above) as visually relevant/visually informative/visualizable regions of the Poincaré Disk. In a similar light, we will refer to embeddings whose points are pushed towards the boundary of the Disk as visually uninformative/unintelligible embeddings.

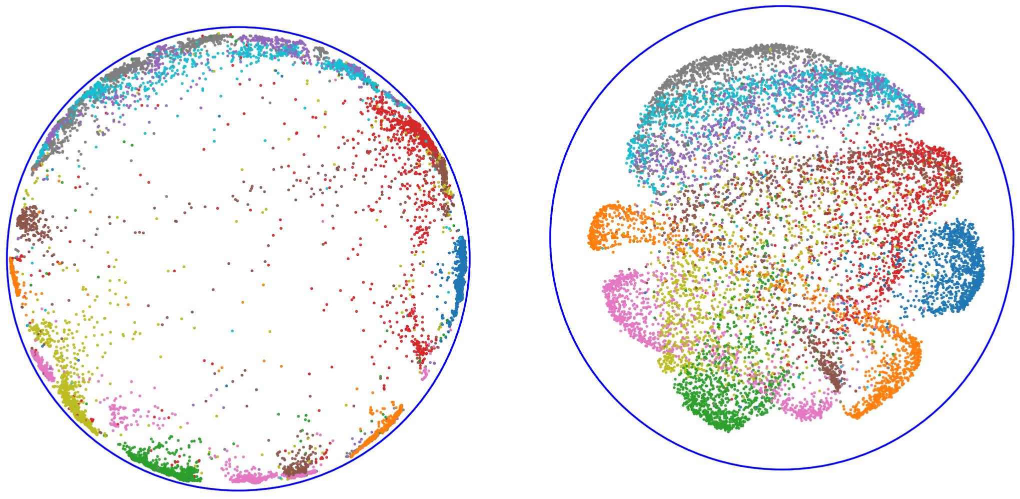
Poincaré Disk distortion example

To further highlight this point, consider the following example that highlights embedding issues arising from the distortion in Poincaré Disk quantitatively:

The distance between two points on disk $\mathbf{y}_1 = (-0.999, 0)$ and $\mathbf{y}_2 = (0.999, 0)$ is:

$$d^{\mathcal{H}}(\mathbf{y}_1, \mathbf{y}_2) = d^{\mathcal{H}}(\begin{bmatrix} -0.999 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.999 \\ 0 \end{bmatrix}) \approx 15.2008 \tag{4.3}$$

[3]Our choice for the value 0.99 is purely heuristical. The main takeaway is that we can not visually discern between points as we approach the boundary of the disk. For example, points between a radius of 0.99 and 0.999 do not appear different visually.



(a) MNIST embedding in the Poincaré Disk using the correct Hyperbolic t-SNE gradient. Note how all the points are pushed towards the boundary. The resulting embedding is not very visually informative.

(b) MNIST embedding in the Poincaré Disk using the SNE gradient (discussed in section 4.3). The embedding is much more visually informative as they now occupy the majority of the visually relevant region of the Poincaré Disk. Furthermore, some structure/relationships can be seen in the visualization.

Figure 4.1

Which can roughly represent the diameter of the Poincaré Disk as it is visually difficult to distinguish between a point such as $(0.999, 0)$ and a boundary point $(1, 0)$.

Next, consider the distance between the following two points:

$$d^{\mathcal{H}}(\begin{bmatrix}-0.999\\0\end{bmatrix}, \begin{bmatrix}-0.99999999999\\0\end{bmatrix}) \approx 16.1181 \tag{4.4}$$

In the visualization, the two points $(-0.999, 0)$ and $(-0.99999999999, 0)$ might as well represent the same point since we can not visually discern between them accurately enough.

However, in terms of distance on the Poincaré Disk, there is more distance between $(-0.999, 0)$ and $(-0.99999999999, 0)$, two points that are visually on top of each other, than $(-0.999, 0)$ and $(0.999, 0)$, two points that are visually on opposite sides of the disk. This means that points embedded on top (or close to on top) of each other, near regions of space at the boundary of the disk, may have similar relationships to each other as points on opposite sides of the disk.

The takeaway from this is that mathematically legitimate embeddings where points live near the boundary do not give useful visualizations as visually these points may sit on top of each other. The most informative visualizations are ones where structure can be observed. This is only possible if we limit ourselves to a region such as $\{(y_1, y_2)|y_1^2 + y_2^2 < 0.99\}$ (here 0.99 is arbitrarily chosen) which corresponds to a small amount of Hyperbolic Space. Embeddings in this region are not as strongly affected by the distortion (compared to points nearing the boundary), and thus (visually) points in this region has more room, leading to less densely packed embeddings, resulting in more informative visualizations.

4.2.2. t-distribution limitations

We have so far seen that the most informative visualizations must attempt to keep embeddings in a certain region of the Poincaré Disk. Or phrased differently, embeddings should be prevented from approaching the border[4] as then the visualization may become unintelligible.

The usage of the t-distribution in the low-dimensional embeddings exacerbate the issues mentioned in the previous section. Originally introduced to combat the Crowding Problem (see section 3.1.3) by

[4]to remain technically correct; the Poincaré Disk does not have a border as it is supposed to represent all of infinite Hyperbolic Space. However our model (which is embedded in Euclidean Space) does appear to have a border. For illustrative purposes we therefore refer to this as the border or boundary.

forcing larger distances to be established between points, it now may push points too strongly towards the boundary leading to uninformative visualizations.

In standard t-SNE this behaviour is desirable since it forces embeddings to spread out more. In Euclidean space there is no distortion and all of the Euclidean plane can in essence be used for visualization since we can simply ”zoom” out for larger embeddings/visualizations. This is not the case for the Poincaré Disk. We can not do analogous ”zooming” without distortion[5]. By introducing a mismatch in distributions used for modeling high dimensional affinities and low dimensional ones, larger distances between points can be enforced so that the resulting embeddings are less cluttered.

However in the Poincaré Disk, this feature makes it so that we *run out of visualizable space*. Even though mathematically speaking there is nothing wrong or less optimal about using the t-distribution. Visually speaking the t-distribution, which encourages larger distances to be taken between embedding points, exacerbates the boundary pushing behaviour and may lead to uninformative visualizations (see figure 4.1a).

Finally, recall that correct Hyperbolic t-SNE gradient contains an extra factor of $d_{ij}^{\mathcal{H}}$ which if greater than 1, scales up the gradient resulting in an even greater amount of boundary pushing behaviour. Note that this is less prominent in the incorrect gradient version as this factor does not exist. Therefore the correction discussed in section 4.1 may exacerbate this issue further.

4.2.3. Moving on from t-distribution

From the previous discussions we can thus conclude that a method must be sought for that allows us to keep the embeddings relatively compact (within the part of the Poincaré Disk where visualizations are still meaningful). One straightforward way to achieve this is by using a distribution for $q_{ij}^{\mathcal{H}}$ with smaller tails. This leads to points being more closely embedded together which may prevent points form approaching the boundary. As a result visually interpretable results may be obtained.

Obvious candidates are distributions with exponentially decaying tails (similar to a Gaussian distribution). This is because we are searching for a function that maps distances to very low or possibly 0 similarity values faster than the t-distribution. A distribution with exponentially decaying tails fits this narrative. In the next section we motivate and discuss the choice of replacing the t-distribution with a Gaussian distribution. Although other options are possible (such as a bump function), the Gaussian distribution has been used in methods such as SNE [9] making it a natural choice to consider.

4.3. Gaussian distribution

To remedy the issues brought up in the previous section, we consider using the Gaussian distribution instead of the t-distribution. The exponentially decaying tails characteristic to the Gaussian distribution results in embeddings that are more strongly clustered together. As points are less encouraged to take larger distances from each other we hope to prevent points from being pushed towards the boundary, resulting in more visually informative embeddings. Going forwards we shall name this method Hyperbolic SNE. The use of a Gaussian distribution to model low-dimensional embeddings may remind the reader of the progenitor of t-SNE: SNE [9].

Finally, the Gaussian distribution introduces a variance parameter σ^2 which allows the user control over the spread of the embedding. This gives us an additional way in which we can confine embeddings to the visually relevant regions of the Poincaré Disk. In our third set of experiments (section 5.3) we will provide the experimental evidence for the claims made in this section.

[5]To be more precice: Hyperbolic Space, if we were able to visualize it fully like Euclidean space, would allow for an analogous form of zooming (in technical terms, Hyperbolic Space is homogeneous and isotropic). However we can only visualize Hyperbolic Space via models that can not capture the space in full. Any form of zooming will therefore give rise to distortion. In technical terms, this is because the Poincaré Disk is a conformal (angle preserving but distance distorting) map of Hyperbolic Space.

4.3.1. Hyperbolic SNE gradient

The use of the Gaussian instead of the t-distribution means the formula of the gradient changes. The Gaussian gradient is:

$$\frac{\delta C^{\mathcal{H}}}{\delta \mathbf{y}_i} = 2\sum_j \frac{\delta C^{\mathcal{H}}}{\delta r^{\mathcal{H}}_{ij}} \frac{\delta r^{\mathcal{H}}_{ij}}{\delta \mathbf{y}_i} = \frac{2}{\sigma^2} \sum_j (p_{ij} - q^{\mathcal{H}}_{ij}) \cdot d^{\mathcal{H}}_{ij} \cdot \frac{\delta d^{\mathcal{H}}_{ij}}{\delta \mathbf{y}_i} \tag{4.5}$$

A full derivation can be found in appendix B.

4.3.2. Gradient analysis

We can deduce the behaviour of the Gaussian gradient by comparing its formula with the Hyperbolic t-SNE gradient. We have already conceptually described this. Below we connect that description with the mathematics involved.

Formulaic dissimilarity

In terms of formulaic dissimilarity, the only relevant difference[6] is that the Hyperbolic t-SNE gradient (equation 4.2) contains the factor $(1 + {d^{\mathcal{H}}_{ij}}^2)^{-1}$ while the Gaussian gradient does not have such a factor. The Gaussian gradient thus does not have a multiplicative term that scales inversely with $d^{\mathcal{H}}_{ij}$. This means that the effect of distance between points on the gradient affects the Gaussian version more strongly. However, this is mitigated by the Gaussian distribution itself.

Gaussian probability distribution term

The most relevant difference is that $q^{\mathcal{H}}_{ij}$ now uses a Gaussian distribution and not a t-distribution. Formally this means:

$$q^{\mathcal{H}}_{ij} = \frac{\exp{(-{d^{\mathcal{H}}_{ij}}^2/2\sigma^2)}}{\sum_{k \neq \ell} \exp{(-{d^{\mathcal{H}}_{k\ell}}^2/2\sigma^2)}} \tag{4.6}$$

As mentioned before, since the Gaussian distribution has exponentially decaying tails, the forces propagate less far compared to using the t-distribution. This means the given the same Hyperbolic distance $d^{\mathcal{H}}_{ij}$, $q^{\mathcal{H}}_{ij}$ in the Gaussian case will be smaller than $q^{\mathcal{H}}_{ij}$ in the t-distribution case since the numerator of the Gaussian $\exp{(\frac{-{d^{\mathcal{H}}_{ij}}^2}{2\sigma^2})}$ goes to 0 faster than the t-distribution's numerator $(1 + {d^{\mathcal{H}}_{ij}}^2)^{-1}$ (see figure 4.2). As a result, embeddings will be pushed out less far since $p_{ij} \approx q^{\mathcal{H}}_{ij}$ for much smaller distances $d^{\mathcal{H}}_{ij}$ in the case of the Gaussian (see figure 4.1b).

4.3.3. Tuning the variance

Using the Gaussian distribution in the Embedding Space results in an additional hyperparameter to tune. Namely, the variance σ^2 of the distribution. In the original SNE paper a variance of $\sigma^2 = \frac{1}{2}$ is used[7]. In our work we also set this value to a constant. However, for optimal embeddings experimentation with the σ^2 value is recommended.

Since we must fit a dataset in a limited amount of space (as described in section 4.2), this σ^2 parameter provides us the means to control the spread of our embeddings. Setting σ^2 too big can result in data being pushed towards the boundaries nonetheless. Setting σ^2 too small could result in embeddings not spreading out at all, concentrating around the origin.

[6]Note that the constant terms (4 for Hyperbolic t-SNE and $\frac{2}{\sigma^2}$ for the Gaussian) are practically irrelevant as their contribution, being simply a scalar, can be mitigated by adjusting the learning rate. Furthermore, if we set $\sigma^2 = \frac{1}{\sqrt{2}}$, these constants become the same.

[7]this is the variance used for the Gaussians in the low-dimensional space. Not to be confused with the high-dimensional Gaussian distributions whose variance is found via the user-defined perplexity value and binary search.

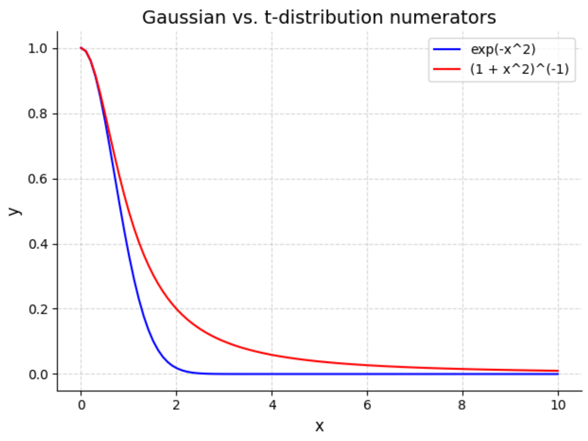
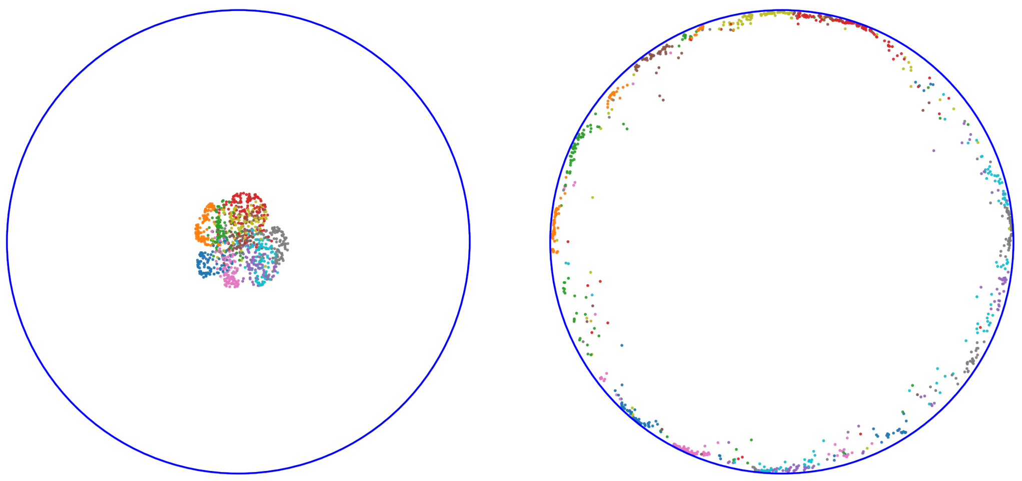


Figure 4.2: We can see that the Gaussian (blue) goes to 0 (decays) much faster than the t-distribution (red). It is this property that makes the Gaussian useful in modeling the similarities as only small distances will be mapped to relatively high probabilities. Remaining distances are quickly mapped to 0. This means embeddings will be more densely clustered as points take smaller distances from each other, resulting in points less pushed towards the boundary (of the visually relevant region of the Poincaré Disk)



(a) MNIST embedding on the Poincaré Disk using Hyperbolic SNE with $\sigma^2 = 0.01$. The embeddings are clustered in the center due to too low of a variance.

(b) MNIST embedding on the Poincaré Disk using Hyperbolic SNE with $\sigma^2 = 5$. Embeddings are pushed to the boundary due a high variance.

Figure 4.3

To summarize, using the Gaussian distribution gives us the means to obtain more visually informative embeddings as it can prevent points from being pushed towards the boundary. This is because the Gaussian forces points to take smaller distances from each other. Additionally, The use of the Gaussian distribution gives us a means of controlling the spread of the embeddings via the variance σ^2 parameter. This allows the user to adjust the embedding process per dataset.

Finally, since we are forcefully restricting embeddings to only occupy a certain amount of space, i.e. the region of the Poincaré Disk that is visually meaningful, one may question whether this induced constraint may affect the embeddings negatively. After all, we are forcing the embeddings to only occupy a small amount of Hyperbolic Space (corresponding to the part on the Poincaré Disk that is visually informative to us). We will not tackle this question but we will see in section 5.3 that structures still emerge and deliver visualizations more interpretable than allowing points to occupy and be pushed onto the boundary.

4.4. Quality assessment

So far we have proposed various changes to standard Hyperbolic t-SNE (as outlined in [7, 33]). As our final contribution, we will assess the quality of each method (standard Hyperbolic t-SNE with the incorrect gradient, with the correct gradient, and Hyperbolic SNE) by means of a qualitative assessment and a quantitative one (besides the embeddings of the datasets produced by each respective method). We will compare results obtained from the proposed quality assessments (discussed below) across the three methods and compare them (see experiments 5.4 and 5.5).

4.4.1. Hierarchical embeddings

Our first form of assessment will be in the form of embedding an artificial hierarchical tree-like dataset.

Recall that the main purpose of embedding data in Hyperbolic Space is to capture Hierarchical and/or tree-like (see section 3.2.1). To test this explicitly we will attempt to embed an artificial tree-like dataset (of various sizes) using methods proposed thus far. We can compare how well each method captures hierarchical structures.

Specifically, we create several tree-like datasets where clusters of points form a node of the tree, with each such cluster connected to the other clusters in a tree-like fashion. The implementation in Python can be found in appendix C

By embedding tree-like data in the Poincaré Disk we can explicitly test that Hyperbolic embeddings are indeed effective at capturing hierarchical and tree-like structures. Furthermore, this provides a qualitative way of comparison between the different Hyperbolic t-SNE methods described. It allows us to assess which methods performs better by visually inspecting for such structures. We can therefore compare how well each method captures these structures (see the experiments in section 5.4 for details).

4.4.2. Measuring (exact) neighbourhood preservation

Despite the primary focus of Hyperbolic (t-)SNE being visualizations, solely relying on a qualitative assessment (visualizations) has limitations. We will therefore use the Precision-Recall (PR) metric [27] as another way of assessing quality.

The Precision-Recall metric

The Precision-Recall metric is a measure for how faithfully neighbourhoods are recreated in the low dimensional embedding relative to the high dimensional data. For example, if we take some point and its neighbourhood (set of closest N points to our chosen point) in high dimensions, and we find that points' corresponding embedding location and neighbourhood in the Poincaré Disk (low-dimensional embedding); then we can compare both sets and determine whether the neighbours match. If all the neighbours match then a neighbourhood is faithfully reconstructed across the embedding procedure. This is then computed for various (possibly all) points X's to obtain an estimate of how well neighbourhoods are reconstructed.

Note that it may not be possible for embeddings to faithfully reconstruct neighbourhoods to the fullest extent, nor does t-SNE necessarily find this solution.

Mathematical details

To calculate the Precision-Recall metric we start by finding a neighbourhood of size k_{max} around some point $\mathbf{x}_i \in \mathbb{R}^n$ in high dimension called $N_{k_{max}}(\mathbf{x}_i)$. We then find neighbourhoods of size $k = \{1, 2, 3, ..., k_{max}\}$ in the low-dimensional embedding of the corresponding point $\mathbf{y}_i \in \mathcal{D}$ (the low-dimensional embedding of point $\mathbf{x}_i$). We denote this neighbourhood as $N_k(\mathbf{y}_i)$.

In practice, we use the indices of the neighbouring points and not the neighbouring points itself for this calculation. The high-dimensional datapoints, and the low-dimensional embedding points are stored as matrices (or arrays). This means that each point is addressable by their index in their corresponding array. Furthermore, the high-dimensional datapoint with index i corresponds directly to the low-dimensional embedding point with index i (i.e. $\mathbf{x}_i$ is embedded as point $\mathbf{y}_i$, they are the same point except that $\mathbf{x}_i$ is that point in the original high-dimensional dataset, and $\mathbf{y}_i$ is that same point but in the low-dimensional embedding, the Poincare Disk $\mathcal{D}$). This means that a comparison involving the indices is meaningful since the indices can substitute for the points $\mathbf{x}_i$ and $\mathbf{y}_i$ themselves as there is a direct correspondence between them. Thus we have that the sets $N_{k_{max}}(\mathbf{x}_i)$ and $N_k(\mathbf{y}_i)$ contain the indices of the neighbouring points in each respective domain, for the same point, addressable by index i.

If we do not use indices but rather the neighbouring points themselves then the intersection is always empty. This is because the elements of $N_{k_{max}}(\mathbf{x})$ are drawn from the set $\mathbb{R}^n$, while the elements of $N_k(\mathbf{y})$ are drawn from $\mathcal{D}$. These are inherently different objects and therefore the intersection of sets containing only objects of each kind respectively is empty.

After obtaining $N_{k_{max}}(\mathbf{x}_i)$ and $N_k(\mathbf{y}_i)$ we compute their intersection for each $k \in \{1, 2, 3, ..., k_{max}\}$ as $TP_k = N_{k_{max}}(\mathbf{x}_i) \bigcap N_k(\mathbf{y}_i)$. This results in a set containing the indices (which correspond to points) that are present in the neighbourhoods of both $\mathbf{x}_i$ and $\mathbf{y}_i$ for each k. In essence, we find out which points that were originally neighbours remain neighbours (after the embedding).

Precision

The Precision metric is defined as $PR_k = \frac{TP_k}{k}$. Since k is the most amount of neighbours X and Y can currently share, $\frac{TP_k}{k}$ thus measures how close the actual neighbourhoods are to the most optimal scenario. By computing this for increasing k, the Precision metric thus measures how faithfully neighbourhoods are preserved as a function of k. We can use this to well neighbourhoods are (or remain) preserved as we consider bigger and bigger neighbourhoods (as we increase k).

Recall

The Recall metric is defined as $RC_k = \frac{TP_k}{k_{max}}$ which gives us the same numerator but now it is taken relative to the maximum number of points (neighbourhood size) being considered k_{max}. Ideally we want to observe a linear increase in Recall as k increases from 1 to k_{max}. This linear increase denotes that neighbourhood preservation is independent from neighbourhood size choice. As we choose bigger and bigger neighbourhoods (up to k_{max}), we hope that RC_k increases proportionally. RC_k therefore can show us how consistent/ideal this growth is with respect to an optimal growth (linear growth with rate $\frac{1}{k_{max}}$).

Finally, we can plot both the Precision PR_k and Recall RC_k against each other to gain a sense of how well neighbourhoods are preserved as a function of k. For the full details please see the experiments in section 5.5.

5 Experiments

In this chapter we build on the points discussed in the Method chapter by providing experimental evidence for the claims made.

In our experiments we will make use of the following datasets chosen in accordance to previous works [25, 33, 7]:

(1) MNIST [17], a dataset based on handwritten digits; 70000 in size with a dimensionality of 784 per data item. **(2)** The C. Elegans [37] dataset containing the cell atlas of the major cell types of the worm C. Elegans, produced via Single-cell RNA sequencing. This dataset contains 89000 items with a dimensionality of 20222. **(3)** Planaria [20]; A dataset consisting of the cell atlas of the regenerative planarian "Schmidtea mediterranea". The dataset is 21612 datapoints large with a dimensionality of 50. **(4)** WordNet [24]. A large lexical database of English nouns, verbs, adjectives, and adverbs. Specifically we use the transitive closure of the mammals subset of Wordnet (see `https://github.com/facebookresearch/poincare-embeddings`)). This dataset contains 82115 items with a dimensionality of 11. In addition, we will make use of an artificially generated dataset (see section 4.4).

5.1. Justifying the Correct Gradient

In section 4.1 we made the observation that the incorrect gradient had been used in Hyperbolic t-SNE related methods. In this experiment we compare cost function values between embeddings produced by the correct and incorrect gradient to experimentally verify that the proposed correction indeed performs better.

Firstly, the cost function we are minimizing is:

$$C = KL(P||Q^{\mathcal{H}}) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}^{\mathcal{H}}} \tag{5.1}$$

Both the incorrect 4.1 and the correct 4.2 gradients attempt to optimize this cost function. Additionally, note that we assume the usage of the t-distribution for modeling the low-dimensional embeddings (see section 3.3.1).

As mentioned in section 4.1, the implementation of Hyperbolic t-SNE in [33] does not take into account the inverse metric tensor $g_{\mathcal{D}}^{-1}$ in their gradient descent implementation, resulting in an incorrect application of Riemannian gradient descent. To be specific, their implementation performs update step according to $\mathbf{y}^{t+1} = \exp_{\mathbf{y}^t}(-\eta \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{y}_i})$ instead of $\mathbf{y}^{t+1} = \exp_{\mathbf{y}^t}(-\eta \cdot g_{\mathcal{D}}^{-1} \frac{\partial \mathcal{L}}{\partial \mathbf{y}_i})$. We will also take this into account for our first experiment. Please see the next section for the details.

Starting from section 5.2 experiments will always use the correct update step, i.e. incorporate the inverse metric tensor in the update step (the exponential map).

5.1.1. Experimental Setup

We embed data in the Poincaré Disk via Hyperbolic t-SNE (see section 3.3.1) using the correct and incorrect gradient (section 4.1). Three different embeddings are then produced and their cost function values are compared side-by-side. We look at the embedding procedure using the incorrect gradient without the inverse metric tensor factor in the optimization step (the version of Hyperbolic t-SNE in the implementation of [33]), the incorrect gradient with the inverse metric factor (as laid out in [7] and in the paper [33]), and finally the correct gradient with the inverse metric tensor factor (our contribution).

Experiment details

Prior to the experiments an initial embedding of the dataset and the distance matrix (corresponding to high-dimensional distances) must be computed. This becomes the starting point for the embedding process.

Data pre-processing

Due to hardware limitations we first pre-process the data. We use PCA to reduce the dimensionality of the data to 50 (if the data's dimensionality is less than 50 the original data is used as is). Note that besides improvements in computational cost, other methods (including t-SNE) also employ PCA to initially reduce the dimensionality as this may reduce noise in the data without severely distorting interpoint (the distances in between any two datapoints) distances [16].

Furthermore, a sample of 10000 points is taken (if the data has more than 10000 datapoints). For each dataset and corresponding cost function experiment the same sampled data is used to ensure consistency across the dataset.

Computing high-dimensional similarities

We then compute the high-dimensional distance matrix. Firstly, the perplexity parameter must be set. In all the experiments a fixed perplexity of 30 was chosen. This parameter can be interpreted as a measure of the effective number of neighbours a datapoint has (see section 3.1). Our choice for $k = 30$ is in accordance to the value chosen in [33] and is chosen to be within the range 5 to 50 of what t-SNE calls typical values for perplexity [16].

Initialization in the Poincaré Disk

Finally, an initial embedding is created by randomly positioning the datapoints we wish to embed within a circle (in our case with radius $r = 1e - 4$) around the origin of the Poincaré Disk.

5.1.2. Hypothesis

The expectation is that using the correct gradient minimizes the cost function better, i.e. achieves lower cost function values compared to the incorrect gradient. Furthermore, the optimization steps without the inverse metric tensor is expected to perform the poorest. Since the gradients are not scaled properly (due to the lack of the inverse metric tensor), update steps will step out of bound of the Poincaré Disk. In the implementation, update steps that would result in embeddings stepping out of bounds are not performed at all. Thus we expect the version without the inverse metric tensor to terminate quickly.

Finally, in our graphs below we exclude the 250 early exaggeration (see section 3.1.1) steps as they do not provide much information regarding the convergence of the methods.

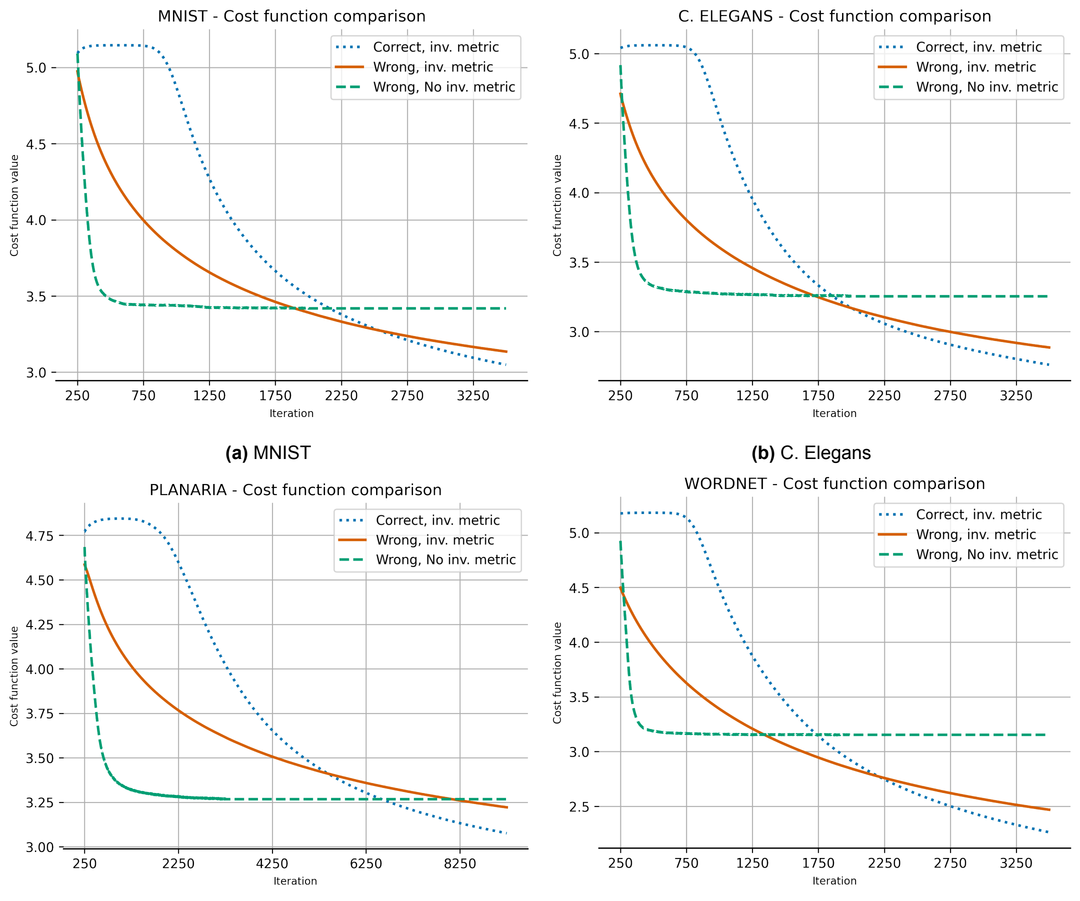
5.1.3. Results

Figure 5.1 contains a series of graphs that compares the cost function value of the correct and incorrect gradient (with and without the inclusion of the inverse metric tensor in its optimization) for several datasets.

5.1.4. Discussion

Incorrect gradient without inverse metric tensor

The embedding produced as a result of the incorrect gradient without the metric tensor present in its optimization (the green/striped curve) achieves the worst cost function values out of the three methods. This occurs consistently across all the datasets tested.



(a) MNIST **(b)** C. Elegans **(c)** Planaria **(d)** Wordnet

Figure 5.1: Cost function comparison graphs

Furthermore, the corresponding curve (green/striped) plateaus early as this method terminates early on due to embeddings getting too close to the boundary. As discussed in the Hypothesis section, not using the inverse metric tensor in the optimization procedure results in optimization steps that go out of bound of the disk, therefore not performing the update step at all. As a result the embedding procedure terminates early as no progress can be made. We are thus stuck with whatever embedding was produced before termination.

Incorrect gradient with inverse metric tensor

The continuous/brown-orange curve represents the cost function values of the embedding procedure using the incorrect gradient with the inverse metric tensor. This curve continues steadily throughout the whole embedding process as now the inverse metric tensor is present in the optimization. Update steps are scaled accordingly as embeddings approach the boundary of the disk. However, we can observe that this curve is eventually overtaken by the dotted/blue curve (the correct gradient). This indicates that the incorrect gradient does not minimize the cost function optimally which makes sense as its gradient does not correctly pertain to the cost function we are optimizing.

Correct gradient with inverse metric tensor

Finally, the correct gradient version performs the best. This makes mathematical sense as it makes use of the correct gradient corresponding to the cost function. We now have additional experimental evidence to verify this.

However initially, during the early iterations, this curve pertaining to the correct gradient (dotted/blue curve) first increases (indicating that the cost function values are increasing) before plateauing (indicating that the cost function values are not changing) and only hereafter, decreases. Furthermore, it is only after some amount of iterations that the correct gradient achieves better cost function values than the others. We will explain this below.

Initial behaviour of the correct gradient

The initial increase in cost function values is suspected to be a consequence of the early exaggeration iterations combined with optimizing via gradient descent. Once regular optimization proceeds, the configuration of embeddings resulting from the early exaggeration is likely in a poor state. Although clusters may have already formed, the overall embedding is bad. For the optimization procedure to then make progress it has to break out of this poor state which could lead to this initial increase in cost function values.

The plateauing of the cost function can be explained by recalling that an extra $d_{ij}^{\mathcal{H}}$ term is present in the gradient formula (see equation 4.2). At the beginning of the embedding procedure points occupy the center of the disk. This results in $d_{ij}^{\mathcal{H}} \approx 0$ for all points i, j since interpoint distances are extremely small. Therefore updates early on happen very slowly and can only pick up once enough distance between points has been established. Once this initial stage has been overcome, the curve pertaining to the correct gradient has a greater slope than the other curves. This indicates that the correct gradient minimizes the objective at a higher rate. Eventually the curve of the correct gradient overtakes the other curves indicating that more optimal (lower) cost function values are attained.

We thus conclude that the proper way to use Hyperbolic t-SNE (as described in [7, 33]) is by using the correct gradient in the optimization.

5.2. Limitations of the t-distribution

In section 4.3 we described how choices in distributions modeling low-dimensional similarities can greatly impact embedding quality. This experiment aims to experimentally verify that claim. We will produce several embeddings using Hyperbolic t-SNE and qualitatively assess these resulting visualizations. Both the correct and incorrect Hyperbolic t-SNE gradient will be used in these embeddings. This is because we want to show that the limitations of the t-distribution (boundary pushing behaviour/uninterpretable visualizations) is independent of the gradient formulation. Rather, it is due to the t-distribution itself.

5.2.1. Experimental Setup

In our first experiment we ran an embedding procedure for each kind of cost function (correct/incorrect) for each kind of dataset considered. In the first experiment only the cost function values corresponding to each embedding procedure were used as that was the only piece of information relevant for that section. In our second experiment we take the resulting embeddings corresponding to each of these embedding procedure (instead of the cost function values) and qualitatively inspect them. In other words, the same embedding procedure is behind the experiment results in the first and second experiment.

5.2.2. Hypothesis

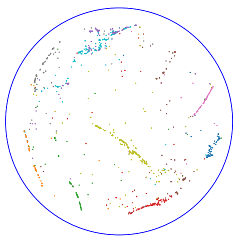
We expect that (in the limit), embeddings will be pushed outwards onto the boundary of the disk due to the nature of the t-distribution's heavy tails (as described in section 4.2). This behaviour is expected to occur for both the incorrect and correct gradient embeddings since it relates to the distribution used.

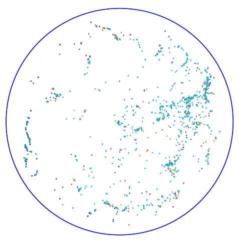
In addition, we also expect that the version with the correct gradient pushes points more outwards or away from each other. This is because an extra distance based term $d_{ij}^{\mathcal{H}}$ is present which if greater than 1, exacerbates this pushing effect since the gradient is scaled by a factor greater than 1.

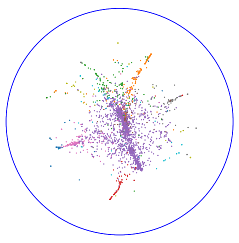
5.2.3. Results

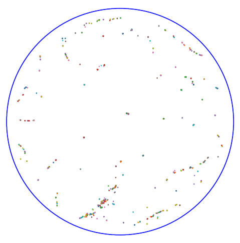
Figure 5.3 contains side-by-side embeddings resulting from using the incorrect and correct gradient for several datasets. Each embedding has a direct correspondence to their related cost function curve in the first experiment, i.e. the cost function graphs tracks the cost function values of the embedding procedure that produced the embeddings below. In addition to the final embeddings, we showcase here intermediate embeddings.

Every step in the optimization procedure has a corresponding embedding (i.e. $\mathbf{y}_i^t$, see section 3.3.1). The final embedding corresponds to $\mathbf{y}_i^{t_{max}}$ where t_{max} is the number of iterations reached until our convergence criteria is met (therefore corresponding to the final embedding). We can also take the embeddings of iterations before convergence has been reached to gain insight into how the embedding process proceeds. Some intermediate embeddings are displayed below.

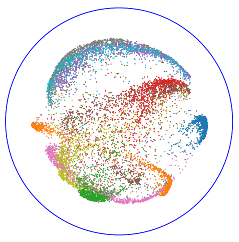

(a) MNIST, $t = 1166$ iterations

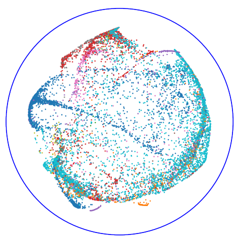

(b) C. Elegans $t = 1166$ iterations

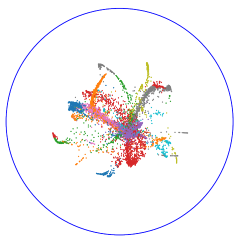

(c) Planaria, $t = 1166$ iterations

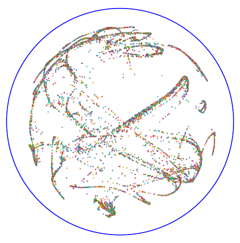

(d) Wordnet, $t = 1166$ iterations

Incorrect gradient, intermediate embeddings

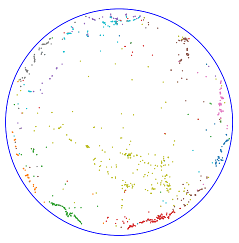

(e) MNIST, $t = 1166$ iterations

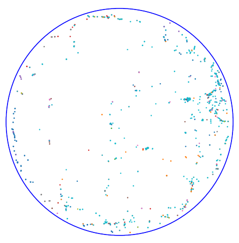

(f) C. Elegans, $t = 1166$ iterations

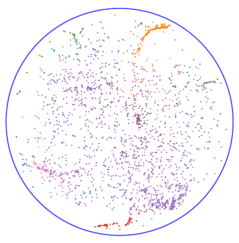

(g) Planaria, $t = 2915$ iterations

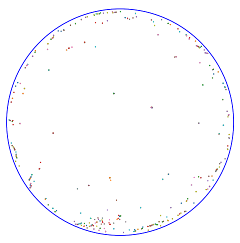

(h) Wordnet, $t = 1166$ iterations

Correct gradient, intermediate embeddings

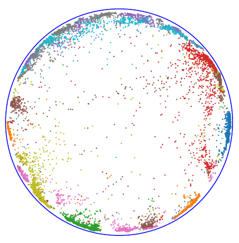

(i) MNIST, $t = 3414$ iterations

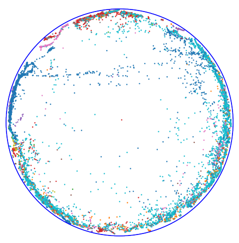

(j) C. Elegans, $t = 3414$ iterations

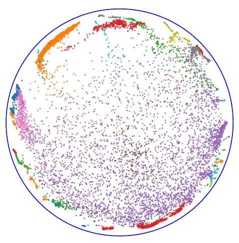

(k) Planaria, $t = 8247$ iterations

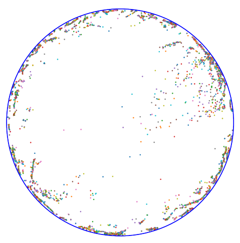

(l) Wordnet, $t = 3414$ iterations

Incorrect gradient, final embeddings


(m) MNIST, $t = 3414$ iterations


(n) C. Elegans, $t = 3414$ iterations


(o) Planaria, $t = 8247$ iterations


(p) Wordnet, $t = 3581$ iterations

Correct gradient, final embeddings

Figure 5.2: Hyperbolic t-SNE embeddings

5.2.4. Discussion

In both versions (both the correct and incorrect gradient versions) we see that the final embeddings are pushed towards the boundary. Any useful structures we may hope to observe is lost, especially for the datasets other than Planaria.

In the intermediate embeddings of the correct gradient we do see some structure emerging. However this structure is lost when we take the final/converged (after the optimization procedure has finished) embeddings. These structures appear similar to the structures that Hyperbolic SNE is capable of capturing and retaining which we will discuss in the next section.

However, as the embedding proceeds, any structure is eventually lost due to the t-distribution. The usage of the t-distribution means the embeddings do not converge until a relatively large distance, a distance too large in the context of embeddings in the Poincaré Disk to result in visually informative results, is established between points.

Finally, qualitatively the embeddings of the same data between the two gradients versions differ. The embeddings using the incorrect gradient appear much less dense. It appears that similar nodes are embedded on top of each other such that in the visualization they appear extremely close to one another. This gives the illusion that there are less datapoints in the embedding. This can be explained using the correction factor $d_{ij}^{\mathcal{H}}$.

During early exaggeration, attractive forces are amplified and thus clusters are formed. When regular optimization starts these forces are no longer amplified. Attractive forces actually play a smaller role now The attractive forces only apply between points that are similar. Such points naturally approach each other due to the applicable forces. The more similar two points are, the closer these points are to be embedded to each other. For embeddings produced by the correct gradient, the factor $d_{ij}^{\mathcal{H}}$ becomes very small. This means that very similar points can not get extremely close as that would mean $d_{ij}^{\mathcal{H}} \approx 0$ resulting in the attractive force going to 0. In other words, the contribution of the attractive forces to the (correct) gradient become extremely small as similar points approach one another. The incorrect gradient embedding does not have this factor thus points can get as close as they like resulting in points possibly being on top of each other.

5.3. Introducing the Gaussian distribution

In section 4.3 we noted that the use of distributions with tails that decay faster may resolve the problems that arise from using the t-distribution. In this section we experimentally show that the use of the Gaussian distribution does in fact overcome the discussed limitations.

5.3.1. Experimental Setup

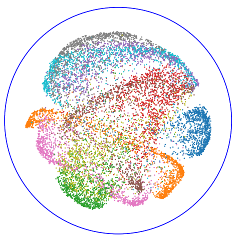
We embed the same datasets used in the previous experiments using the Gaussian gradient as described in section 4.3. This means low-dimensional similarities are now modeled using the Gaussian distribution. In our experiments a σ^2 value of 0.2 was used for every dataset.

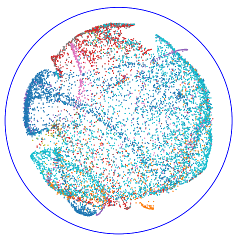
5.3.2. Hypothesis

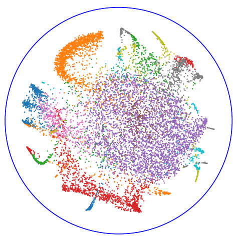
We expect that using the Gaussian leads to embeddings that are not pushed out towards the boundary. This allows us to visually interpret them better.

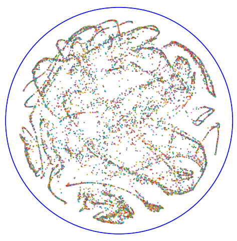
5.3.3. Results

Figure 5.3 contains a series of embeddings of the aforementioned datasets using Hyperbolic SNE (using the Gaussian distribution). Two different embeddings per datasets, corresponding to two different amounts of optimization iterations are depicted.

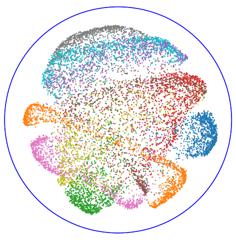

(a) MNIST, 3414 iterations

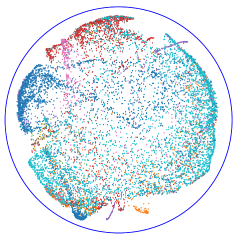

(b) C. Elegans, 3414 iterations

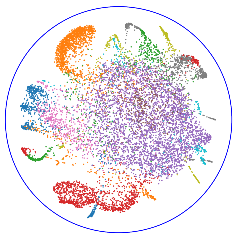

(c) Planaria, 4248 iterations

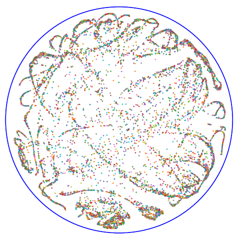

(d) Wordnet, 3414 iterations

Gaussian gradient, intermediate embeddings


(e) MNIST, 5909 iterations


(f) C. Elegans, 5909 iterations


(g) Planaria, 9580 iterations


(h) Wordnet, 5909 iterations

Gaussian gradient, final embeddings

Figure 5.3: Hyperbolic SNE embeddings

5.3.4. Discussion

Each dataset that is embedded using the Gaussian distribution shows a much more visually interpretable result than the visualizations in the previous experiment 5.2 .

Points in the embeddings are no longer pushed towards the boundaries and structures in the embeddings emerge and remain in the final embeddings (at convergence). This is a direct consequence of preventing the embeddings from taking up too much (Hyperbolic) space, i.e. preventing embeddings from being pushed towards the boundary.

Note that the final Planaria and Wordnet Hyperbolic SNE (figure 5.3) embeddings contain structures similar to the corresponding intermediate embeddings of Hyperbolic t-SNE (see figure 5.2, second row). However unlike in Hyperbolic t-SNE where these structures were lost after the embeddings converge, the corresponding Hyperbolic SNE embeddings retain these structures after convergence. This indicates that Hyperbolic SNE is able to capture the same structures that Hyperbolic t-SNE is also capable of capturing (but incapable of retaining) which adds credence to the utility of Hyperbolic SNE over Hyperbolic t-SNE (at least in terms of visualizing Hyperbolic embeddings). The only difference is that the Hyperbolic SNE visualizations retain these structures after convergence.

Finally, we can observe that it is not necessary to wait for the optimization procedure to fully converge. Much of the structure present in the final embeddings is also present in the intermediate embeddings. The final embedding appears to be a slightly expanded version of the intermediate embeddings.

5.4. Embedding Tree-like Data

Section 4.4 discussed two ways to assess the quality of embeddings. In this section we focus on the qualitative form of assessment by embedding artificial tree-like datasets (described in section 4.4.1) using Hyperbolic t-SNE with the correct gradient, and Hyperbolic SNE (the variant with the Gaussian distribution).

5.4.1. Experimental Setup

We first generate tree-like datasets using the code in appendix C. These trees are then embedded in the Poincaré Disk using the correct gradient version of Hyperbolic t-SNE (with the t-distribution) and Hyperbolic SNE (with a Gaussian distribution). In addition to Hyperbolic embeddings, we also embed the tree-like datasets using standard Euclidean t-SNE [16]. This is to highlight how standard t-SNE struggles with capturing tree-like structures compared to Hyperbolic embeddings.

Trees of various sizes are generated by varying the number of children per node and the depth of the trees generated. The total number of nodes a tree contains can be computed via the formula $n = \sum_i^d c^i$ where d is the maximum depth of the tree and c is the number of children per node.

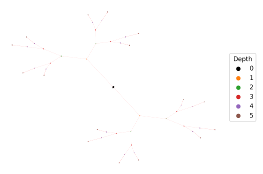
5.4.2. Hypothesis

Similarly to the results in section 4.2, we expect Hyperbolic t-SNE to push embedding points towards the boundary of the disk. For smaller trees hierarchical structure may still be visible, however embeddings of larger trees are expected to appear distorted . We thus expect Hyperbolic SNE to produce better embeddings for larger trees whereas smaller trees are expected to be embedded with similar quality.

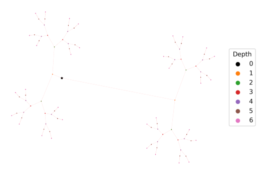
5.4.3. Results

In figure 5.4 we find visualizations produced by embedding various (generated) tree-like datasets using (standard) t-SNE, Hyperbolic t-SNE, and Hyperbolic SNE. Each kind of embedding is produced in two variations; the standard visualization that is produced as well as a visualization where edges are drawn between neighbouring nodes in the embedding. The visualizations have the nodes coloured based on the depth of a node (which can be read off from the legend). The black star represents the root node of the tree. The edges aim to showcase to what extent a tree-like structure is preserved. Below we showcase a mix of both as for smaller trees the images containing edges are more informative. The other versions can be found in the appendix D.

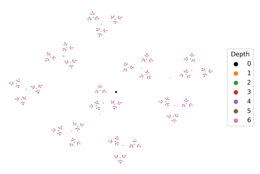
Note that these edges do not represent the actual geodesic between neighbouring points. The edges only aim to highlight which nodes are directly related to each other, revealing the parent-child structure of the tree.



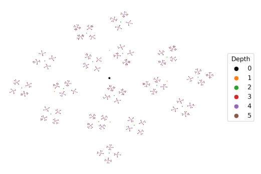
(a) 2 children per node, depth of 5. 192 total nodes



(b) 2 children per node, depth of 6. 448 total nodes

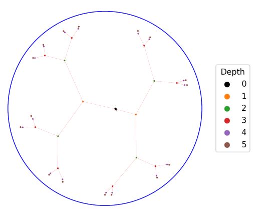


(c) 3 children per node, depth of 6. 5103 total nodes

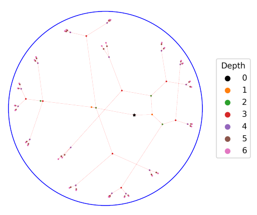


(d) 4 children per node, depth of 5. 6144 total nodes

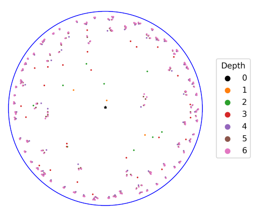
Standard t-SNE tree embeddings



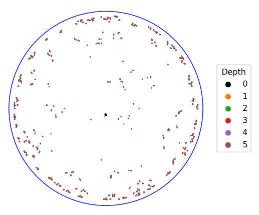
(e) 2 children per node, depth of 5. 192 total nodes



(f) 2 children per node, depth of 6. 448 total nodes

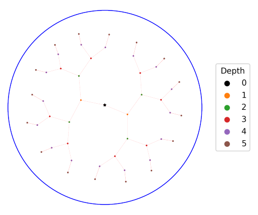


(g) 3 children per node, depth of 6. 5103 total nodes

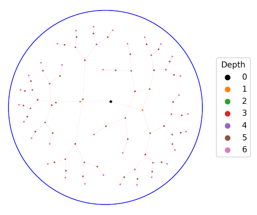


(h) 4 children per node, depth of 5. 6144 total nodes

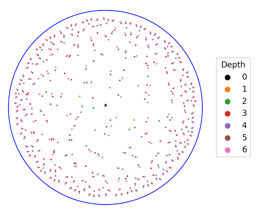
Hyperbolic t-SNE (t-distribution) tree embeddings



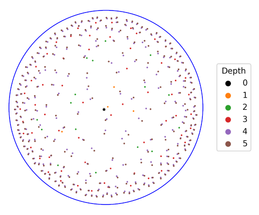
(i) 2 children per node, depth of 5. 192 total nodes



(j) 2 children per node, depth of 6. 448 total nodes



(k) 3 children per node, depth of 6. 5103 total nodes



(l) 4 children per node, depth of 5. 6144 total nodes

Hyperbolic SNE (Gaussian distribution) tree embeddings

Figure 5.4: Tree embeddings

5.4.4. Discussion

In the case of standard t-SNE, a tree structure can still be somewhat inferred from these datasets. One can imagine picking up the root node in 3-dimensions and the result taking the shape of a 3-dimensional tree where the depth of the tree lives in the third dimension. However this is purely an imaginary way to assess for tree-like structures and without the edges and labels there is no clear way of interpreting the resulting embedding as originating from a tree-like dataset. Furthermore, as can be seen from the largest dataset, embeddings of large trees quickly lose their tree-like structure. One could still infer that the embedding is that of a tree, however as the tree grows in size this becomes more and more difficult as points start to get cluttered heavily. Furthermore, without the edges the tree-like relationship between nodes would be even more unclear. In practical settings data is rarely a perfect tree. This means that even in the ideal scenario the t-SNE embedding struggles to capture tree-like structure.

The Hyperbolic embeddings fare better at capturing the tree-like structure we are looking for. Recall from section 3.2.1 that points embedded at a radius r from the origin represent nodes at a similar depth level. Since the number of nodes in a layer of the tree grows exponentially with depth, we thus expect to see an increase in density of points as points spread out towards the border. This is because space closer to the border corresponds to greater depths of the tree. At a greater depth, more nodes reside. This expectation is matched by the Hyperbolic embedding results and especially visible in the larger trees. For the smaller trees we have drawn edges between neighbouring points (points that are neighbours in the original tree-like dataset) to highlight these structures. Due to a small number of points, the density increase and hierarchical structure is not very visible. From the edges we can observe how the embedding represents a tree that effectively grows outwards towards the boundary from the origin. Furthermore, based on the color of the points, we can notice that the points belonging to a greater depth are generally placed closer to the edge of the Poincaré Disk. This is visible across all the trees embedded. We thus conclude that the Hyperbolic embeddings are capable of capturing the hierarchical relationships in the dataset.

However a perfect tree is not recreated. We are after all using gradient descent to find embeddings and therefore likely have to settle for a solution that is not globally optimal.

Finally, we note the differences between the Hyperbolic embeddings produced by Hyperbolic SNE and Hyperbolic t-SNE. Hyperbolic t-SNE quickly pushes the embeddings towards the boundary whereas Hyperbolic SNE embeddings capture the density increase more gradually. For smaller trees both methods are capable of capturing tree-like structure, however Hyperbolic SNE utilizes the available (visualizable) space more efficiently which benefits larger trees (larger datasets) as the increase in density is shown more explicitly visible.

5.5. Measuring (exact) neighbourhod preservation

Finally, as discussed in section 4.4.2 we compute the Precision-Recall metric over the various datasets and different embedding methods discussed so far. In this section we present this metric in the form of PR-curves and compare and analyze the results.

5.5.1. Experimental Setup

We compute the Precision-Recall values for the (non-artificial) datasets used in experiments 1 to 3 using the incorrect and correct gradient for the t-distribution version of Hyperbolic embeddings, as well as the correct gaussian gradient version.

Parameter settings

Before computing the PR metrics, several parameters must be set. These parameters are k_{max}, the maximum neighbourhood size we consider per point, and a value n that indicates over how many samples we wish to calculate the metric over. The final result is then the mean of the sum of PR-values per sample. In this experiment we compute the PR-metric over all datapoints in our embedding. This means $n = 10000$ as this is the size of the data used in the experiments.

Finally we set k_{max} to 30 to reflect the fact that the perplexity value, a measure of the effective number of neighbours, was chosen to be 30 when computing the high-dimensional similarities (see experiment 5.1). In addition to this, previous work applying the PR-metric to Hyperbolic embeddings [33] and non-Hyperbolic neighbour embedding [27] use a value of $k_{max} = 30$.

5.5.2. Hypothesis

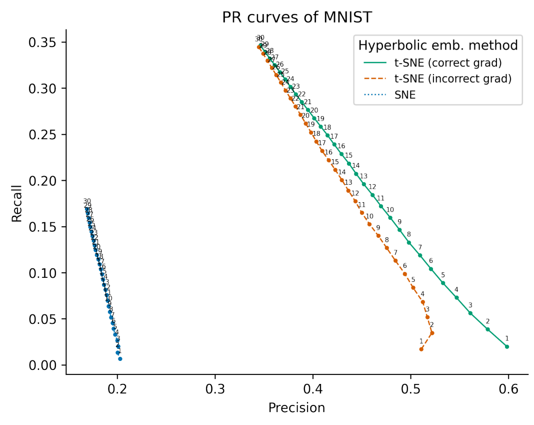
Between the incorrect and correct Hyperbolic t-SNE gradient versions we expect the metric to be more optimal for the correct gradient. This is because one formulation of the gradient corresponds correctly to the cost function being minimized thus leading to more optimal embeddings as discussed in section 5.1.

Hyperbolic SNE (Gaussian distribution version) utilizes less space for embedding points by design (recall from section 4.2 that we are essentially forcing the embedding to only occupy the visually relevant regions of the disk). As a result the embeddings are much more clustered which may distort the exact neighbourhoods (which is measured by the PR metric). Therefore the PR metric values for this method may be worse compared to the Hyperbolic t-SNE version.

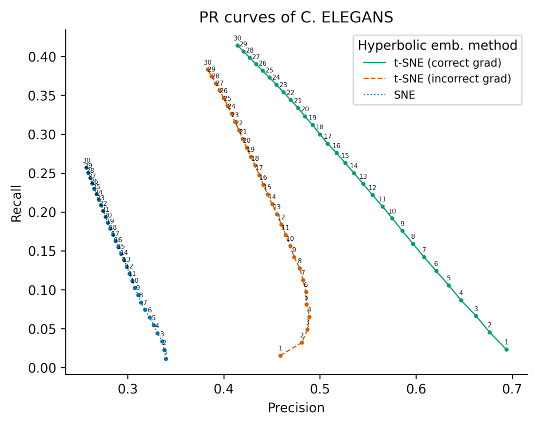
5.5.3. Results

Figure 5.5 contains PR-metric graphs of the embeddings produced by the above experiments. The green/continuous curve corresponds to Hyperbolic t-SNE embeddings using the correct gradient, the orange/striped curve corresponds to Hyperbolic t-SNE embeddings using the incorrect gradient, and the blue/dotted curve corresponds to Hyperbolic SNE embeddings.

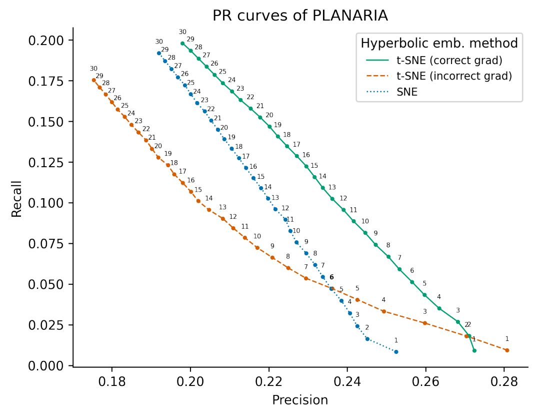
The dots/markings along the curve represent the k value for which the PR-metric is computed against. These markings are also numbered to reflect which k value they correspond to. The curve itself is an interpolation between these points.



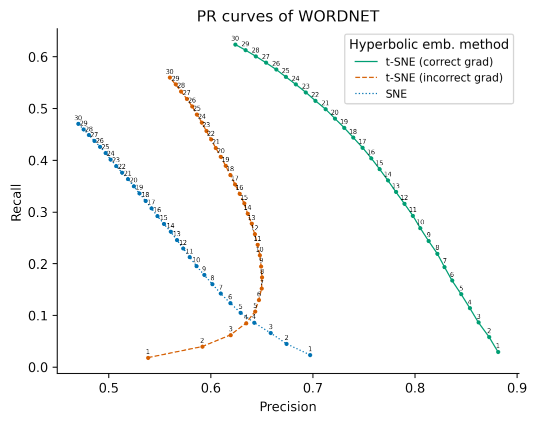
(a) MNIST dataset



(b) C. Elegans dataset



(c) Planaria dataset



(d) Wordnet dataset

Figure 5.5: PR curves of real world datasets used. The numberings indicate the current value of k up to $k_{max} = 30$

5.5.4. Discussion

From the PR-curves we can see that the correct Hyperbolic t-SNE gradient version improves on the incorrect Hyperbolic t-SNE gradient version which matches our hypothesis. However, the PR-values between the correct and incorrect Hyperbolic t-SNE gradient appear to not differ by too much for larger values of k. This is likely due to how similar both gradients are in their formulation. We can see from the visualizations in experiment 5.2 that the two gradients do not differ too much in how the resulting embeddings appear visually. The incorrect gradients' embeddings can be interpreted as a more dense version of the correct gradient embeddings. Similar neighbourhoods appear to be preserved for both the incorrect and correct gradient embeddings.

The Gaussian gradient however achieves the poorest performance (except for the Planaria dataset). Below we will attempt at an explanation for this phenomena.

Gaussian gradient PR-metric performance

The Gaussian distribution force embeddings to occupy a relatively small amount of space, which may distort exact neighbourhood reconstruction for points (meaning how exactly the high-dimensional neighbourhoods of points are reconstructed after embedding those points in the lower-dimensional space). Especially points that in the high-dimensional case are neighbours (i.e. constitute a cluster/neighbourhood), are embedded in such a way that their distances between each other in the Poincaré Disk is small. This is a direct consequence of using the Gaussian distribution when converting distances to probabilities/similarities (see section 4.3).
As a result, there is less space (atleast compared to Hyperbolic t-SNE) for points to maneuver around in

which may affect the exact neighbourhood reconstructions. Since the points are more densely packed, small disturbances in a points positioning (due to the optimization procedure) may change the (exact) neighbourhood of a point as many other points (due to the high density) exist in the surrounding area. For example, if some point is slightly moved in some direction it may suddenly find itself with a whole new set of points that it would call its neighbourhood.
Another reason may have to do with how distances are mapped to probabilities in the Gaussian case. Since embeddings are positioned in such a way that the similarity matrices coincide as well as possible, the Gaussian may perform worse than the t-distribution. This is because the Gaussian maps distances to 0 faster (due to its smaller tails). Any point that is mapped to 0 probability/similarity is treated the same. The t-distribution thus enforces more constraints on the positionings of points because more points have nonzero similarity. As a result, neighbourhoods are better preserved because of these additional constraints (nonzero similarities). The Gaussian on the other hand may have mapped points to 0 similarity that the t-distribution variant would not have mapped to 0. This means those points have more freedom in where they can be placed which affects the exact reproduction of neighbourhoods.

In summary, neighbourhoods in Hyperbolic SNE are very sensitive to small changes in points positioning, something less present in Hyperbolic t-SNE. In Hyperbolic t-SNE much greater distances between points are attained leaving more space for points to move around in without affecting their exact neighbourhoods. In addition, the way these distributions map distances to similarities affect the resulting positioning of points.

From the PR-curves we can also observe that the Gaussian gradient performs differently for different datasets. It could be that the nature of a dataset, for example whether the dataset is strongly hierarchical, affects this quantitative performance of the Gaussian gradient. For example, the discrepancy in metric performance between the Hyperbolic SNE (Gaussian distribution) and the Hyperbolic t-SNE versions (t-distribution) for the more hierarchical datasets (C. Elegans, Planaria, Wordnet) is smaller as compared to a non-hierarchical dataset (MNIST). In the Planaria graph, Hypernolic SNE's PR-metric performance seem to approach the correct gradient version of Hyperbolic t-SNE and in Wordnet the differences are also much smaller. However, further research is required to understand the exact relationship between the PR-metric performance of Hyperbolic SNE, Hyperbolic t-SNE, and the dataset used.

We thus are faced with a limitation of the PR-metric. Since the PR-metric is a measure of exact neighbourhood reproductions, it is very sensitive to the positioning of embeddings. However (t-)SNE uses probabilistic similarities instead of a distance based one which results in more freedom in positionings as potentially many different distances can be mapped to the same or nearly the same probability. This means all such distances are treated the same whereas in a purely distance-based setting this would not happen. Thus the PR-metric is not an ideal quality measure for probabilistic similarities.

Finally, different choices for computing high-dimensional similarities or in our case, constructing a k-nearest neighbour graph of the high dimensional data for other k's than $k = 30$ (thus also choosing different values for k_{max}) may tell a different story. This is because larger and larger neighbourhoods are then considered which may mitigate the distortion of neighbourhoods due to clustered embeddings. Further experiments need to be performed to investigate the full scope of consequences relating to the Gaussian gradient.

6 Conclusion

6.0.1. Summary of contributions

In this thesis we have highlighted a mistake made in the gradient formulation of previous t-SNE adaptations to Hyperbolic space. We have delved into the origin of the mistake and have provided a derivation of a corrected version of the gradient. This lead us to realize that embedding data in the Poincaré Disk using a t-distribution resulted in embeddings that are pushed out towards the boundary of the disk, causing the visualizations to be difficult to interpret. We then explain how the fatter tails of the t-distribution gives rise to this phenomena, and provide a means of tackling this issue by using a distribution (Gaussian distribution) with smaller tails. This method, named Hyperbolic SNE, allowed us to obtain visually interpretable results in the Poincaré Disk.
Finally, we assessed the quality of the corrected version of Hyperbolic t-SNE and our proposed method of Hyperbolic SNE. We have shown that both are capable of revealing tree-like structure in their respective visualizations with Hyperbolic SNE being more visually informative for larger trees. We have also shown that by the PR-metric, the corrected version of Hyperbolic t-SNE outperforms the incorrect gradient version of Hyperbolic t-SNE. Hyperbolic SNE (Gaussian distribution), attains less optimal PR-metric values which has to do with the nature of the Gaussian distribution.

6.0.2. Main takeaway

The main takeaway is that effective visualizations in the Poincaré Disk must utilize the space well by embedding into the visually relevant region(s) of the disk. If the visualization embeds points towards the boundary of the disk, then the results are no longer visually informative. We therefore recommend using Hyperbolic SNE (with the Gaussian distribution) for this purpose as it is able to use the space more effectively than Hyperbolic t-SNE (with the t-distribution)

6.0.3. Future work

In future work the consequences of using the Gaussian distribution (and possibly other distributions) should be further explored. As of now we only have qualitative evidence that the Gaussian produces better results. Our quantitative measure of choice for assessing embedding quality does not seem to agree with the qualitative evidence for the use of the Gaussian. Thus far we have speculated as to why Hyperbolic t-SNE performs worse according to the PR-metric. This speculation can be further investigated by using metrics that are less sensitive to exact neighbourhood reproductions. Metrics that are more tailored towards probabilistic similarity measures. Different metrics may provide additional insights as no one metric is universally and objectively superior. Alternatively, one could further investigate the embeddings of Hyperbolic SNE qualitatively using domain knowledge to confirm that the revealed structures are in fact informative.

Other directions of research may investigate adjustments to the embedding procedure itself. Firstly, changes to how high-dimensional similarities are computed can be made. In our current implementation a k-nearest neighbour method is used. However such a construction may not capture the high-dimensional data manifold well as it is sensitive to the choice of k. Different kinds of methods may results in different kinds of embeddings which is something unexplored.

Another interesting approach would be to vary the initial embedding. Currently points are initialized randomly near the origin of the disk. However other forms of initialization can be explored and may yield different results.

Finally, in our works we did not test different hyperparameter values extensively. Since Hyperbolic SNE (using a Gaussian gradient) has a variance parameter σ^2 to tune, experiments that vary this parameter is another direction of investigation.

References

[1] Aaron B. Adcock, Blair D. Sullivan, and Michael W. Mahoney. “Tree-Like Structure in Large Social and Information Networks”. In: *2013 IEEE 13th International Conference on Data Mining*. 2013, pp. 1–10. DOI: 10.1109/ICDM.2013.77.

[2] Andreas Bloch. *https://andbloch.github.io/Stochastic-Gradient-Descent-on-Riemannian-Manifolds/*. 2019.

[3] Silvere Bonnabel. “Stochastic Gradient Descent on Riemannian Manifolds”. In: *IEEE Transactions on Automatic Control* 58.9 (Sept. 2013), pp. 2217–2229. ISSN: 1558-2523. DOI: 10.1109/tac.2013.2254619. URL: http://dx.doi.org/10.1109/TAC.2013.2254619.

[4] Andrej Cvetkovski and Mark Crovella. *Multidimensional Scaling in the Poincare Disk*. 2016. arXiv: 1105.5332 [stat.ML]. URL: https://arxiv.org/abs/1105.5332.

[5] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. *Hyperbolic Neural Networks*. 2018. arXiv: 1805.09112 [cs.LG]. URL: https://arxiv.org/abs/1805.09112.

[6] Pavel Grinfeld. *Introduction to Tensor Analysis and the Calculus of Moving Surfaces*. 2013.

[7] Yunhui Guo, Haoran Guo, and Stella Yu. *CO-SNE: Dimensionality Reduction and Visualization for Hyperbolic Data*. 2022. arXiv: 2111.15037 [cs.LG]. URL: https://arxiv.org/abs/2111.15037.

[8] D. Hilbert. “Ueber Flächen von constanter Gaussscher Krümmung”. In: *Transactions of the American Mathematical Society 2 (1901)* 1 (1901). https://doi.org/10.2307/1986308, pp. 87–99.

[9] Geoffrey E Hinton and Sam Roweis. “Stochastic Neighbor Embedding”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Becker, S. Thrun, and K. Obermayer. Vol. 15. MIT Press, 2002. URL: https://proceedings.neurips.cc/paper_files/paper/2002/file/6150ccc6069bea6b5716254057a194ef-Paper.pdf.

[10] H Hotelling. “Analysis of a complex of statistical variables into principal components.” In: *The Journal of Educational Psychology* (1933).

[11] Brian Keng. *https://bjlkeng.io/posts/hyperbolic-geometry-and-poincare-embeddings/*. 2018.

[12] Klimovskaia and D. Lopez-Paz andL. Bottou andM. Nickel. “Poincaré maps for analyzing complex hierarchies in single-cell data”. In: *Nature Communications* (2020). DOI: 10.1038/s41467-020-16822-4.

[13] Berens P. Kobak D. “The art of using t-SNE for single-cell transcriptomics”. In: *Nat Commun.* (2019). DOI: doi:10.1038/s41467-019-13056-x..

[14] Dmitri Krioukov et al. “Hyperbolic geometry of complex networks”. In: *Phys. Rev. E* 82 (3 2010), p. 036106.

[15] J.B. Kruskal. “Multidimensional Scaling by optimizing goodness of fit to a nonmetric hypothesis”. In: *PSYCHOMETRIKA* 9 NO. 1 (1964).

[16] Geoffrey Hinton Laurens van der Maaten. “Visualizing data using t-SNE”. In: *Journal of machine learning research* 9 (2008). URL: https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf.

[17] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[18] James R. Lee, Assaf Naor, and Yuval Peres. *Trees and Markov convexity*. 2007. arXiv: 0706.0545 [math.MG]. URL: https://arxiv.org/abs/0706.0545.

[19] John A Lee and Michel Verleysen. *Nonlinear Dimensionality Reduction*. 2007.

[20] Plass M et al. “N. Cell type atlas and lineage tree of a whole complex animal by single-cell transcriptomics.” In: *Science* (2018). DOI: 10.1126/science.aaq1723. URL: https://shiny.mdc-berlin.de/psca/.

[21] F. Papadopoulos M. Boguñá and D. Krioukov. “Sustaining the internet with hyperbolic mapping”. In: *Nature Communications* (2010). DOI: 10.1038/ncomms1063.

[22] Laurens van der Maaten. *Barnes-Hut-SNE*. 2013. arXiv: 1301.3342 [cs.LG]. URL: https://arxiv.org/abs/1301.3342.

[23] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2020. arXiv: 1802.03426 [stat.ML]. URL: https://arxiv.org/abs/1802.03426.

[24] George A. Miller. “WordNet: A Lexical Database for English”. In: *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*. 1992. URL: https://aclanthology.org/H92-1116/.

[25] Maximilian Nickel and Douwe Kiela. *Poincaré Embeddings for Learning Hierarchical Representations*. 2017. arXiv: 1705.08039 [cs.AI]. URL: https://arxiv.org/abs/1705.08039.

[26] J Ontrup and Helge Ritter. “Hyperbolic Self-Organizing Maps for Semantic Navigation”. In: *Adv Neural Inf Process Syst* 14 (Apr. 2002).

[27] Nicola Pezzotti et al. “Hierarchical Stochastic Neighbor Embedding”. In: *Computer Graphics Forum (Proc. of EuroVis)* 35.3 (June 2016), pp. 21–30. URL: http://graphics.tudelft.nl/Publications-new/2016/PHLEV16.

[28] Erzsébet Ravasz and Albert-László Barabási. “Hierarchical organization in complex networks”. In: *Phys. Rev. E* 67 (2 Feb. 2003), p. 026112. DOI: 10.1103/PhysRevE.67.026112. URL: https://link.aps.org/doi/10.1103/PhysRevE.67.026112.

[29] Carlos P. Roca1 et al. *A Cross Entropy test allows quantitative statistical comparison of t-SNE and UMAP representations*. 2021. arXiv: 2112.04172 [q-bio.QM]. URL: https://arxiv.org/abs/2112.04172.

[30] Christopher De Sa et al. *Representation Tradeoffs for Hyperbolic Embeddings*. 2018. arXiv: 1804.03329 [cs.LG]. URL: https://arxiv.org/abs/1804.03329.

[31] R. Sarkar. “Low distortion delaunay embedding of trees in hyperbolic plane”. In: *Int. Symp. on Graph Drawing* (2011).

[32] Badrul Sarwar et al. “Application of dimensionality reduction in recommender system-a case study”. In: *ACM WebKDD workshop*. Vol. 1625. 1. Citeseer. 2000, pp. 285–295.

[33] Martin Skrodzki et al. *Accelerating hyperbolic t-SNE*. 2024. arXiv: 2401.13708 [cs.HC]. URL: https://arxiv.org/abs/2401.13708.

[34] Zhou Y andSharpee TO. “Hyperbolic geometry of gene expression”. In: (2021). DOI: 10.1016/j.isci.2021.102225.

[35] Antony Unwin. “Why Is Data Visualization Important? What Is Important in Data Visualization?” In: *Harvard Data Science Review* 2.1 (Jan. 2020). https://hdsr.mitpress.mit.edu/pub/zok97i7p.

[36] Looz M. v., Meyerhenke H., and Prutkin R. “Generating random hyperbolic graphs in subquadratic time”. In: *International Symposium on Algorithms and Computation* (2015).

[37] Eduardo da Veiga Beltrame. *Packer et al 2019 scRNAseq dataset wrangled into standard WormBase anndata- 89k cells profiled with 10xv2 across multiple timepoints of development*. Apr. 2021. DOI: 10.22002/D1.1945.

[38] Kevin Verbeek and Subhash Suri. “Metric embedding, hyperbolic space, and social networks”. In: *Computational Geometry* 59 (2016), pp. 1–12. ISSN: 0925-7721. DOI: https://doi.org/10.1016/j.comgeo.2016.08.003. URL: https://www.sciencedirect.com/science/article/pii/S0925772116300712.

[39] Junpeng Wang et al. “Visual Analytics for RNN-Based Deep Reinforcement Learning”. In: *IEEE Transactions on Visualization and Computer Graphics* 28.12 (2022), pp. 4141–4155. DOI: 10.1109/TVCG.2021.3076749.

[40] Wikipedia. *Arc length - https://en.wikipedia.org/wiki/Arc_length*.

[41] Wikipedia. *Complex network - https://en.wikipedia.org/wiki/Complex_network*.

[42] Wikipedia. *First Fundamental Form - https://en.wikipedia.org/wiki/First_fundamental_form*.

[43] Wikipedia. *Gaussian Curvature - https://en.wikipedia.org/wiki/Gaussian_curvature*.

[44] Wikipedia. *Geodesic - https://en.wikipedia.org/wiki/Geodesic*.

[45] Wikipedia. *Gradient Descent - https://en.wikipedia.org/wiki/Gradient_descent*.

[46] Wikipedia. *Hyperboloid Model of Hyperbolic Space - https://en.wikipedia.org/wiki/Hyperboloid_model*.

[47] Wikipedia. *Latent space - https://en.wikipedia.org/wiki/Latent_space*.

[48] Wikipedia. *Manifold hypothesis - https://en.wikipedia.org/wiki/Manifold_hypothesis*.

[49] Wikipedia. *Nonlinear dimensionality reduction - https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction*.

[50] Jiang Wu et al. “TacticFlow: Visual Analytics of Ever-Changing Tactics in Racket Sports”. In: *IEEE Transactions on Visualization and Computer Graphics* 28.1 (2022), pp. 835–845. DOI: 10.1109/TVCG.2021.3114832.

[51] Jiazhi Xia et al. “Revisiting Dimensionality Reduction Techniques for Visual Cluster Analysis: An Empirical Study”. In: *IEEE Transactions on Visualization and Computer Graphics* 28.1 (2022), pp. 529–539. DOI: 10.1109/TVCG.2021.3114694.

[52] Yuansheng Zhou, Brian Smith, and Tatyana Sharpee. “Hyperbolic geometry of the olfactory space”. In: *Science Advances* 4 (Aug. 2018), eaaq1458. DOI: 10.1126/sciadv.aaq1458.

A

Correct Gradient derivation

In the main text we have used the term "gradient" interchangeably for both the Euclidean gradient ($\frac{\partial \mathcal{L}}{\partial \mathbf{y}}$, i.e. in the context of Riemannian Manifolds this is the tangent vector belonging to the tangent space), as well as the Riemannian gradient $g_{\mathcal{D}}^{-1} \frac{\partial \mathcal{L}}{\partial \mathbf{y}}$. To remove any confusion, we shall now refer to $\frac{\partial \mathcal{L}}{\partial \mathbf{y}}$ as the variation of the loss or cost function (as in, how the cost function C varies if its inputs $\mathbf{y}_i$ varies). This will now be denoted as $\frac{\delta C}{\delta \mathbf{y}_i}$.

The derivation of the variations of the cost function follows along the same lines as those for the Euclidean t-SNE gradient [16].

We first define some helper notation with $d_{ij}^{\mathcal{H}} = d^{\mathcal{H}}(\mathbf{y}_i, \mathbf{y}_j)$, the Hyperbolic distance (see equation:equation (3.13)). Secondly we define $r_{ij}^{\mathcal{H}} = {d_{ij}^{\mathcal{H}}}^2$ to be the squared Hyperbolic distance. Finally $Z^{\mathcal{H}} = \sum_{k \neq \ell}(1 + {d_{ij}^{\mathcal{H}}}^2)^{-1}$ represents the normalization factor (see equation: 3.15).

Now, we proceed to derive the variation of the cost function of Hyperbolic t-SNE (see section: 3.16):

$$\begin{aligned} C^{\mathcal{H}} = KL(P||Q) &= \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}^{\mathcal{H}}} \\ &= \sum_i \sum_j p_{ij} \log(p_{ij}) - p_{ij} \log(q_{ij}^{\mathcal{H}}), \end{aligned}$$

with low-dimensional probabilities $q_{ij}^{\mathcal{H}}$ according to equation (3.15).

As changing $\mathbf{y}_i$ only impacts $d_{ij}^{\mathcal{H}}$ and $d_{ji}^{\mathcal{H}}$ we can therefore by the chain rule and by using that $r_{ij}^{\mathcal{H}} = r_{ji}^{\mathcal{H}}$ (since the squared distance is symmetric) obtain:

$$\begin{aligned} \frac{\delta C^{\mathcal{H}}}{\delta \mathbf{y}_i} &= \sum_j \left(\frac{\delta C^{\mathcal{H}}}{\delta r_{ij}^{\mathcal{H}}} \frac{\delta r_{ij}^{\mathcal{H}}}{\delta \mathbf{y}_i} + \frac{\delta C^{\mathcal{H}}}{\delta r_{ji}^{\mathcal{H}}} \frac{\delta r_{ji}^{\mathcal{H}}}{\delta \mathbf{y}_i} \right) \\ &= 2 \sum_j \frac{\delta C^{\mathcal{H}}}{\delta r_{ij}^{\mathcal{H}}} \frac{\delta r_{ij}^{\mathcal{H}}}{\delta \mathbf{y}_i}, \end{aligned} \tag{A.1}$$

where the variation according to $r_{ij}^{\mathcal{H}}$ is given as:

$$\begin{aligned} \frac{\delta C^{\mathcal{H}}}{\delta r_{ij}^{\mathcal{H}}} &= -\sum_{k \neq \ell} p_{k\ell} \frac{\delta(\log q_{k\ell}^{\mathcal{H}})}{\delta r_{ij}^{\mathcal{H}}} = -\sum_{k \neq \ell} p_{k\ell} \frac{\delta(\log q_{k\ell}^{\mathcal{H}} Z^{\mathcal{H}} - \log Z^{\mathcal{H}})}{\delta r_{ij}^{\mathcal{H}}} \\ &= -\sum_{k \neq l} p_{k\ell} \left(\frac{1}{q_{k\ell}^{\mathcal{H}} Z^{\mathcal{H}}} \frac{\delta((1 + r_{k\ell}^{\mathcal{H}})^{-1})}{\delta r_{ij}^{\mathcal{H}}} - \frac{1}{Z^{\mathcal{H}}} \frac{\delta Z^{\mathcal{H}}}{\delta r_{ij}^{\mathcal{H}}} \right). \end{aligned}$$

In the above derivation we use $\log$ properties and $Z^{\mathcal{H}}$ to rewrite $\log q_{k\ell}^{\mathcal{H}}$ as $\log q_{k\ell}^{\mathcal{H}} Z^{\mathcal{H}} - \log Z^{\mathcal{H}}$. t-SNE [16] employs a similar trick and it allows us to obtain a convenient form for the gradient.

The variation $\frac{\delta((1+r^{\mathcal{H}}_{k\ell})^{-1})}{\delta r^{\mathcal{H}}_{ij}}$ is only nonzero when $k=i$ and $\ell=j$, thus:

$$\frac{\delta C^{\mathcal{H}}}{\delta r^{\mathcal{H}}_{ij}} = \frac{p_{ij}}{q^{\mathcal{H}}_{ij} Z^{\mathcal{H}}}(1+r^{\mathcal{H}}_{ij})^{-2} - \sum_{k\neq\ell} p_{k\ell}\frac{(1+r^{\mathcal{H}}_{ij})^{-2}}{Z^{\mathcal{H}}} \tag{A.2}$$

Since $\sum_{k\neq\ell} p_{kl} = 1$ the equation simplifies to:

$$\frac{\delta C^{\mathcal{H}}}{\delta r^{\mathcal{H}}_{ij}} = p_{ij}(1+r^{\mathcal{H}}_{ij})^{-1} - q^{\mathcal{H}}_{ij}(1+r^{\mathcal{H}}_{ij})^{-1} = (p_{ij} - q^{\mathcal{H}}_{ij})(1+r^{\mathcal{H}}_{ij})^{-1} \tag{A.3}$$

Substituting into equation (A.1) we get:

$$\frac{\delta C^{\mathcal{H}}}{\delta \mathbf{y}_i} = 2\sum_j (p_{ij} - q^{\mathcal{H}}_{ij})(1+r^{\mathcal{H}}_{ij})^{-1}\frac{\delta r^{\mathcal{H}}_{ij}}{\delta \mathbf{y}_i}$$

Finally, recall that $r^{\mathcal{H}}_{ij} = {d^{\mathcal{H}}_{ij}}^2$. Performing this substitution leads us to the final form of the variation of the cost function:

$$\begin{aligned}\frac{\delta C^{\mathcal{H}}}{\delta \mathbf{y}_i} &= 2\sum_j (p_{ij} - q^{\mathcal{H}}_{ij})(1+{d^{\mathcal{H}}_{ij}}^2)^{-1}\frac{\delta {d^{\mathcal{H}}_{ij}}^2}{\delta \mathbf{y}_i} \\ &= 4\sum_j (p_{ij} - q^{\mathcal{H}}_{ij})(1+{d^{\mathcal{H}}_{ij}}^2)^{-1} d^{\mathcal{H}}_{ij}\frac{\delta d^{\mathcal{H}}_{ij}}{\delta \mathbf{y}_i}.\end{aligned}$$

We can observe an extra factor $d^{\mathcal{H}}_{ij}$ which is not present in the incorrect derivations of the gradient (see section: 4.1).

Additionally, $\frac{\delta d^{\mathcal{H}}_{ij}}{\delta \mathbf{y}_i}$ can be derived by taking the derivative of 3.13:

$$\frac{\delta d^{\mathcal{H}}_{ij}}{\delta \mathbf{y}_i} = \frac{4((||\mathbf{y}_j||^2 - 2\langle \mathbf{y}_i, \mathbf{y}_j\rangle + 1)\mathbf{y}_i/\alpha - \mathbf{y}_j)}{\alpha\beta\sqrt{\gamma^2-1}} \tag{A.4}$$

Where $\alpha = 1 - ||\mathbf{y}_u||^2$, $\beta = 1 - ||\mathbf{y}_j||^2$, $\gamma = 1 + \frac{2}{\alpha\beta}||\mathbf{y}_i - \mathbf{y}_j||^2$. For more details please see [5].

B

Gaussian Gradient derivation

We begin from the same starting point as the derivation for the Hyperbolic t-SNE gradient (see appendix: A).

$$\begin{aligned}\frac{\delta C^{\mathcal{H}}}{\delta \mathbf{y}_i} &= \sum_j \left(\frac{\delta C^{\mathcal{H}}}{\delta r^{\mathcal{H}}_{ij}} \frac{\delta r^{\mathcal{H}}_{ij}}{\delta \mathbf{y}_i} + \frac{\delta C^{\mathcal{H}}}{\delta r^{\mathcal{H}}_{ji}} \frac{\delta r^{\mathcal{H}}_{ji}}{\delta \mathbf{y}_i} \right) \\ &= 2 \sum_j \frac{\delta C^{\mathcal{H}}}{\delta r^{\mathcal{H}}_{ij}} \frac{\delta r^{\mathcal{H}}_{ij}}{\delta \mathbf{y}_i},\end{aligned} \tag{B.1}$$

Similarly to A.1 we focus on the $\frac{\delta C^{\mathcal{H}}}{\delta r^{\mathcal{H}}_{ij}}$ term first:

$$\frac{\delta C^{\mathcal{H}}}{\delta r^{\mathcal{H}}_{ij}} = -\sum_{k \neq \ell} p_{k\ell} \frac{\delta (\log q^{\mathcal{H}}_{k\ell})}{\delta r^{\mathcal{H}}_{ij}} = -\sum_{k \neq \ell} p_{k\ell} \frac{\delta (\log q^{\mathcal{H}}_{k\ell} Z^{\mathcal{H}} - \log Z^{\mathcal{H}})}{\delta r^{\mathcal{H}}_{ij}}$$

Note that we are now using the Gaussian distribution instead of the t-distribution (see section: 4.6). This means that $q^{\mathcal{H}}_{k\ell} = (-{d^{H}_{ij}}^2/2\sigma^2)$ and the normalization factor $Z^{\mathcal{H}} = \sum_{k \neq \ell} \exp{(-{d^{H}_{k\ell}}^2/2\sigma^2)}$.

Furthermore, we can split up $\frac{\delta C^{\mathcal{H}}}{\delta r^{\mathcal{H}}_{ij}}$ into two terms as follows:

$$-\sum_{k \neq \ell} p_{k\ell} \frac{\delta (\log q^{\mathcal{H}}_{k\ell} Z^{\mathcal{H}} - \log Z^{\mathcal{H}})}{\delta r^{\mathcal{H}}_{ij}} = -\sum_{k \neq \ell} p_{k\ell} \frac{\delta (\log q^{\mathcal{H}}_{k\ell} Z^{\mathcal{H}})}{\delta r^{\mathcal{H}}_{ij}} + \sum_{k \neq \ell} p_{k\ell} \frac{\delta (\log Z^{\mathcal{H}})}{\delta r^{\mathcal{H}}_{ij}}.$$

This allows us to tackle each term individually:

$$-\sum_{k \neq \ell} p_{k\ell} \frac{\delta \log q^{\mathcal{H}}_{k\ell} Z^{\mathcal{H}}}{\delta r^{\mathcal{H}}_{ij}} \tag{B.2}$$

$$\sum_{k \neq \ell} p_{k\ell} \frac{\delta \log Z^{\mathcal{H}}}{\delta r^{\mathcal{H}}_{ij}} \tag{B.3}$$

Term (B.2)

Note that $q^{\mathcal{H}}_{k\ell} Z^{\mathcal{H}} = \exp(\frac{-r^{H}_{ij}}{2\sigma^2})$ since we use a Gaussian. Taking the log of this (assuming natural log, otherwise we can do a change of basis and rewrite it as a natural log) we obtain $\log q^{\mathcal{H}}_{k\ell} Z^{\mathcal{H}} = \frac{-r^{H}_{ij}}{2\sigma^2}$ Furthermore, for $k \neq i$ and $\ell \neq j$ the terms in the sum are 0. We thus arrive at:

$$-\sum_{k \neq \ell} p_{k\ell} \frac{\delta \log q^{\mathcal{H}}_{k\ell} Z^{\mathcal{H}}}{\delta r^{\mathcal{H}}_{ij}} = -\sum_{k \neq \ell} p_{k\ell} \frac{\delta}{\delta r^{\mathcal{H}}_{ij}} (\frac{-r^{H}_{ij}}{2\sigma^2}) = p_{ij} \frac{1}{2\sigma^2} \tag{B.4}$$

Term (B.3)

If we focus on the $\frac{\delta \log Z^{\mathcal{H}}}{\delta r_{ij}^{\mathcal{H}}}$ term, and use the chain rule we obtain $\frac{1}{Z^{\mathcal{H}}}\frac{\delta Z^{\mathcal{H}}}{\delta r_{ij}^{\mathcal{H}}}$. From this we obtain a term $\frac{\delta Z^{\mathcal{H}}}{\delta r_{ij}^{\mathcal{H}}}$. We first focus on this term:

First of all we note that:

$$Z^{\mathcal{H}} = \sum_{k \neq \ell} \exp(\frac{-r_{k\ell}^{H}}{2\sigma^2}) \tag{B.5}$$

This means that when we take the derivative (variation) of this with respect to $r_{ij}^{\mathcal{H}}$, all terms where $k \neq i$ and $\ell \neq j$ are 0, thus we are left with the following:

$$\frac{\delta Z^{\mathcal{H}}}{\delta r_{ij}^{\mathcal{H}}} = \exp(\frac{-r_{ij}^{H}}{2\sigma^2})\frac{-1}{2\sigma^2} \tag{B.6}$$

Next, we have that $q_{ij}^{\mathcal{H}} = \frac{\exp(\frac{-r_{ij}^{H}}{2\sigma^2})}{Z^{\mathcal{H}}}$ This means we can rewrite the second part into the following:

$$\frac{\delta \log Z^{\mathcal{H}}}{r_{ij}^{\mathcal{H}}} = \frac{1}{Z^{\mathcal{H}}}\frac{\delta Z^{\mathcal{H}}}{\delta r_{ij}^{\mathcal{H}}} = \frac{\exp(\frac{-r_{ij}^{H}}{2\sigma^2})}{Z^{\mathcal{H}}}\frac{-1}{2\sigma^2} = q_{ij}^{\mathcal{H}}\frac{-1}{2\sigma^2} \tag{B.7}$$

Finally, combining everything together we get the following:

$$\begin{aligned}\frac{\delta C^{\mathcal{H}}}{\delta r_{ij}^{\mathcal{H}}} = -\sum_{k \neq \ell} p_{k\ell} \frac{\delta(\log q_{k\ell}^{\mathcal{H}} Z^{\mathcal{H}} - \log Z^{\mathcal{H}})}{\delta r_{ij}^{\mathcal{H}}} &= -\sum_{k \neq \ell} p_{k\ell}\frac{\delta \log q_{k\ell}^{\mathcal{H}} Z^{\mathcal{H}}}{\delta r_{ij}^{\mathcal{H}}} + \sum_{k \neq \ell} p_{k\ell}\frac{\delta \log Z^{\mathcal{H}}}{\delta r_{ij}^{\mathcal{H}}} \\ &= p_{ij}\frac{1}{2\sigma^2} - q_{ij}^{\mathcal{H}}\frac{1}{2\sigma^2} = \frac{1}{2\sigma^2}(p_{ij} - q_{ij}^{\mathcal{H}})\end{aligned} \tag{B.8}$$

So we get that:

$$\frac{\delta C^{\mathcal{H}}}{\delta r_{ij}^{\mathcal{H}}} = \frac{1}{2\sigma^2}(p_{ij} - q_{ij}^{\mathcal{H}}) \tag{B.9}$$

Plugging everything back into the main formula, and with ${d_{ij}^{\mathcal{H}}}^2 = r_{ij}^{\mathcal{H}}$ we get the following:

$$\frac{\delta C^{\mathcal{H}}}{\delta \mathbf{y}_i} = 2\sum_j \frac{\delta C^{\mathcal{H}}}{\delta r_{ij}^{\mathcal{H}}}\frac{\delta r_{ij}^{\mathcal{H}}}{\delta \mathbf{y}_i} = \frac{2}{\sigma^2}\sum_j (p_{ij} - q_{ij}^{\mathcal{H}}) \cdot d_{ij}^{\mathcal{H}} \cdot \frac{\delta d_{ij}^{\mathcal{H}}}{\delta \mathbf{y}_i} \tag{B.10}$$

Which is the expression for the variation of Hyperbolic SNE.

C

Python code for generating tree-like data (Distance Matrix)

The code below generates a distance matrix of some dataset following a tree-like structure. We do not need to generate an actual tree-like dataset as data embedding algorithms such as t-SNE do not use the data directly. Rather they first compute the distance matrix for the data and use that.

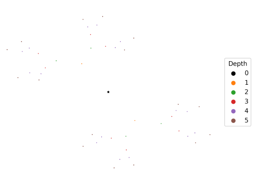
Each node in our artificial tree-like dataset consists of a cluster of points that together form the tree-node. One point per cluster (tree-node) is the center of that cluster to whom we compute the inter-cluster distances against (i.e. the distance between 2 points within a cluster is the distance of 1 point to the cluster center, and then the distance of that cluster center to the 2nd point). Distances of points between clusters are calculated as the distance of that point to it's cluster center, then the distance of that cluster center to another cluster center, then distance of that center to corresponding point. Finally to find the remaining distances the Floyd-Warshall algorithm is used.

```python
from scipy.sparse.csgraph import floyd_warshall
import numpy as np

def tree_D(n_children=2, depth=5, cluster_size=10, dist=5, floyd=True):
    # total nr. of clusters in the tree
    n_clusters = sum(np.power(n_children, d) for d in range(depth + 1))
    sigma = 1            # std. dev. of distances of points in a cluster
    mu = 0               # mean dist. between points in a cluster

    D = np.zeros((n_clusters, cluster_size, n_clusters, cluster_size))
    centers = np.random.randint(low=0, high=cluster_size, size=n_clusters)

    # Compute distances between points in a cluster and its center
    # and compute distances of cluster center to it's childrens' center
    for n in range(n_clusters):
        # Compute distances of nodes within a cluster
        # Choose random node in cluster to be the center
        center = centers[n]

        # Randomly generate a distance for points within a cluster
        D[n, :, n, center] = np.abs(sigma * np.random.randn(cluster_size) + mu)
        D[n, center, n, center] = 0

        # Compute distances of cluster center to cluster center of its children
        # i.e. find distances between neighbouring clusters
        for i in range(1, n_children + 1):
            child_n = n_children * n + i
            if child_n < n_clusters:
                child_n_center = centers[child_n]
                D[n, center, child_n, child_n_center] = dist

    # Use floyd-warshall to find remaining distances between clusters
    D = D.reshape(n_clusters * cluster_size, n_clusters * cluster_size)
    D = D + D.T
    D = floyd_warshall(D, directed=False)

    return D
```

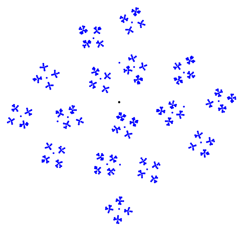
D Additional images

D.1. Embedding Tree-like data

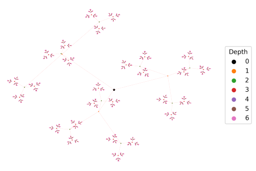
Additional images corresponding to experiment: 5.4. Included are the embeddings without edges for the smaller trees, and embeddings with edges for the larger trees.



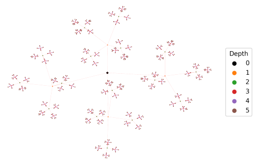
(a) 2 children per node, depth of 5. 192 total nodes



(b) 2 children per node, depth of 6. 448 total nodes

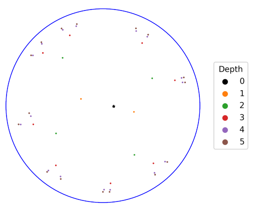


(c) 3 children per node, depth of 6. 5103 total nodes

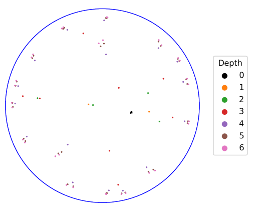


(d) 4 children per node, depth of 5. 6144 total nodes

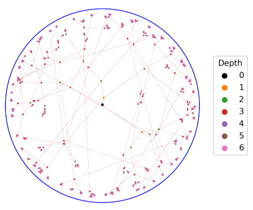
Standard t-SNE tree embeddings



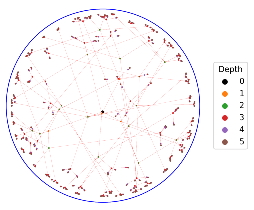
(e) 2 children per node, depth of 5. 192 total nodes



(f) 2 children per node, depth of 6. 448 total nodes

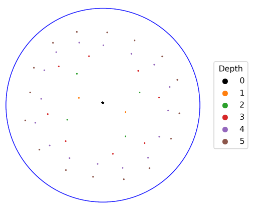


(g) 3 children per node, depth of 6. 5103 total nodes

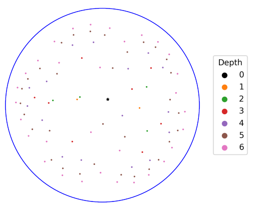


(h) 4 children per node, depth of 5. 6144 total nodes

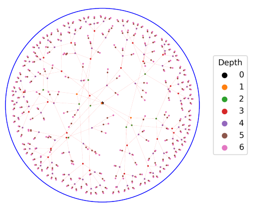
Hyperbolic t-SNE (t-distribution) tree embeddings



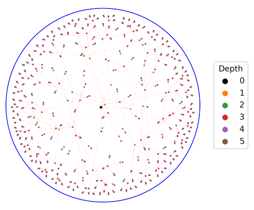
(i) 2 children per node, depth of 5. 192 total nodes



(j) 2 children per node, depth of 6. 448 total nodes



(k) 3 children per node, depth of 6. 5103 total nodes



(l) 4 children per node, depth of 5. 6144 total nodes

Hyperbolic SNE (Gaussian distribution) embeddings