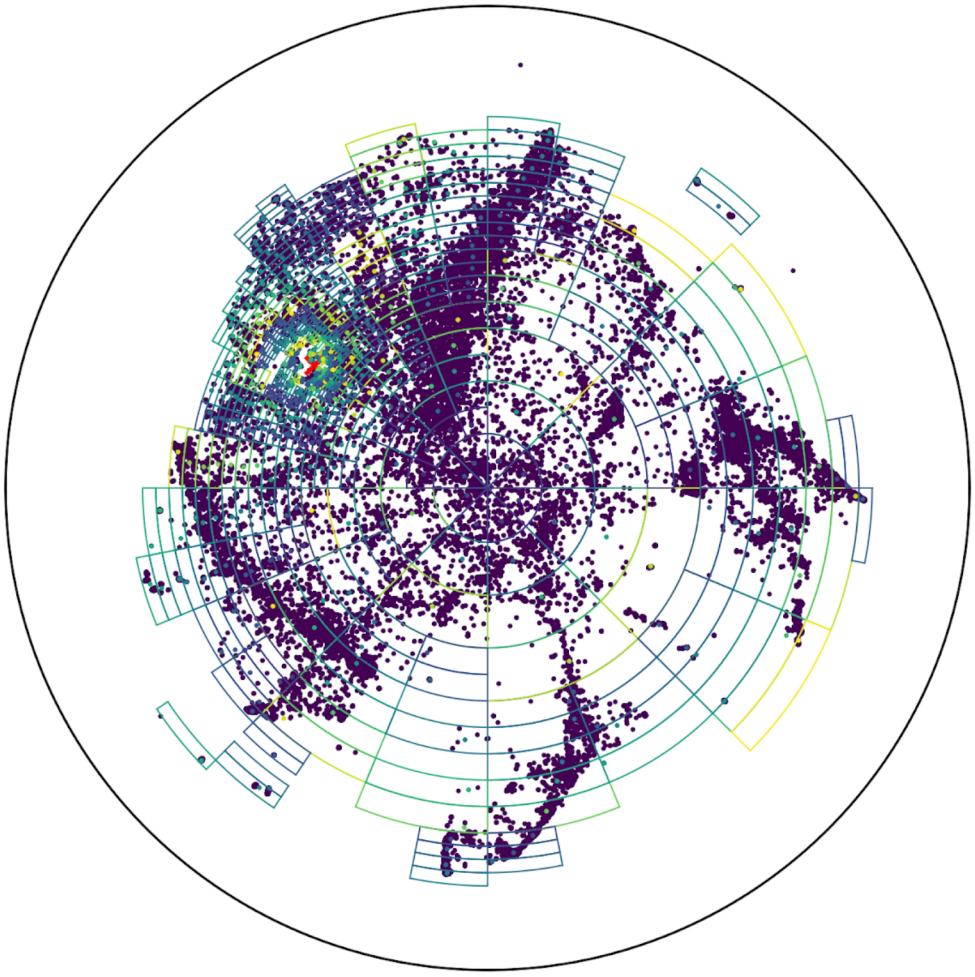
Accelerating t-SNE for hyperbolic embeddings

H.C. van Geffen



TU Delft

Accelerating t-SNE for hyperbolic embeddings

by

H. van Geffen

to obtain the degree of Master of Science
at the Delft University of Technology.
to be defended publicly on Tuesday September 27, 2022 at 13:00 PM.

Thesis committee:	Prof. Dr. E. Eisemann,	TU Delft, Thesis advisor
	Dr. K. Hildebrandt,	TU Delft, Daily supervisor
	Dr. M. Skrodzki,	TU Delft, Daily co-supervisor
	Dr. D. Tax,	TU Delft

TU Delft

Abstract

t-distributed stochastic neighbor embedding (t-SNE) is a dimensionality reduction technique able to embed high-dimensional data into 2 or 3 dimensions for the purpose of visualization, and has proven to be useful in various applications, such as single-cell analysis. Normally t-SNE embeds into Euclidean space, but recent work shows that embedding into hyperbolic space can lead to higher quality embeddings. Most notably on data containing hierarchical structures, as seen in single-cell measurements. As such it would be interesting to generalize computation of t-SNE embeddings to embeddings in hyperbolic space. A missing tool for this is the approximation of the objective and the gradient. This is needed as the cost for evaluating the objective and gradient is quadratic in the number of data points to be embedded. In practice such approximations are already used for Euclidean embeddings. In this thesis we introduce the first approximation scheme for hyperbolic t-SNE embeddings. To achieve this we generalize a Barnes-Hut approximation scheme to hyperbolic space. A major difficulty for this is to transfer the use of a quadtree in the Euclidean plane to hyperbolic space. We propose the usage of a polar quadtree in the Poincaré disk and show that our solution yields substantial gain in run time, in particular when embedding larger data sets, with the resulting embeddings being similar in quantitative and qualitative aspects.

Contents

1 Introduction

t-distributed stochastic neighbor embedding (t-SNE)(van der Maaten & Hinton, 2008) is a dimensionality reduction technique for visualizing high-dimensional data. It is able to embed high-dimensional data into 2 or 3 dimensions for the purpose of visualization, and has proven to be useful in various applications, such as single-cell analysis (Kobak & Berens, 2019). Research has been done on analyzing the hierarchical structure of single-cell measurements (Tanay & Regev, 2017), which has lead to visualization in low-dimensional Euclidean embeddings (Haghverdi et al., 2015), such as the ones created by t-SNE. But since complex tree-like networks, such as the hierarchical structures found in single-cell data, are better approximated by a hyperbolic space (Krioukov et al., 2010), it would be useful to generalize the computation of t-SNE embeddings to hyperbolic space. Some papers (Guo et al., 2022; Klimovskaia et al., 2020; Zhou & Sharpee, 2021) have already looked at doing this but a missing tool for this is the approximation of the objective and the gradient. This is needed as the cost for evaluating the objective and gradient is quadratic in the number of data points to be embedded. For Euclidean embeddings there are already some approximations which can be applied in practice, most notably Barnes-Hut t-SNE (van der Maaten, 2014), FIt-SNE (Linderman et al., 2019) and GPGPU t-SNE (Pezzotti et al., 2019).

In this thesis we introduce the first approximation scheme for hyperbolic t-SNE embeddings. To achieve this we generalize a Barnes-Hut approximation scheme to hyperbolic space. Barnes-Hut t-SNE is able to approximate its pairwise interactions in the gradient using the Barnes-Hut algorithm (Barnes & Hut, 1986). It is able to do so by laying out points in the Euclidean plane and computing a quadtree, which can be used to summarize point interactions. As such a major difficulty for the generalization to hyperbolic space is to transfer the use of this quadtree in the Euclidean plane to hyperbolic space.

Hyperbolic space has different models, all having different advantages and disadvantages. For our approximation scheme we use the Poincaré disk model, which is a 2d modal where all points are contained within the unit disk. We propose the usage of a polar quadtree in the Poincaré disk as a generalization of the quadtree in the Euclidean plane. As a polar quadtree is circular it naturally lends itself well to the division of space within a unit disk such as the Poincaré disk.

Besides the quadtree another difficulty arises from the requirement of the notion of center of mass of a collection points. In the Euclidean plane this is simply an averaging of point coordinates, but we are able to generalize this concept using the Einstein midpoint.

In the end we show that our solution yields substantial gain in run time, in particular when embedding larger data sets, while the resulting embeddings are similar in quantitative and qualitative aspects.

2

Related Work

Dimensionality reduction technique can be divided into two main approaches, linear and nonlinear. In the case of linear dimensionality reduction a traditional method is principal component analysis (PCA) (Hotelling, 1933). PCA is able to reduce the dimensions of high-dimensional data and has applications in fields such as research of the human genome, financial risk management or image compression. However linear projection only allows the data to be mapped using linear functions, this can lead to problems such as seen on the Swiss roll dataset, Figure 2.1.

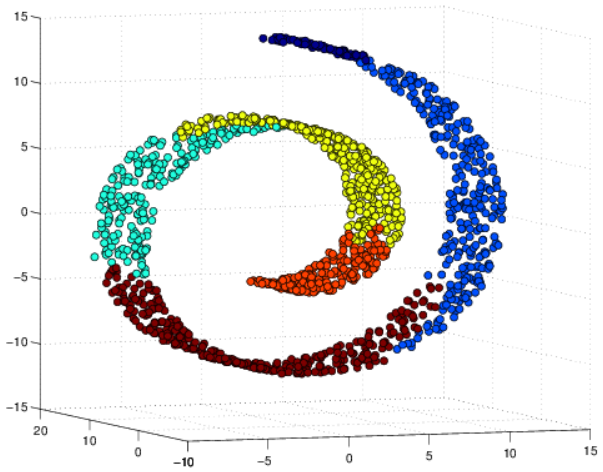


Figure 2.1: Swiss roll dataset

Source: (Yisheng Li, 2015)

Here PCA would describe the points in yellow to be closely related to the blue points to the right of them. But these points are actually quite distant when considering the Swiss roll structure which they lie on. This is because it is not able to learn the inherent structure, the Swiss roll structure not just the flat plane.

As such nonlinear dimensionality reduction techniques were developed which are able to capture the underlying structure of such high dimensional data as they can use more complex functions, giving more freedom and flexibility when mapping. There are two prominent nonlinear dimensionality reduction technique's we look at here, t-distributed stochastic neighbor embedding (t-SNE)(van der Maaten & Hinton, 2008) and UMAP (McInnes & Healy, 2018). t-SNE and

UMAP by default produce different visualization, both with their own merits, but they can actually produce quite similar images, t-SNE for instance can be used with constant exaggeration to produce a more UMAP like visualization (Kobak & Berens, 2019). Besides this UMAP is said to have better performance than t-SNE but this is in comparison to base t-SNE but similar run time can be achieved using approximations.

As complex tree-like networks, such as the hierarchical structures found in single-cell data, are better approximated by a hyperbolic space (Krioukov et al., 2010) there are a few works which have looked at combining t-SNE with a hyperbolic metric. Poincaré Map (Klimovskaia et al., 2020) for instance uses a modified t-SNE in combination with a hyperbolic metric. They first construct a k nearest neighbour graph to estimate local geometries and compute the global distances from the kNN graph using a Relative Forest Accessibility (RFA) (Chebotarev & Shamis, 2006) matrix. Their shows that the combination of these techniques and choices result in a better embedding than possible with other techniques. The influence of these changes however are difficult to determine, as most of their significance to the final results is not explained. But mainly missing is an approximation of the objective and gradient as they rely on taking subsamples of larger datasets.

Another technique h-SNE (Zhou & Sharpee, 2021) tries to improve upon the results of (Klimovskaia et al., 2020) by using their own g-SNE (Zhou & Sharpee, 2018) as a base. G-SNE is a version of t-SNE that uses an additional term in the cost function to take into account global structure. Using this they get similar local clusters but with distinct global patterns. h-SNE then replaces the distance metric of the global term in g-SNE with a hyperbolic one. They show improved visualization compared to the Poincaré Map on the same dataset. Again missing an approximation due to implementing base t-SNE.

co-SNE (Guo et al., 2022) takes a similar approach as h-SNE to embedding into hyperbolic space using t-SNE. They also create a balance in the cost function by adding an additional loss term creating a separation with two parts, one for capturing local information and one for global information. They however use a different term to capture global structure. Besides this they also change the metric for the high dimensional data to a hyperbolic one, replacing the distributions used for the similarities with a hyperbolic equivalent. As their paper does not provide a direct comparison of this with other hyperbolic versions of t-SNE it is difficult to see the influence of this change. Their tests show improved results over other techniques, but no comparisons to the above two methods. And again the use of exact t-SNE results in a quadratic run time for the computation of the objective and gradient.

Exact t-SNE has quadratic time complexity for evaluation of its objective and gradient functions, but new techniques have been developed to improve upon this. A few years after their paper on t-SNE (van der Maaten & Hinton, 2008) the main authors developed the Barnes-Hut t-SNE (van der Maaten, 2014) approximation. This approximation uses a technique used for n-body simulations to approximate t-SNE in loglinear time ($O(n \log(n))$, explained in section 4.1. Over the years even faster approximations have been found, namely FIt-SNE (Linderman et al., 2019) and GPGPU t-SNE (Pezzotti et al., 2019), both using interpolation, and are able to provide almost linear time complexity while also improving accuracy compared to the Barnes-Hut t-SNE approximation.

3

Background

3.1. t-SNE

t-distributed stochastic neighbor embedding (t-SNE) is a nonlinear dimensionality reduction technique allowing for visualization of high dimensional data. It reduces the amount of dimensions of high-dimensional data to a low-dimensional representation, 2 or 3d, which in turn is used for the purpose of visualization.

t-SNE can be applied to a high-dimensional dataset to create a low-dimensional embedding while trying to preserve relevant features of the data. To do this, we first calculate probabilities on the high-dimensional input data defining the similarities between data points. These probabilities are joined probabilities defined for every point pair. To calculate these we first require the conditional probabilities given by

$$p_{j|i} = \frac{\exp\left(-\|X_i - X_j\|^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-\|X_i - X_k\|^2/2\sigma_i^2\right)}. \tag{3.1}$$

here $p_{i|i} = 0$ and σ_i is the variance of the Gaussian centered on point X_i, which is derived based on a predefined perplexity parameter. The perplexity can be seen as defining how many neighbours each point should be concerned with. Based on the input datasets size this value is often chosen to be in the range from 5 to 50. Using binary search a value for σ_i is found which induces Gaussian's with this perplexity. To convert these conditional probabilities to joined probabilities we use

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n} \tag{3.2}$$

These are the probabilities for the high-dimensional representation. t-SNE starts out with either a random low-dimensional point embedding or one created using PCA and calculates its probabilities in a similar way using

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k\neq l}(1 + \|y_l - y_k\|^2)^{-1}} \tag{3.3}$$

t-SNE uses gradient descent on this low-dimensional representation to gradually align the low and high-dimensional probability distributions. Gradient descent does this by using an error function, in this case the Kullback–Leibler divergence, to determine the direction in which to change the low-dimensional embedding. The Kullback–Leibler divergence is a formula measuring the distance between two probability distributions and is defined as

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{3.4}$$

. t-SNE then iteratively reduces the error and repeatedly moves the embedding along the gradient

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(1 + ||y_i - y_j||^2)^{-1}(y_i - y_j) \tag{3.5}$$

until either a set amount of iterations has been reached or until convergence, which happens when the gradient becomes really small and as such the embedding will not change anymore. Resulting in a point mapping in low-dimensional space reflecting its high-dimensional counterpart.

Since at every iteration we have to determine for all point pairs the value of q_{ij} (Equation 3.3), summing over all other points, we have that t-SNE scales quadratically. As such this is part is the main speed bottleneck that approximations (section 4.1) try to tackle.

3.2. Barnes-Hut

Barnes-Hut t-SNE was one the first approximations developed for t-SNE and published by the original authors of t-SNE. It uses the Barnes-Hut algorithm (Barnes & Hut, 1986) most prominently used in astronomy to speed up n-body simulations. A variation of this algorithm can be applied to t-SNE using it to approximate the gradient. In this case the gradient (Equation 3.5) is split in two parts, the attractive forces F_{attr} and the repulsive forces F_{rep}

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4\left(F_{attr} - F_{rep}\right) = 4\left(\sum_{j \neq i} p_{ij} q_{ij} Z\left(\mathbf{y}_i - \mathbf{y}_j\right) - \sum_{j \neq i} q_{ij}^2 Z\left(\mathbf{y}_i - \mathbf{y}_j\right)\right) \tag{3.6}$$

where $Z = \sum_{k \neq l}(1 + \|y_l - y_k\|^2)^{-1}$. Here the attractive forces can be calculated efficiently in almost linear time, using a nearest-neighbour search for the p_{ij} term and noting that $q_{ij}Z = (1 + \|y_l - y_k\|^2)^{-1}$, which thus can be computed in $O(1)$ (van der Maaten, 2014). The repulsive forces however normally scale quadratically, this is due to the squared q_{ij} term's summation not being cancelled out by the Z term, as happens in the attractive case. The q_{ij} term requires the calculation of a pairwise distance matrix resulting in this quadratic scaling property. These repulsive forces are the part that is approximated using Barnes-Hut.

Instead of looping over all points when calculating the pairwise distance matrix, Barnes-Hut t-SNE uses a quadtree, a tree consisting of quads, an example of which can be seen in Figure 3.1, to subdivide the space and group related points together and loop over the grouped points instead. By merging points when sufficiently far away in this manner the algorithm reduces the necessary interactions it needs to calculate.

This process is computed in the following way, at every iteration of gradient descent the algorithm computes a new quadtree, which can be done in linear time ($O(n)$). The root of this tree is a bounding box which is large enough to contain all the points. Such a tree is created by initially creating this root bounding box containing all points, then inserting points one by one. When adding a point results in a box containing more than one point a split is performed, equally splitting the box horizontally and vertically into four sections. The point is then inserted into the respective bounding box which contains the point.

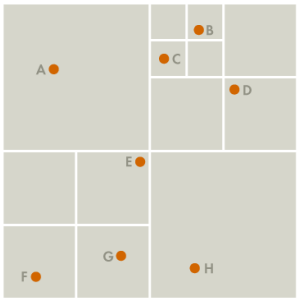


Figure 3.1: Quadtree

Source: (Tom Ventimiglia, 2011)

This tree can then be used to approximate forces. This is done by traversing the tree depth-first assessing at each node whether it can be summarized. The tree traversal takes on average $O(n \log(n))$. The condition for this is

$$\frac{||\mathbf{y}_i - \mathbf{y}_{cell}||^2}{r_{cell}} < \theta \tag{3.7}$$

This condition is met when the node's center of mass ($\mathbf{y}_{cell}$) is sufficiently far away from the respective point $\mathbf{y}_i$ and can be perceived as a single larger particle. Here r_{cell} is the maximum distance between any two points in a cell, in this case the diagonal of the cell. The θ is a parameter of the Barnes-Hut algorithm and can be tuned to the desired trade off between speed and accuracy. This value is normally set to $\theta = 0.5$ but even for values as large as $\theta = 1$ the forces are still accurate to $\sim 1\%$ with little dependence on the number of data points N (Barnes & Hut, 1986).

Even though the quadtree has to be created at every iteration the structure still gives great performance benefits for large enough datasets. This is because quadratic scaling grows so much faster than the loglinear time complexity ($O(n \log(n))$ of Barnes-Hut and the almost linear time complexity of the tree creation.

3.3. Hyperbolic Space

Hyperbolic space is a space that has constant negative Gaussian curvature. Gaussian curvature can be zero, positive or negative leading to three ways in which space can curve: (1) space is flat, (2) space is positively curved, or (3) space is negatively curved. Euclidean space has no curvature and in two dimensions is a flat plane. Here two parallel lines will never cross. When curvature is positive we get Spherical geometry, this would be analogous to the surface of a sphere. In this case parallel lines cross paths. Hyperbolic space has negative curvature everywhere and looks saddle shaped. This time parallel lines diverge leading to them never crossing each other. Figure 3.2 shows the three possible curved spaces visually and their influence on a simple element such as the triangle.

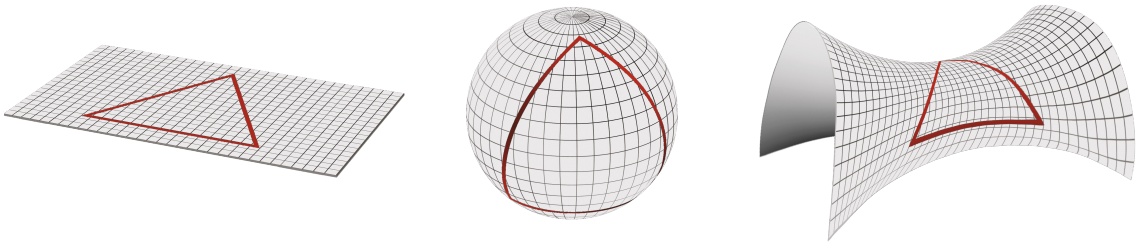


Figure 3.2: Left flat Euclidean space, middle spherical positively curved space, right negatively curved hyperbolic space

Source: (Haensch, 2021)

Mathematically speaking hyperbolic space is a Riemannian manifold. A Riemannian manifold is a smooth manifold equipped with a Riemannian metric. This smooth manifold is a name for a space which locally resembles Euclidean space and a Riemannian metric is the collection of inner products describing the relation between points in the space. This combination grants us the necessary tools to define points and calculate things such as distance and angles.

3.3.1. Why Hyperbolic?

Hyperbolic embeddings work especially well for tree-like data. This is any data with a hierarchical structure. Data with a hierarchical structure can range from genome data to the relations between vocabulary in languages. This tree-like property mathematically stems from the fact that the shortest path between a pair of points is almost the same as the path through the origin (Sala et al., 2018). More specifically, as we move away from the origin we have that for any two points, x and y, $d_H(x, y)$ approaches $d_H(0, x) + d_H(0, y)$, where d_H is the hyperbolic distance and 0 the origin. This reflects the property of trees where the shortest path between siblings is the path through their parent. An example of how hyperbolic space works well for tree-like data is shown below in Figure 3.3 where Poincaré map, a t-SNE like approach embedding to hyperbolic space, is able to display the inherent hierarchical structure but t-SNE embedding to Euclidean space is not.

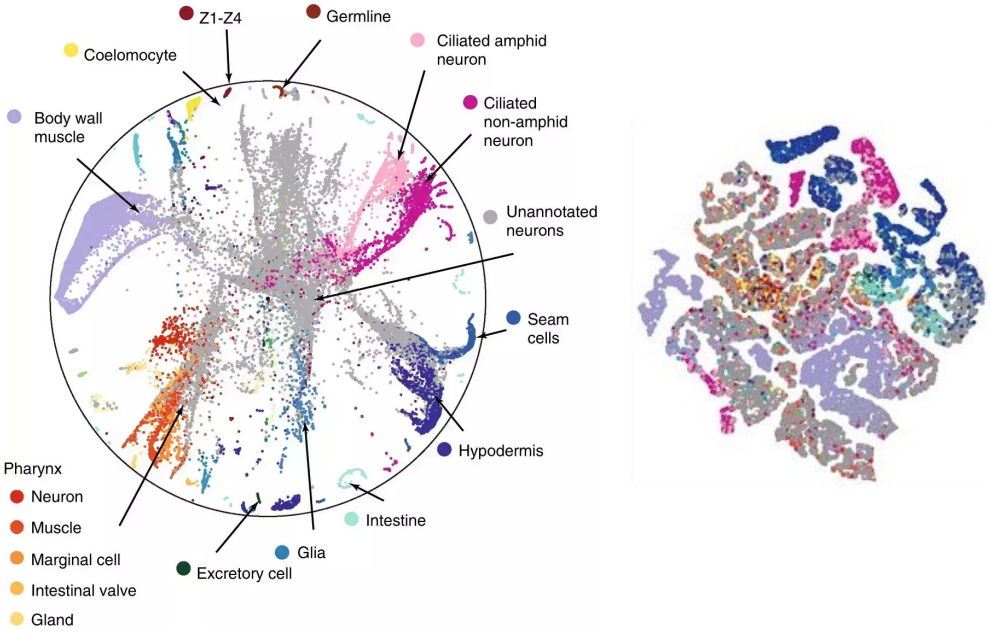


Figure 3.3: Left: C. Elegans dataset (Packer et al., 2019) embedded using Poincaré map (Klimovskaia et al., 2020)
Right: C. Elegans embedded with t-SNE.

Source: (Klimovskaia et al., 2020)

Hyperbolic embeddings also have in comparison to Euclidean embeddings an exponential advantage in space (Sala et al., 2018). As we move away from the origin in Euclidean space, space grows polynomial. In hyperbolic space, this is exponential. This allows for visuals more compactly displaying an entire embedding. This can be seen below in Figure 3.4 which shows a tree-like tessellation of triangles in hyperbolic space. Triangles appear to grow smaller as we go towards the edge but all triangles are the same size.

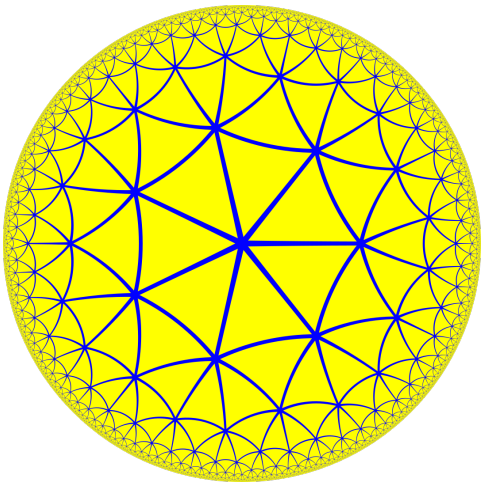


Figure 3.4: Triangular tessellation in the Poincaré disk

Source: (Taxel, 2018)

This space is best used when embedding short bushy hierarchies, meaning tree-like with a shallow depth, but many children per node. Graph data with long paths will take less advantage

of this extra space.

3.3.2. Hyperbolic Models

There are different models representing hyperbolic space, which each have distinct formulas for the Riemannian metric and coordinate systems for points. There is not one definitive model in the same way that there is no true mapping from the earth to a flat map. The choice of hyperbolic model has influence on a number of factors of t-SNE, some might have nice visual representations while others might show better performance.

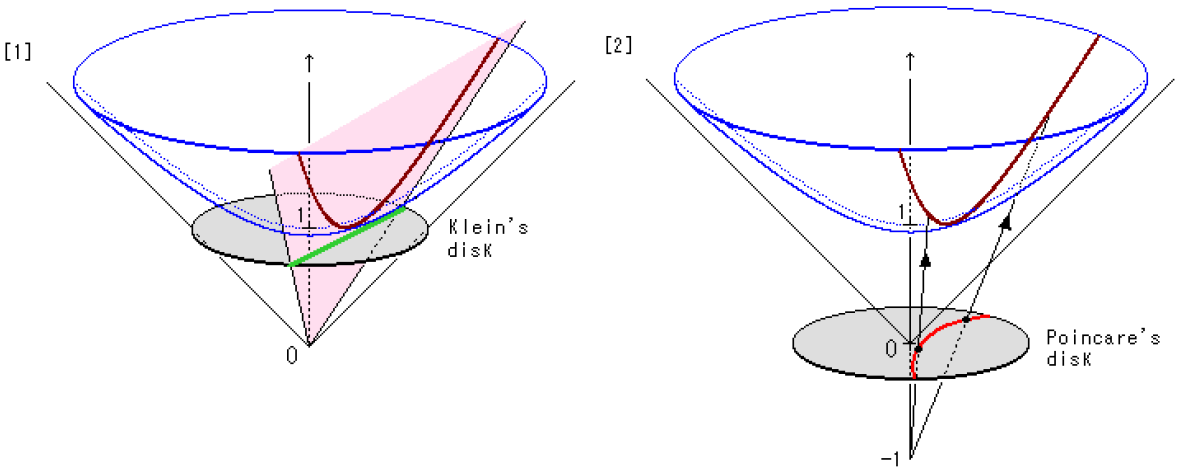


Figure 3.5: Left: relation between Beltrami Klein model (disk) and Lorentz model (blue hyperboloid) Right: relation between Poincaré disk model (disk) and Lorentz model (blue hyperboloid)

Source: (Ito, 2007)

Poincaré Disk Model

A common representations of hyperbolic geometry is the Poincaré disk model. This model is a two dimensional disk as seen on the bottom right in Figure 3.5 with its relation to the Lorentz model (Section 3.3.2). On manifolds, such as the Poincaré disk, the notion of "straight" lines (shortest path between two points) is called a geodesic. Notably the disk shows that geodesics are displayed curved on the Poincaré disk. This does not sound like a desirable trait visually, but all the models have trade offs. What makes this model desirable however is that it is a model that maps conformally to Euclidean space, meaning angles are preserved between hyperbolic and Euclidean space. When viewing an embedding in the Poincaré disk this means that circles and angles will not be distorted.

As everything is bounded in the unit circle and angles are preserved when viewing a final embedding in the Poincaré disk this gives a nice representation of points in a single image. Because of this, the model is also often used as a final visual representation of an embedding. In such a case, points are only converted to the Poincaré Disk Model after having been embedded using a different model. Doing this conversion at the end is often quite trivial and allows for a combination of the computational benefits of other models and the visual qualities of the Poincaré disk model.

One of the bigger downsides of performing gradient descent in the Poincaré disk is that it can suffer from numerical instabilities. These arise mainly from the fraction in its distance function (Equation 3.10), leading to floating point errors. Besides this, points quite quickly move towards the edge of the disk and distance grow to very large values. These values become numerically hard to handle for floating point notation, as they approach infinity.

In mathematical terms this model is a Riemannian manifold (B^d, g_x) where

$$B^d = \{x \in R^d \mid ||x|| < 1\} \tag{3.8}$$

is the open d-dimensional unit ball and

$$g_x = \lambda_x^2 g^E, \lambda_x = \left(\frac{2}{1 - ||x||^2}\right) \tag{3.9}$$

the metric tensor. A metric tensor is a function which allows one to measure distance in a given space encapsulating its geometrical characteristics. In the case of a Riemannian manifold this takes the form of a tensor taking as input two vectors in the tangent space $T_x\mathcal{M}$ and resulting in their scalar product. The tangent space $T_x\mathcal{M}$ is a space defined for every point on the manifold and in the case of 2 dimensional hyperbolic space is the plane tangent to that point, for example the plane in Figure 3.6.

Based on this notion of distance through the Riemannian metric a formula for the hyperbolic distance between two points $\mathbf{u}$ and $\mathbf{v}$ was defined to be

$$d(\mathbf{u}, \mathbf{v}) = \cosh^{-1}\left(1 + 2\frac{||\mathbf{u} - \mathbf{v}||^2}{(1 - ||\mathbf{u}||^2)(1 - ||\mathbf{v}||^2)}\right) \tag{3.10}$$

. During the gradient step of t-SNE we want to calculate the derivative of the distance function. As such given two points $\mathbf{u}$ and $\mathbf{v}$ the derivative of the above distance function with respect to $\mathbf{u}$ is

$$\frac{\partial d(\mathbf{u}, \mathbf{v})}{\partial \mathbf{u}} = \frac{4}{\beta\sqrt{\gamma^2 - 1}}\left(\frac{||\mathbf{v}||^2 - 2\langle \mathbf{u}, \mathbf{v}\rangle + 1}{\alpha^2}\mathbf{u} - \frac{1}{\alpha}\mathbf{v}\right) \tag{3.11}$$

where $\alpha = 1 - ||\mathbf{u}||^2, \beta = 1 - ||\mathbf{v}||^2$ and $\gamma = 1 + \frac{2}{\alpha\beta}||\mathbf{u} - \mathbf{v}||^2$. We can then use this in the gradient of t-SNE (Equation 3.5) as the derivative of the distance function as seen at the end of t-SNE gradient derivation which can be found in Appendix A.

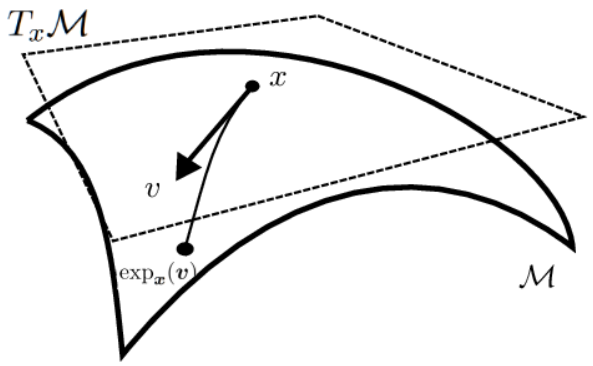


Figure 3.6: Point being projected using the exponential map

Source: (Tripuraneni et al., 2018)

Once we have the step for gradient descent in the tangent space, which is the Euclidean gradient at the corresponding point, we need to project this step onto the manifold. A simple manner in which to do this is to just project the points linearly downwards onto the manifold. This is called a retraction and in the case of the Poincaré disk an approximation of the projection onto the manifold. As such a better solution requires the usage of an exponential map. An

exponential map maps a vector from the tangent space onto the manifold, as seen in Figure 3.6. Derived in (Ganea et al., 2018) a closed form exponential map in the Poincaré disk is defined as

$$\exp_{\boldsymbol{x}}(\boldsymbol{v}) = \boldsymbol{x} \oplus \left(\tanh\left(\frac{\lambda_{\boldsymbol{x}} \|\boldsymbol{v}\|}{2} \right) \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|} \right) \tag{3.12}$$

where $\oplus$ is the Möbius addition (Ungar, 2008)

$$\boldsymbol{z} \oplus \boldsymbol{y} = \frac{\left(1 + 2\langle \boldsymbol{z}, \boldsymbol{y} \rangle + \|\boldsymbol{y}\|^2\right) \boldsymbol{z} + \left(1 - \|\boldsymbol{z}\|^2\right) \boldsymbol{y}}{1 + 2\langle \boldsymbol{z}, \boldsymbol{y} \rangle + \|\boldsymbol{z}\|^2 \|\boldsymbol{y}\|^2} \tag{3.13}$$

.

Some manifolds have very complex exponential maps which limit their ability to be able to be calculated exactly at every step of gradient descent. The Poincaré disk having a closed form solution allows us to skip translating to a different model for this step. Although other models do still have simpler closed forms, such as the Lorentz model (Equation 3.3.2).

When using the Poincaré disk with t-SNE there are some hyperparameters that behave differently. Perplexity works similar as in the Euclidean case as it only affects the probability values of the high dimensional representation, which still uses the Euclidean metric. Learning rate however has to be drastically lower, one hundredth of the Euclidean learning rate, as the coordinate values are numerically scaled from 0 to 1 instead 0 to infinity. A too high learning rate will lead to points taking a too large step during gradient descent and stepping outside the disk. This can also happen in general even when using an exponential map, in contrast to a retraction. As such we project points back into disk in case this happens using

$$\text{proj}(\boldsymbol{x}) = \begin{cases} \boldsymbol{x}/\|\boldsymbol{x}\| - \varepsilon & \text{if } \|\boldsymbol{x}\| \geq 1 \\ \boldsymbol{x} & \text{otherwise} \end{cases} \tag{3.14}$$

from (Nickel & Kiela, 2017).

Lorentz Model

The Lorentz model, also called the hyperboloid model, is a model of hyperbolic space often used for its computational qualities. It can be seen in Figure 3.5 as the blue hyperboloid on either side contrasted with the other mentioned models.

In comparison to the the Poincaré Disk Model it remedies some of the numerical instabilities and allows for simpler formulas for the exponential map. Here the biggest difference numerically is that it does not have a fraction in the distance formula. Because of this it has recently gained traction in hyperbolic embedding research (Wilson & Leimeister, 2018).

As points in this model are defined unbounded in three dimensional space it is more difficult to obtain a clear overview of the desired information in a single image. Therefore as mentioned before points are often translated to the Poincaré disk model after having been embedded using this model.

Here we do not use the Lorentz model, as our approximation, see section 4.1, depends on the Poincaré disk and we are able to overcome some of its shortcomings. It is however easy to translate back and forth and as such it would be interesting to in the future use the Lorentz model for a potentially more stable embedding calculation.

Beltrami-Klein Model

The Beltrami-Klein model, seen as the two dimensional disk on the bottom left of Figure 3.5, models hyperbolic space in such a way that geodesics are visually straight lines on the disk. This may seem as a nice visual property but in practice the fact that it trades off this property by distorting angles and thus circles leads to a less desirable visual in comparison to the Poincaré

disk. However as we perform gradient descent on a point embedding the notion off non distorted angles and straight lines are less important as t-SNE is noisy in and of itself. As such we could have applied the Beltrami-Klein model instead of the Poincaré disk model but as it is less used in related research we ultimately ended up following suit. We do however apply the Beltrami-Klein model numerically by going back and forth for the calculation of a single function (Einstein midpoint subsection 4.1.3).

4

Method

4.1. Approximation

Base t-SNE does not scale well with datasets as they grow larger. This is because the cost of evaluating its objective and gradient scales quadratically in the number of data points. Over time approximations have been developed to mitigate this. Some of the more prominent ones are Barnes-Hut t-SNE(van der Maaten, 2014), FIt-SNE (Linderman et al., 2019) and GPGPU t-SNE(Pezzotti et al., 2019).

4.1.1. Polar Quadtree

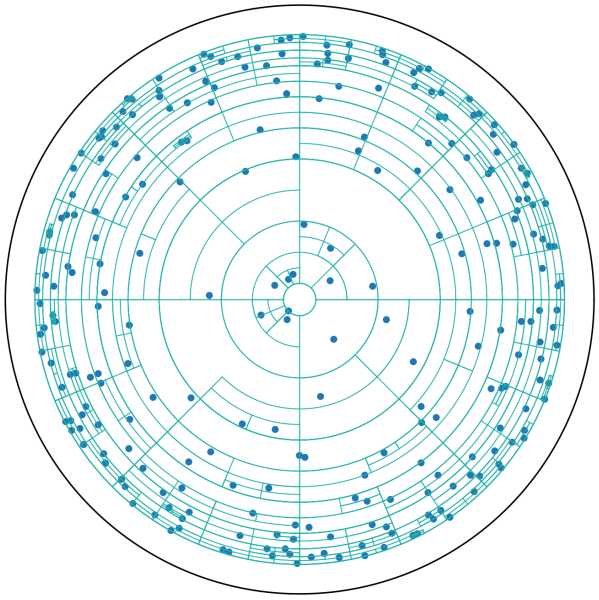


Figure 4.1: Polar quadtree in the Poincaré disk

If we were to take a similar approach to Barnes-Hut t-SNE to approximate an embedding into hyperbolic space there are some things which do not generalize to the hyperbolic case. One big thing is that the quadtree structure relies on creating a grid on the Euclidean plane where each cell subdivides into four equally sized squares. In the case of hyperbolic space, however, this is not directly possible. There are a few ways to tile hyperbolic space (Conway et al., 2016) but none which subdivides internally into similar tilings, i.e., none that brings a hierarchical

structure such as the Euclidean quadtree. One structure for the Poincaré disk we can employ is the polar quadtree (Looz et al., 2015), as seen in Figure 4.1. Here points are defined using polar coordinates and a cell has bounds for range and angle instead. These bounds form a polar rectangle as seen in Figure 4.2.

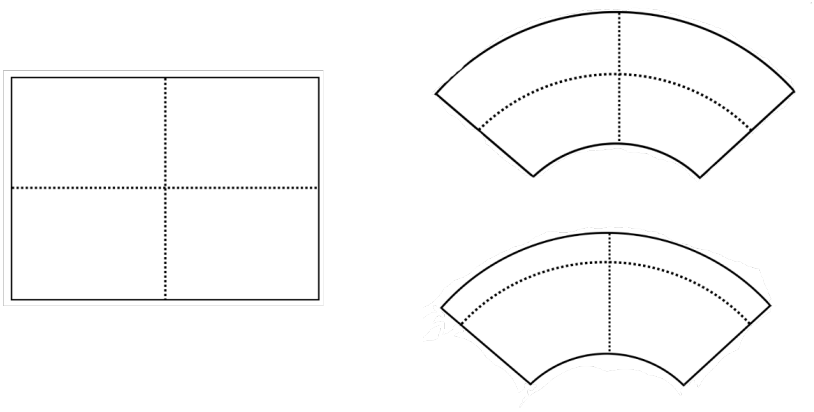


Figure 4.2: Left: A quad as seen in the Euclidean quadtree rectangle and its splitting
Right: Polar rectangles from a polar quadtree with the top one using equal side lengths splitting and the bottom one using equal area splitting

Initially we start with a root cell whose bounds are defined to contain the entire embedding. This means that its range minimum is the minimum range of the points of the embedding and its maximum is the maximum of the points in the embedding, this can be seen in Figure 4.3.

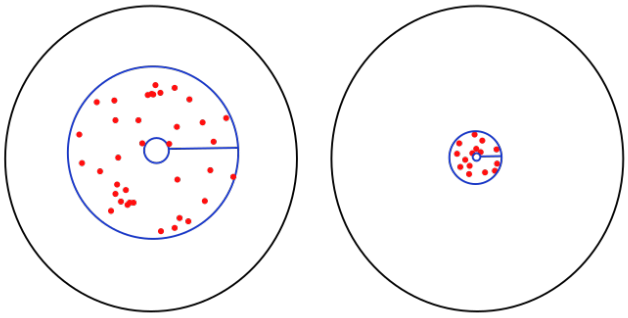


Figure 4.3: Left initial root cell for a larger embedding
Right initial root cell for a smaller embedding.

For the angular bound of the root we always use the complete angular bound setting it to a minimum of 0 and a maximum of 2π. After this we have to define how to split this root node to subdivide the space and create the quadtree.

Splitting The Tree
When splitting a cell to create child cells we create four new quadrants. In the polar quadtree this is done by splitting across range and angle.

Equal side lengths splitting In the Euclidean case splitting would entail halving a square both horizontally and vertically each time. In the hyperbolic case we could also split both range

and angle in halve in the same Euclidean way. This results in the equal side lengths splitting method with splitting happening according to

$$\text{mid}_r = \frac{\min_r + \max_r}{2}$$

$$\text{mid}_\theta = \frac{\min_\theta + \max_\theta}{2}$$

.

Equal area splitting Another interesting approach is to split the range to account for the hyperbolic properties of the space. One such way is to split the range using the formula from (Looz et al., 2015). This equal area splitting then results in splitting happening according to

$$\text{mid}_{r_\mathcal{H}} = \text{acosh}\left(\frac{\cosh\left(\alpha \max_{r_\mathcal{H}}\right) + \cosh\left(\alpha \min_{r_\mathcal{H}}\right)}{2}\right) \cdot \frac{1}{\alpha} \tag{4.1}$$

$$\text{mid}_\theta = \frac{\min_\theta + \max_\theta}{2}$$

.

Here $r_\mathcal{H}$ is the hyperbolic range and α a growth parameter where $\alpha = 1$ gives a uniform distribution on hyperbolic space in the Poincaré disk. This formula is used to split the cells radially in such a way that both halves have equal hyperbolic area, giving a uniform distribution.

4.1.2. Key properties of Barnes-Hut

When translating the Quadtree to the hyperbolic setting there is a trade off to be made for the desired properties of the Barnes-Hut technique. The most important ones to making the Barnes-Hut algorithm work, allowing it to operate in loglinear time ($O(n \log(n))$, are (1) that the tree is as balanced as possible for an average distribution of points and (2) that the error, the maximum distance between any 2 points in a quadrant, almost halves at each layer as we traverse down the tree.

In the Euclidean case, the tree is guaranteed to be maximally balanced as the area is the same for all child quadrants of a parent. In the hyperbolic case, we can, instead of just splitting angle and range in half, use equal area splitting (Equation 4.1), which should help in making the tree balanced as this also ensures equal area in the hyperbolic setting.

For the error reduction it is important that going down a layer in the tree should reduce the maximal distance (the error) within a cell as much as possible. There are two important differences between the Euclidean cells and the hyperbolic ones for this, the maximal error of a cell and its aspect ratio.

The maximal error in a cell is the maximum distance between any two points within such a cell. In the Euclidean case this is simply the diagonal of the quadrant. But in the polar tree the resulting quadrants do not have the property that their maximum is the diagonal, but instead it is always either the top edge or a side edge, left or right as they are equal in length.

The aspect ratio of a cell is the ratio of the length of the top edge and of one of the side edges, similar to the aspect ratio of an image. In the Euclidean case this is always 1:1 as they are both of equal length. But in the hyperbolic setting this ratio is not 1:1 as either edge, often the top one, has a larger length than the other. If this grows too much, we get less "squarish" cells resulting in the algorithm not summarizing enough points.

4.1.3. Center of mass

During Barnes-Hut t-SNE (van der Maaten, 2014) the points which are summarized together will have their forces be approximated by being treated as a single larger point located on the center of mass of the respective points. In the case of the base Euclidean Barnes-Hut t-SNE this center of mass is just the Euclidean mean of the points. The natural hyperbolic equivalent would be the Fréchet mean (Fréchet, 1948). The Fréchet mean is given as a solution to the optimization problem

$$\mu_{fr} = \arg\min_{\mu \in \mathcal{M}} \sum_{i=1}^{N} d\left(x_i, \mu\right)^2 \tag{4.2}$$

Here μ is the mean which minimizes the sum of squared distances to the points x_i on the manifold. This unfortunately has no closed form solution and as such requires an iterative solver.

A few papers have circumvented this problem by applying a pseudo-Fréchet mean. One of these is the Einstein midpoint (Gulcehre et al., 2018), which has a nice closed form solution

$$m_i\left(\left\{\alpha_{ij}\right\}_j, \left\{\mathbf{v}_{ij}\right\}_j\right) = \sum_j \left[\frac{\alpha_{ij}\gamma\left(\mathbf{v}_{ij}\right)}{\sum_\ell \alpha_{i\ell}\gamma\left(\mathbf{v}_{i\ell}\right)}\right] \mathbf{v}_{ij} \tag{4.3}$$

with the lorentz factors $\gamma\left(\mathbf{v}_{ij}\right)$ defined as

$$\gamma\left(\mathbf{v}_{ij}\right) = \frac{1}{\sqrt{1 - ||\mathbf{v}_{ij}||^2}} \tag{4.4}$$

. In our case the weights α_{ij} and α_{il} are all just set to 1. Here the points $\mathbf{v}_{ij}$ are in the Klein model and as such points from the Poincaré disk model will have to be converted back and forth. Although this is not really a problem since such conversions have minimal overhead.

These pseudo-Fréchet means however do not truly solve the Fréchet mean. (Lou et al., 2020) show in their experiments that the Einstein midpoint suffers from a 6.8% relative error compared to the true mean. Besides this they also propose their own solution which results in an exact Fréchet mean. Implementation wise this solution is more involved and it also lacks one key property of the Einstein midpoint which is that it can be calculated as a moving average. This means that when building the tree at every point insertion we do not have to loop over all the points in the cell. This small property could potentially make it not viable run time wise for our implementation.

5

Experiments

This chapter discusses different experiments that were performed to asses the properties of the approximation and the used polar quadtree.

5.1. Performance as the input size increases

As the goal of using the Barnes-Hut like approach is to improve the performance we would like to be able to show that it improves it in a way that its growth rate, the rate at which the running time grows as we increase the number of data points, has a substantial change. In this case the base t-SNE relies on n-body force calculations which grow quadratic $O(n^2)$ in run time as the input increases. The Barnes-Hut t-SNE in Euclidean improves this run time to be loglinear $O(n \log(n))$ by summarizing point clusters using a quadtree. Here we would like to determine in what way our solution improves the run time and to see how this compares to the Euclidean version.

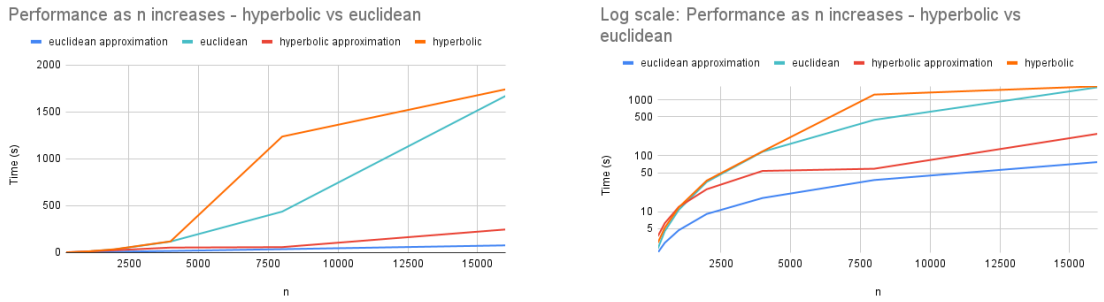
Hypothesis We expect our solution to in practice improve the growth rate of the algorithm. This growth rate change will most likely be similar to the original Barnes-Hut version of t-SNE and thus result in a loglinear $O(n \log(n))$ algorithm.

However since the properties of the tree define the upper bound of computational complexity our solution will probably differ. This is because the original used a Euclidean quadtree which always behaves the same independent of the bounds of the total embedding, if points were to grow outward away from the center the properties of the resulting tree would not change. In the hyperbolic case as the embedding grows and takes up more space in the Poincaré disk the properties, such as quadrant aspect ratio, of the tree changes. Therefore and because we use more complex distant function calculations we think our solution will perform slower than the Euclidean counterpart but still behave similarly when considering the asymptotic behavior.

Method To test this we calculate all 4 versions of t-SNE, the Euclidean and hyperbolic versions and both their approximations. We run these 4 versions on the MNIST dataset, a collection of 70.000 images with a resolution of 28x28. As this dataset is not intrinsically hyperbolic it will not necessarily benefit from the usage of hyperbolic t-SNE but it is still a research wide benchmark for many different data processing applications and will serve as a fine performance benchmark. We sample points starting from $n = 250$ and doubling for each run.

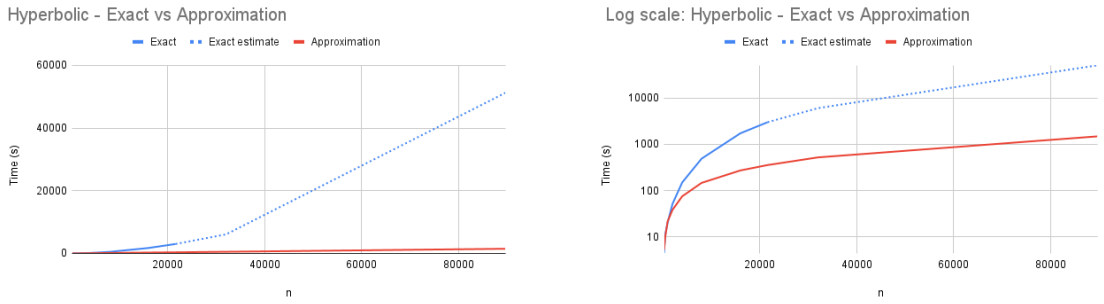
For the hyperbolic approximation we also run it on some larger datasets, PLANARIA (Plass et al., 2018) and C. Elegans (Packer et al., 2019) to provide a more detailed direct comparison to its exact version.

Results These runs resulted in the following graphs



(a) Normal scale (b) Log scale

Figure 5.1: Performance on MNIST for the different techniques as n increases



(a) Normal scale (b) Log scale

Figure 5.2: Performance on different datasets (lower values of n using mnist subsamples and higher testing on PLANARIA and C. Elegans) for hyperbolic t-SNE as n increases with dotted estimates based on first few iteration durations

Discussion In Figure 5.1 we see a comparison between our base and approximation and the base and approximation of Euclidean t-SNE. As we can see it show us that our solution is slower than Euclidean but does indeed grow similarly. In Figure 5.2 we see just our solution, comparing base and approximate, were we clearly see the performance gains of our solution especially as numbers grow even larger than the previous experiments.

5.2. Splitting condition of the Polar quadtree

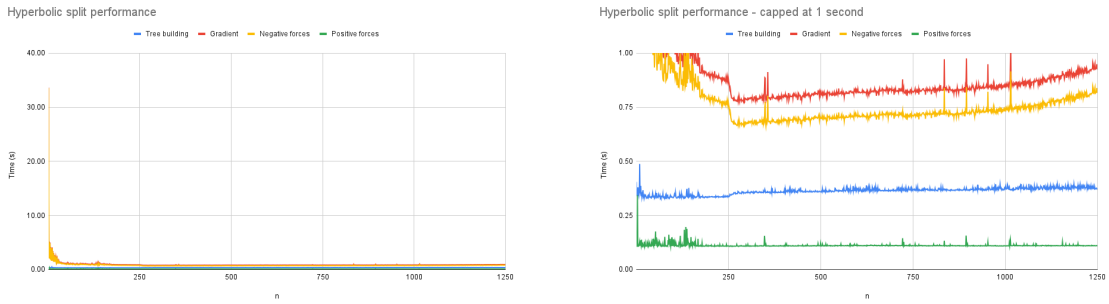
When splitting the quadrants in the Polar quadtree we would like to create the optimal tree. This involves (1) maintaining a good aspect ratio to smooth the error reduction at each layer and (2) have points be evenly split among the different sub-quadrants. For this second point it would be interesting to test if using an equal area split as described in Figure 4.1.1 results in a better performing tree (more summarizations).

Hypothesis We expect the equal area split to perform better as it takes into account the hyperbolic nature of the space leading to better point coverage. It could however suffer from bad quadrant aspect ratios resulting in very longish squares were the max error then becomes one of these long sides, resulting in less summarizations.

Method To test this we run both versions on the Planaria dataset (Plass et al., 2018) and time for each iteration in the process the duration of the negative force calculations (the part which summarizes using the tree). Next to this we compare with section 5.1 using the same methodology.

We also inspect the shapes of the summarized quadrants of the tree to see how the shapes hold up when applied at such scale.

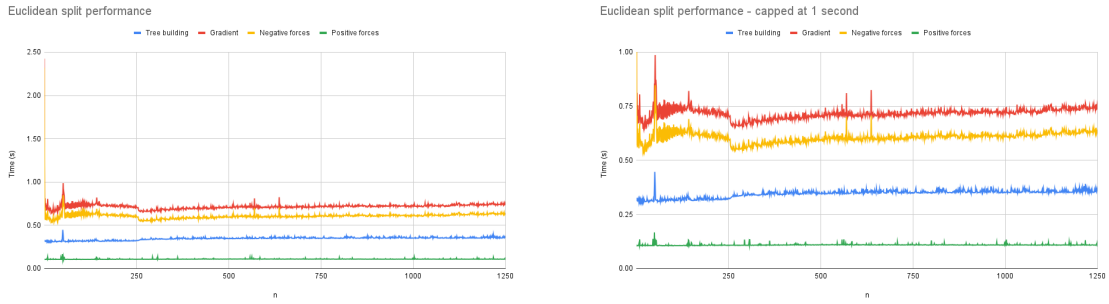
Results The experiment resulted in the following graphs



(a) Full view

(b) View with time capped at 1 sec

Figure 5.3: Timings of different parts of the gradient calculation of the Planaria dataset with equal area split splitting at iteration n



(a) Full view

(b) View with time capped at 1 sec

Figure 5.4: Timings of different parts of the gradient calculation of the Planaria dataset with equal side lengths splitting at iteration n

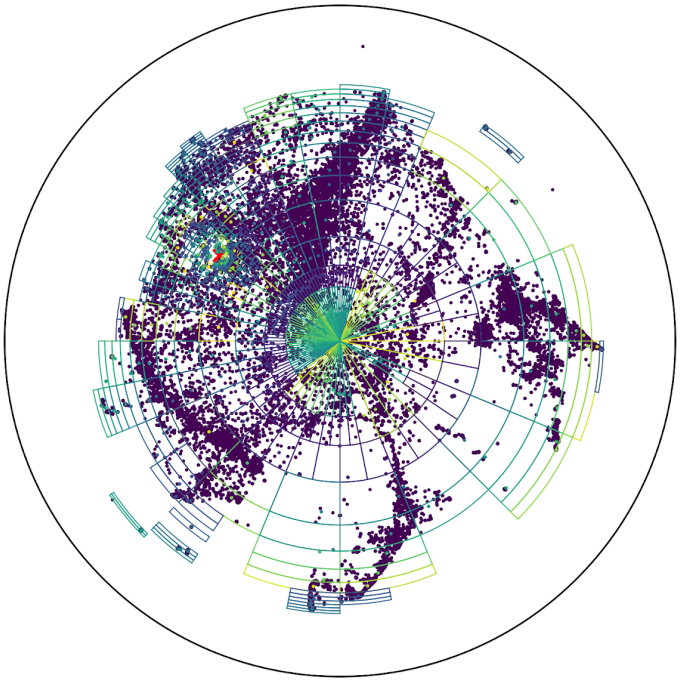


Figure 5.5: Example of summarized squares on Planaria using equal area split

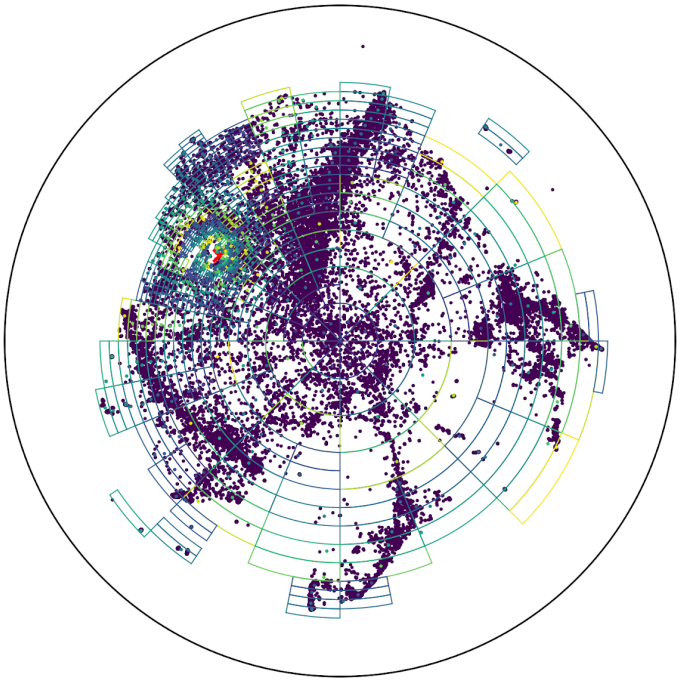
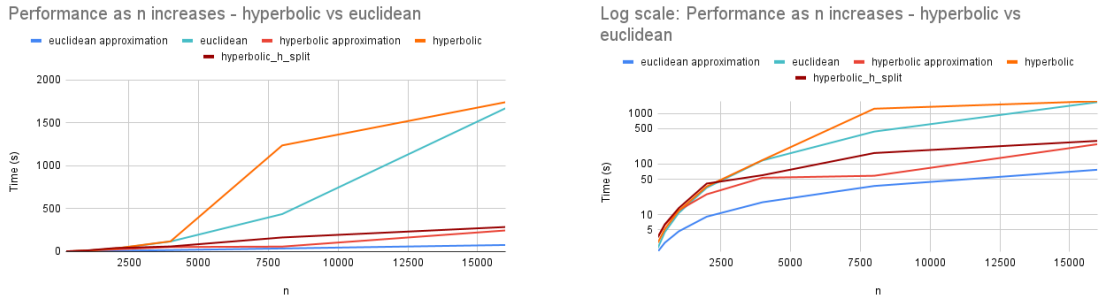


Figure 5.6: Example of summarized squares on Planaria using equal side lengths split



(a) Normal scale

(b) Log scale

Figure 5.7: Performance on MNIST for the different techniques as n increases with equal area split

Discussion As can be seen in the equal area split Figure 5.3(a) there seems to be a very slow start to embedding the data, almost the speed as the exact version, especially when comparing to the equal side lengths split Figure 5.4(a), which shows similarities to the spikes seen in the Euclidean case. Even when capping at 1 second, Figure 5.3(b) and Figure 5.4(b), we can still see that the equal side lengths split is performing better overall. This problem seems to stem from the fact that the equal area split creates very thin rectangles, as seen in Figure 5.5. When comparing to Figure 5.6 we see that the tree is much more balanced, in the sense that the "squares" are more even. As the results when adding the equal area splitting method to the previous experiment, section 5.1, as seen in Figure 5.7 also do not show an improvement in performance we think that this aspect ratio of the quadrants overshadows the potential gain from uniformly splitting hyperbolic space to better space out the distribution of points.

5.3. Cause of the splitting condition difference

When applying the splitting conditions to t-SNE we get that the equal side lengths splitting leads to better performance. It could be that the equal area guarantees is not as important due to the fact that hyperbolic t-SNE resides for most of its time in the center of the Poincaré disk, which is more aligned with Euclidean properties when compared to the edge of the disk. Another option is that we want to be as close as possible to a 1:1 aspect ratio for the quads. Here we test whether, given perfect conditions, the equal area split's are property can assist in summarization performance.

Hypothesis As a previous experiment, section 5.2, has shown that in practice equal area splitting performs less good, we suspect it is indeed the case that the combination of the relative insignificance in the difference of area properties of the splits and the significance of the aspect ratio leads to, in the case of t-SNE, that the equal side lengths performs better.

Method We begin by creating a point sampling for our test dataset. This is done by fully splitting a polar quadtree with equal area splitting up till depth 4, then placing a single point randomly into each leaf node. We then insert these points into two trees one with again equal area splitting and one with equal side lengths splitting checking for each point how many points it is able to summarize and the tree depth of all the points.

Results The experiment resulted in the following graphs

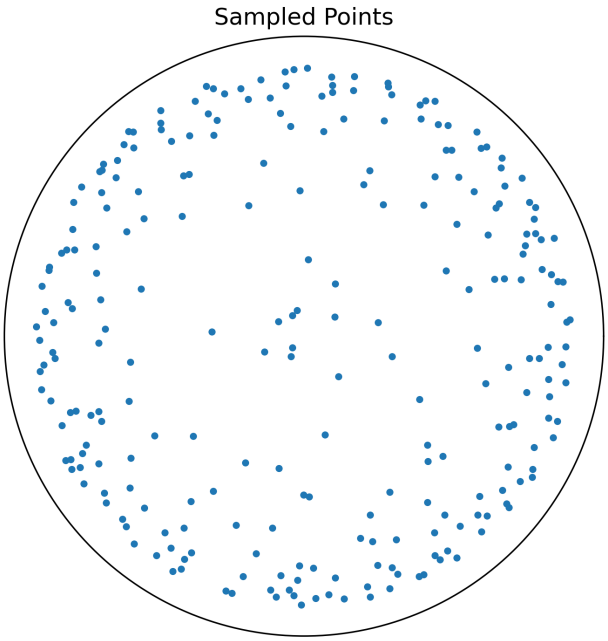
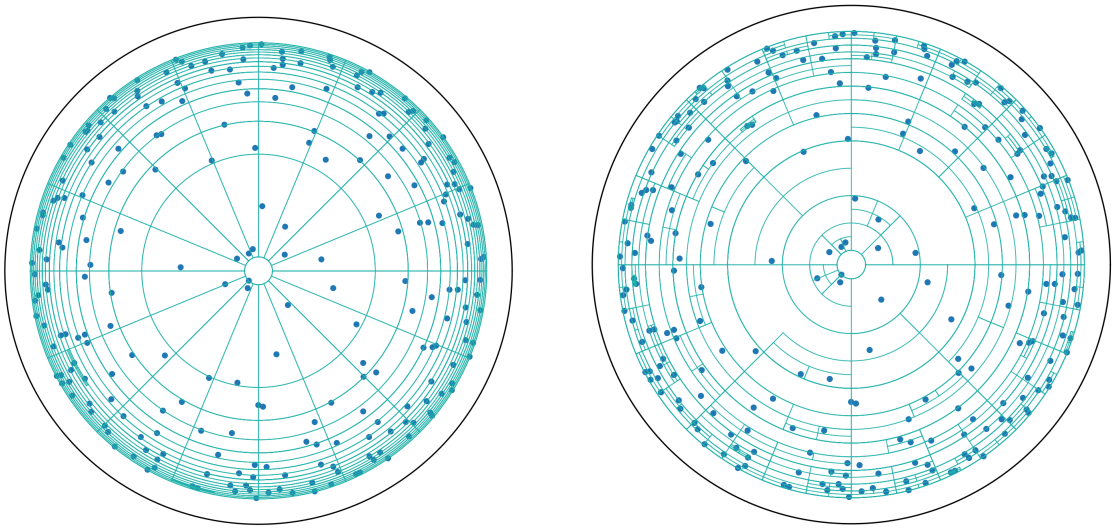


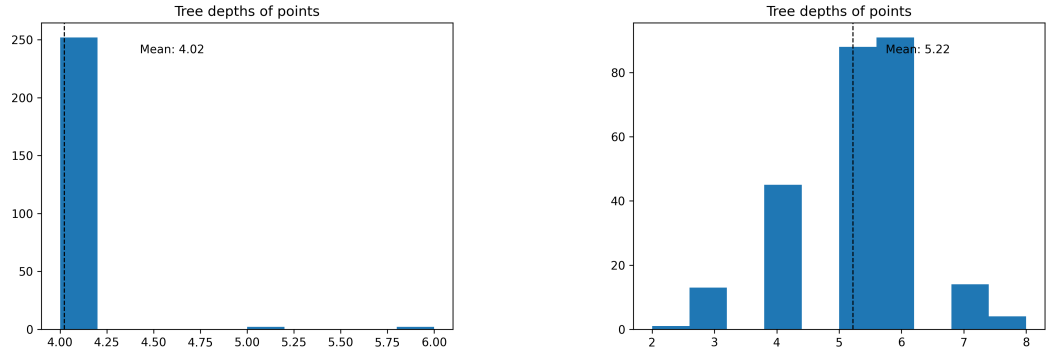
Figure 5.8: Points sampled using equal area splitting polar quadtree



(a) With equal area splitting

(b) With equal side lengths splitting

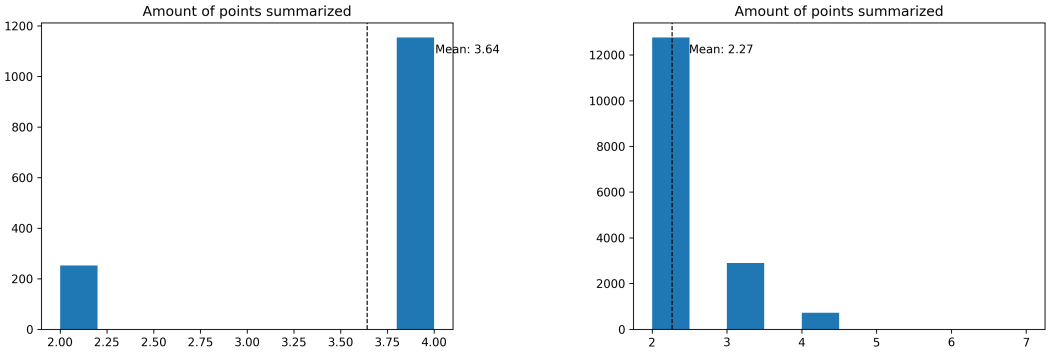
Figure 5.9: Sampled points added to a polar quadtree



(a) With equal area splitting

(b) With equal side lengths splitting

Figure 5.10: Histogram displaying the depth of points



(a) With equal area splitting

(b) With equal side lengths splitting

Figure 5.11: Histogram counting the number of summations greater than 1

Discussion The sampling using the equal area splitting tree resulted in a sampling as seen in Figure 5.8. As this experiment was setup in such a way that the equal area splitting would be perfect, we can indeed see in Figure 5.9 that the equal area tree has split perfectly, in comparison to the equal side lengths splitting tree. We also see that because of the perfect split we get that the tree depth does not exceed $d = 4$ (Figure 5.10), except for 2 outliers due to the difference in initial tree node min and max range. For the equal area split tree we see that its depths average $d = 5.22$ meaning that points are deeper in the tree and that the performance of our approximation should suffer. However when looking at Figure 5.11 we see that the equal side lengths split is able to summarize way more points using its construction, taking note of the differing y axis ranges we see 1200 for equal area and sometimes 12000 for equal side lengths splitting. As such it looks like it is indeed the aspect ratio property that outshines the gain in tree depth, in this case $d = 5.22 - 4.02 = 1.20$.

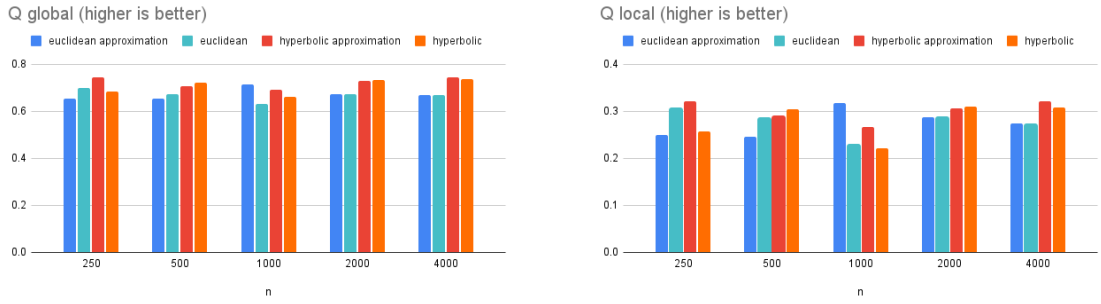
5.4. Accuracy of the tree

An approximation's final embedding should not be too different from the base version. In the case of the Euclidean Barnes-Hut t-SNE, the quality of the constructed embeddings is negligible (van der Maaten, 2014). Here we try to determine if our solution behaves in a similar way as its Euclidean counterpart with respect to accuracy.

Hypothesis We expect our solution to produce embeddings which are a good enough approximation of the data doing so in a similar way to the Euclidean counterpart. Some more approximate embeddings may occur from the slightly inaccurate Einstein midpoint used but since these are already sufficiently far away the solution should still perform well. We also expect the quality of the exact embeddings to be higher.

Method To test this we run both versions on MNIST (LeCun, 1998) as in section 5.1 and determine the scale-independent quality criteria (Lee & Verleysen, 2010) as also used in (Klimovskaia et al., 2020) to determine the quality of the resulting embeddings in its ability to capture global and local structure and to determine the differences between the methods.

Results The experiment resulted in the following graphs



(a) Q global

(b) Q local

Figure 5.12: Results from the scale-independent quality criteria on MNIST

Discussion These results show that based on the scale-independent quality criteria, globally Figure 5.12(a) and locally Figure 5.12(b), our solution behaves in a semi similar way to the Euclidean versions, although both do not always show the expected result of the exact version having a better quality score.

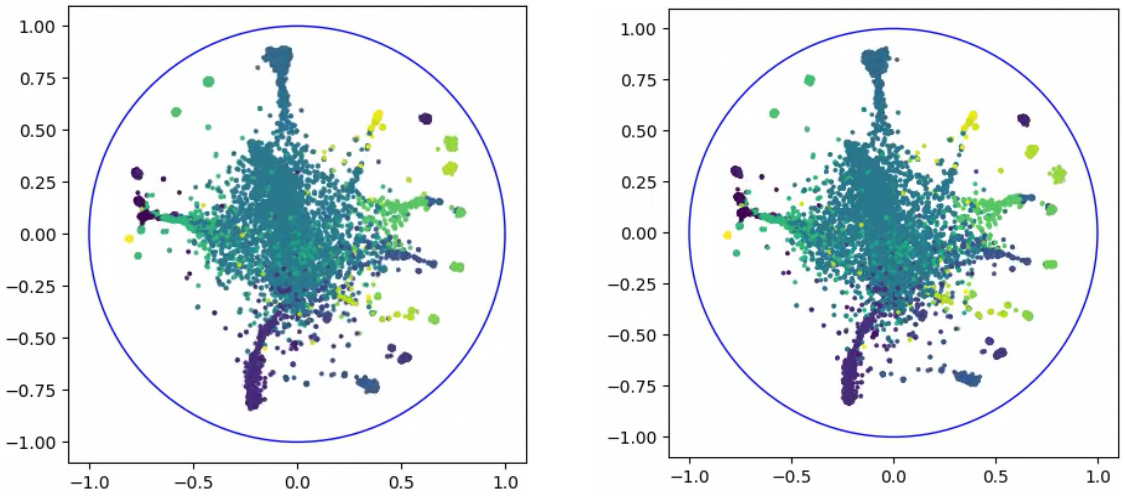
5.5. Results on different datasets

The most important thing of course is whether in practice we can create embeddings which look good and are calculated quickly on real world large datasets. Here we run both our approximation and the exact hyperbolic t-SNE on different real world datasets to see how they perform.

Hypothesis We expect our solution to create similar looking embeddings to the exact version and do so with a significant reduction in execution time.

Method To test this we run both versions on Planaria (Plass et al., 2018) (21612 data points) and C. Elegans (Packer et al., 2019) (89701 data points).

Results The experiment resulted in the following graphs



(a) Exact (b) Approximate

Figure 5.13: Planaria embedding created with hyperbolic t-SNE

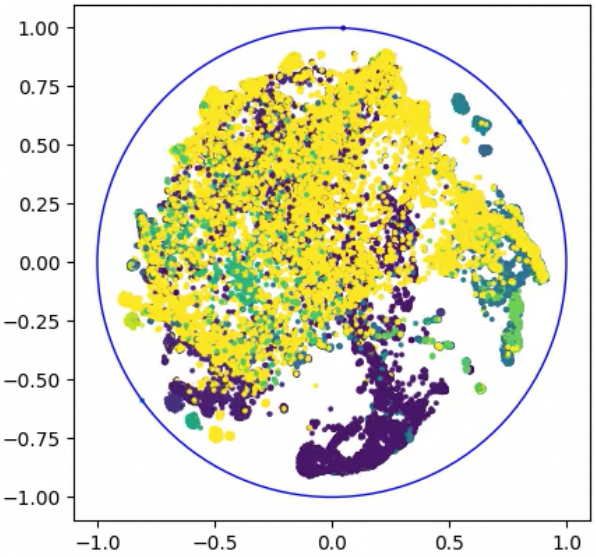


Figure 5.14: C. Elegans embedding using the approximate version

Discussion The Planaria embeddings, Figure 5.13 (a) and (b), look very similar, but the approximation took only six minutes compared to the 50 minutes of the exact version. In the case of the C. Elegans dataset our solution took 24 minutes, Figure 5.14, while the exact version based on the timing of the initial few iterations was projected to take 14 hours, and as such for now not ran to completion. Notably these datasets were also ran in (Klimovskaia et al., 2020) were Planaria would take 15 minutes and a random 40000 subsample of the C. Elegans dataset, not the full 89701 data points, would take 2-3 hours based on their estimates.

6

Conclusion

In conclusion we presented a technique to speed up the creation of hyperbolic embeddings in the Poincaré disk using t-SNE. With this approach we are able to create embeddings for datasets which are out of reach in the exact case. This allows us to embed data from large datasets such as C. Elegans (Packer et al., 2019) in minutes compared to hours in the case of the exact embedding, showing growth factors for time complexity similar to those in the Euclidean case. Even though we calculate an approximation, the embeddings are similar in quantitative and qualitative aspects.

In the future, it might be interesting to see if integration of the Fréchet mean from (Lou et al., 2020) to the approximation algorithm would be possible without increasing time complexity. Applying the Lorentz model could also be interesting for a more stable embedding calculation as numbers in the Poincaré disk model can sometimes explode leading to more sensitivity towards hyperparameters. Next to this there is also a variant of the Barnes-Hut technique, the Dual tree algorithm (Gray & Moore, 2000), which checks not for every point every cell during summarization but does cell to cell. This could lead to performance improvements but in (van der Maaten, 2014) these seemed to be negligible. Our solution for splitting is still probably not the perfect balance between a nice tree spread and the aspect ratio of the cells. Some ways that might improve these are when splitting to greedily optimize for one of the desired properties, for instance minimizing the maximum distance in a cell.

A

Derivation

We first define some helper notation with $r_{ij} = d_{ij}^2 = acosh(1 + 2\frac{||y_i - y_j||^2}{(1-||y_i||^2)(1-||y_j||^2)})^2$ and $Z = \sum_{k \neq l}(1 + d_{kl}^2)^{-1}$.

We now define the cost function and its related equations

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \tag{A.1}$$

$$q_{ij} = q_{ji} = \frac{(1 + d_{ij}^2)^{-1}}{\sum_{k,l \neq k}(1 + d_{kl}^2)^{-1}} \tag{A.2}$$

$$\begin{aligned} C = KL(P||Q) &= \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \\ &= \sum_i \sum_j p_{ij} \log p_{ij} - p_{ij} \log q_{ij} \end{aligned} \tag{A.3}$$

.

Now we can derive the gradient of this function. As changing y_i only impacts d_{ij} and d_{ji} we can therefore by the chain rule:

$$\frac{\partial C}{\partial y_i} = \sum_j (\frac{\partial C}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial y_i} + \frac{\partial C}{\partial r_{ji}} \frac{\partial r_{ji}}{\partial y_i}) \tag{A.4}$$

$$\begin{aligned} \frac{\partial C}{\partial r_{ij}} &= -\sum_{k \neq l} p_{kl} \frac{\partial(\log q_{kl})}{\partial r_{ij}} \\ &= -\sum_{k \neq l} p_{kl} \frac{\partial(\log q_{kl} Z - \log Z)}{\partial r_{ij}} \\ &= -\sum_{k \neq l} p_{kl} (\frac{1}{q_{kl} Z} \frac{\partial((1 + r_{kl})^{-1})}{\partial r_{ij}} - \frac{1}{Z} \frac{\partial Z}{\partial r_{ij}}) \end{aligned} \tag{A.5}$$

The gradient $\frac{\partial((1+r_{kl})^{-1})}{\partial r_{ij}}$ is only nonzero when $k = i$ and $l = j$, thus:

$$\frac{\partial C}{\partial r_{ij}} = 2\frac{p_{ij}}{q_{ij} Z}(1 + r_{ij})^{-2} - 2\sum_{k \neq l} p_{kl} \frac{(1 + r_{ij})^{-2}}{Z} \tag{A.6}$$

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

$$\begin{aligned} \frac{\partial C}{\partial r_{ij}} &= 2p_{ij}(1 + r_{ij})^{-1} - 2q_{ij}(1 + r_{ij})^{-1} \\ &= 2(p_{ij} - q_{ij})(1 + r_{ij})^{-1} \end{aligned} \tag{A.7}$$

Substituting into equation A.4:

$$\frac{\partial C}{\partial y_i} = 2 \sum_j (p_{ij} - q_{ij})(1 + r_{ij})^{-1} (\frac{\partial r_{ij}}{\partial y_i} + \frac{\partial r_{ji}}{\partial y_i}) \tag{A.8}$$

Since the distances in both directions are minimized by the same change to y_i they can be combined finally resulting in

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(1 + d_{ij}^2)^{-1} \frac{\partial d_{ij}}{\partial y_i} \tag{A.9}$$

.

Bibliography

Barnes, J., & Hut, P. (1986). A hierarchical o (n log n) force-calculation algorithm. nature, 324(6096), 446–449.

Chebotarev, P., & Shamis, E. (2006). The matrix-forest theorem and measuring relations in small social groups. arXiv preprint math/0602070.

Conway, J. H., Burgiel, H., & Goodman-Strauss, C. (2016). The symmetries of things. CRC Press.

Fréchet, M. (1948). Les éléments aléatoires de nature quelconque dans un espace distancié. Annales de l'institut Henri Poincaré, 10(4), 215–310.

Ganea, O., Bécigneul, G., & Hofmann, T. (2018). Hyperbolic neural networks. Advances in neural information processing systems, 31.

Gray, A., & Moore, A. (2000). N-body'problems in statistical learning. Advances in neural information processing systems, 13.

Gulcehre, C., Denil, M., Malinowski, M., Razavi, A., Pascanu, R., Hermann, K. M., Battaglia, P., Bapst, V., Raposo, D., Santoro, A., et al. (2018). Hyperbolic attention networks. arXiv preprint arXiv:1805.09786.

Guo, Y., Guo, H., & Yu, S. X. (2022). Co-sne: Dimensionality reduction and visualization for hyperbolic data. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 21–30.

Haensch, A. (2021). Reflections on hyperbolic space.

Haghverdi, L., Buettner, F., & Theis, F. J. (2015). Diffusion maps for high-dimensional single-cell analysis of differentiation data. Bioinformatics, 31(18), 2989–2998.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. Journal of educational psychology, 24(6), 417.

Ito, T. (2007). Hyperbolic non-euclidean world. Retrieved September 2022, from http://web1.kcn.jp/hp28ah77/us3_poinc.htm

Klimovskaia, A., Lopez-Paz, D., Bottou, L., & Nickel, M. (2020). Poincaré maps for analyzing complex hierarchies in single-cell data. Nature communications, 11(1), 1–9.

Kobak, D., & Berens, P. (2019). The art of using t-sne for single-cell transcriptomics. Nature communications, 10(1), 1–14.

Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A., & Boguná, M. (2010). Hyperbolic geometry of complex networks. Physical Review E, 82(3), 036106.

LeCun, Y. (1998). The mnist database of handwritten digits. http://yann. lecun. com/exdb/mnist/.

Lee, J. A., & Verleysen, M. (2010). Scale-independent quality criteria for dimensionality reduction. Pattern Recognition Letters, 31(14), 2248–2257.

Linderman, G. C., Rachh, M., Hoskins, J. G., Steinerberger, S., & Kluger, Y. (2019). Fast interpolation-based t-sne for improved visualization of single-cell rna-seq data. Nature methods, 16(3), 243–245.

Looz, M. v., Meyerhenke, H., & Prutkin, R. (2015). Generating random hyperbolic graphs in subquadratic time. International Symposium on Algorithms and Computation, 467–478.

Lou, A., Katsman, I., Jiang, Q., Belongie, S., Lim, S.-N., & De Sa, C. (2020). Differentiating through the fréchet mean. International Conference on Machine Learning, 6393–6403.

McInnes, L., & Healy, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. ArXiv e-prints.

Nickel, M., & Kiela, D. (2017). Poincaré embeddings for learning hierarchical representations. Advances in neural information processing systems, 30.

Packer, J., Zhu, Q., Huynh, C., Sivaramakrishnan, P., Preston, E., Dueck, H., Stefanik, D., Tan, K., Trapnell, C., Kim, J., et al. (2019). A lineage-resolved molecular atlas of. Science, 365.

Pezzotti, N., Thijssen, J., Mordvintsev, A., Höllt, T., Van Lew, B., Lelieveldt, B. P., Eisemann, E., & Vilanova, A. (2019). Gpgpu linear complexity t-sne optimization. IEEE transactions on visualization and computer graphics, 26(1), 1172–1181.

Plass, M., Solana, J., Wolf, F. A., Ayoub, S., Misios, A., Glažar, P., Obermayer, B., Theis, F. J., Kocks, C., & Rajewsky, N. (2018). Cell type atlas and lineage tree of a whole complex animal by single-cell transcriptomics. Science, 360(6391), eaaq1723.

Sala, F., De Sa, C., Gu, A., & Ré, C. (2018). Representation tradeoffs for hyperbolic embeddings. International conference on machine learning, 4460–4469.

Tanay, A., & Regev, A. (2017). Scaling single-cell genomics from phenomenology to mechanism. Nature, 541(7637), 331–338.

Taxel, P. (2018). Order-7 triangular tiling. Retrieved September 2022, from https://commons.wikimedia.org/wiki/File:Order-7_triangular_tiling.svg

Tom Ventimiglia, K. W. (2011). The barnes-hut algorithm. Retrieved September 2022, from http://arborjs.org/docs/barnes-hut

Tripuraneni, N., Flammarion, N., Bach, F., & Jordan, M. I. (2018). Averaging stochastic gradient descent on riemannian manifolds. Conference On Learning Theory, 650–687.

Ungar, A. A. (2008). A gyrovector space approach to hyperbolic geometry. Synthesis Lectures on Mathematics and Statistics, 1(1), 1–194.

van der Maaten, L. (2014). Accelerating t-sne using tree-based algorithms. The Journal of Machine Learning Research, 15(1), 3221–3245.

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. Journal of machine learning research, 9(11).

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

Yisheng Li, L. L. (2015). Stat 545: Glm and categorical data analysis. Retrieved September 2022, from http://yinsenm.github.io/stat545/2015/08/26/Welcome/

Zhou, Y., & Sharpee, T. O. (2018). Using global t-sne to preserve inter-cluster data structure. bioRxiv, 331611.

Zhou, Y., & Sharpee, T. O. (2021). Hyperbolic geometry of gene expression. Iscience, 24(3), 102225.