A hierarchically growing hyperbolic self-organizing map for rapid structuring of large data sets

Jörg Ontrup and **Helge Ritter**
Neuroinformatics Group
Faculty of Technology
Bielefeld University
Germany
jontrup@techfak.uni-bielefeld.de

Abstract - *We introduce the Hierarchically Growing Hyperbolic Self-Organizing Map (H^2SOM) featuring two extensions of the HSOM (hyperbolic SOM): (i) a hierarchically growing variant that allows for incremental training with an automated adaptation of lattice size to achieve a prescribed quantization error and (ii) an approximate best match search that utilizes the special structure of the hyperbolic lattice to achieve a tremendous speed-up for large map sizes. Using the MNIST database as a benchmark dataset, we show that the H^2SOM yields a highly efficient visualization algorithm that combines the virtues of the SOM with extremely rapid training and low quantization & classification errors.*

Key words - Hyperbolic Self-organizing maps; Growing network; Hierarchical Clustering; Exploratory Data Analysis

1 Introduction

The rapid employment of technological advances has led to a continuously growing volume of large data sets. At the same time, the meaning of "large" is perpetually under revision. The Self-Organizing Maps as introduced by Kohonen [3] have become a standard tool for the exploratory analysis of such data and have been extensively used for visualization purposes. A central parameter affecting the resolution of the SOM is the area of its map size. With a linearly increasing map area, the number of nodes in a SOM increase quadratically. Therefore, the training of large maps can be computationally quite expensive. Several approaches have been suggested to overcome this problem. Koikkalainen and Oja [4] proposed the Tree-Structured Self-Organizing Map (TS-SOM), which consists of a fixed number of SOMs arranged in a pyramidal structure. The training of the pyramid is computed level-wise where the best match search is performed as a tree search reducing the complexity to $\mathcal{O}(log\ N)$. A Growing Hierarchical SOM has been proposed by Rauber et al. [11]. Their approach combines individually growing SOMs with a hierarchical architecture and has successfully been applied to the organization of document collections and music repositories. Lately, Pakkanen et al. [10] have described the Evolving Tree, which is constructed as a freely growing network utilizing the shortest path between two nodes in a tree as the neighborhood function for the self-organizing process. All of these approaches achieve a favorable computational complexity. However, the visualization of the learned hierarchies remains a demanding task. Either a map metaphor is not applicable, or the transition between maps within or across the hierarchies introduces discontinuities making it hard to visualize and maintain the surrounding context. Thus, without guidance the user might be easily lost within the tree structure. Lamping and Rao [5] discovered that hyperbolic space is ideally suited to embed large hierarchical structures. Their discovery motivated the introduction of the hyperbolic SOM (HSOM) [12]. In this contribution we show that the HSOM can be naturally extended to a *Hierarchically Growing Hyperbolic SOM* (H^2SOM) that combines the virtues of hierarchical data organization, adaptive growing to a required granularity, good scaling behaviour and smooth, map-based browsing, thereby combining several strengths of separate, previous approaches with a single, uniform architecture.

2 Hyperbolic Geometry

Most of our spatiotemporal thinking is deeply rooted in the world of Euclidean geometry following Euclid's five axioms. However, hyperbolic space [1] offers a completely consistent non-Euclidean geometry that is characterized by being negatively "curved". Standard textbooks on Riemannian geometry, e.g. [1, 6] show that the relationships for the area A and circumference C for a circle of radius r are then given by $A(r) = 4\pi \sinh^2(r/2)$ and $C(r) = 2\pi \sinh(r)$, respectively. This bears two remarkable asymptotic properties: *(i)* for small radius r the space "looks flat" since $A(r) \approx \pi r^2$ and $C(r) \approx 2\pi r$. *(ii)* For larger r both A and C grow asymptotically *exponentially* with the radius.

Naturally, there exists no isometric embedding of $\mathbb{H}^2$ into $\mathbb{R}^2$, since a projection of the negatively curved space into flat space introduces distortions in either length, area or angle. However, a locally isometric embedding into $\mathbb{R}^3$ is possible: we obtain a "wrinkled" structure, which resembles a saddle at every point of the surface. Sometimes, Nature approximated the growth behaviour of a hyperbolic surface, e.g. in some corals that need to maximize their contact area with the surrounding water that carries vital nutrients. In Figure 1 it can be seen, that this is leading to structures resembling a 3-dimensional local embedding (of a patch) of the hyperbolic plane remarkably well.

Figure 1: A local embedding of $\mathbb{H}^2$ in $\mathbb{R}^3$ would look very similar to such a leather-coral for which nature found a solution to maximize its contact area in order to absorb vital nutrients from the surrounding water.

The geometric properties discussed above make the hyperbolic space an ideal candidate for embedding large hierarchical structures [5, 7]. For its display on a flat 2D screen one may choose the projection of $\mathbb{H}^2$ on the Poincaré Disk that has a number of convenient beneficial properties for visualization: First, it is locally shape preserving, with a strong "fish-eye" effect: The origin of $\mathbb{H}^2$ - corresponding to the "fish-eye" fovea - is mapped almost faithfully, while distant regions become exponentially "squeezed". Second, the model allows to translate the original $\mathbb{H}^2$ in a very elegant way. Thus, the fovea can be moved to any other part of the infinite hyperbolic plane. This enables the user to selectively focus on interesting portions of a map painted on $I\!H^2$ while still keeping a coarser view of its surrounding context. For further details on the construction of the Poincaré Disk and the Möbius transformations to translate the fovea, see e.g. [12, 8, 13].

3 Hierarchically Growing Hyperbolic Maps

3.1 Network Architecture

The core idea of the hierarchically growing Hyperbolic Self-Organizing Map is to employ the same sort of grid already underlying the plain HSOM and its applications [12, 8, 13]:

1. **Initialization:** We start with a ring of n_b equilateral hyperbolic triangles centered at the origin of $I\!H^2$ as shown in Figure 2(a). The $n_b + 1$ vertices of the triangles form the network nodes of the first level of the hierarchy.
2. **Growth Step:** We can expand each node in the periphery of the existing network by surrounding it with the vertices of additional $n_b - 2$ equilateral triangles (forming a regular hyperbolic "n_b-gon" around the selected node).

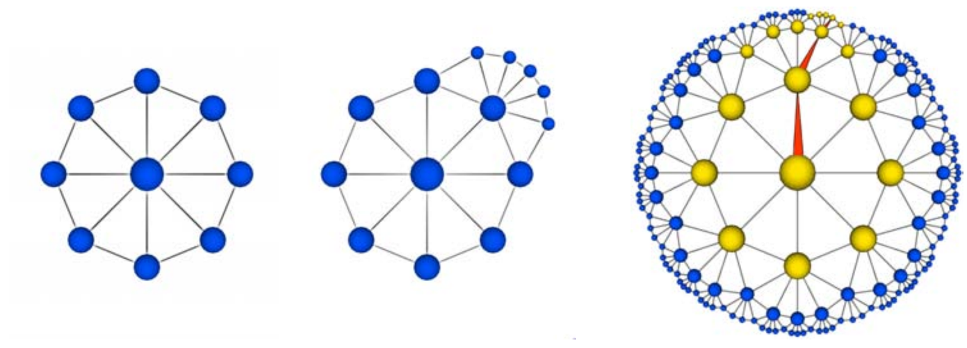
(a) (b) (c)

Figure 2: (a) The nodes at the vertices of - in this case $n_b = 8$ - equilateral triangles form the first level in the hierarchy of the H^{2}SOM. (b) When a node meets a growth criterion, we generate a set of $n_b - 3$ children nodes. In (c) the search path and the set of nodes visited during a best match search are highlighted within a fully expanded hierarchy of 3 levels. Note, that the nodes are placed on an equidistant grid in hyperbolic space, only the projection on the 2D Poincaré Disk introduces a strong distortion of distances at the perimeter of the disk.

By repeating step 2 for all border nodes, we can generate a new "ring" level of the hierarchy. The "branching" factor n_b determines how many nodes are generated at each level and how "fast" the network is reaching out into the hyperbolic space. It has a lower bound ruled by hyperbolic geometry: The sum of the angles in a hyperbolic triangle is always less than π. Therefore, the angle α of a equilateral triangle has to obey $\alpha < \pi/3$. Since we are covering a full circle with our triangles, c.f. Figure 2(a), also $\alpha = 2\pi/n_b$ holds, where n_b is the number of nodes placed on the full circle. When combining the two conditions we see that we need a branching factor of $n_b > 6$ for the tessellation scheme. Consequently, since each node in the tessellation has one parent and two sibling nodes, it has to have at least four children nodes. Note, that there exists no upper bound for the number of childrens a node can have.

3.2 Learning and Growing Procedure

The training of the hierarchical network largely follows the traditional SOM approach. To each node a a reference vector $\mathbf{w}_a$ is attached, projecting into the input data space X. In addition, it will be convenient to attach to each node also its 2D position $\mathbf{z}_a \in \mathbb{C}$ in the complex Poincaré Disk $|\mathbf{z}| \leq 1$ [1]. The center and the first n_b nodes are initialized with the mass of center of the training data and small variations from that, respectively. This initial configuration is then trained in the usual way: After finding the best match neuron a^*, i.e. the node which has its prototype vector $\mathbf{w}_a$ closest to the given input $\mathbf{x}$, $a^* = \text{argmin}_a \|\mathbf{w}_a - \mathbf{x}\|$ all reference vectors are updated by the well known adaptation rule

$$\Delta \mathbf{w}_a = \epsilon(t)\ h(a, a^*)\ (\mathbf{x} - \mathbf{w}_a), \quad \text{with} \quad h(a, a^*) = \exp\left(-\frac{d^2_{a,a^*}}{\sigma(t)^2}\right) \tag{1}$$

Here $h(a, a^*)$ is a bell shaped Gaussian centered at the winner a^* and decaying with increasing distance d_{a,a^*} of the neurons. We can then compute the hyperbolic node distances d_{a,a^*} conveniently from their associated positions $\mathbf{z}_a$ in the Poincaré Disk:

$$d_{a,a^*} = 2\ \text{arctanh}\left(\frac{|\mathbf{z}_a - \mathbf{z}_{a^*}|}{|1 - \mathbf{z}_a \bar{\mathbf{z}}_{a^*}|}\right). \tag{2}$$

During the course of learning, the width $\sigma(t)$ of the neighborhood bell function and the learning step size $\epsilon(t)$ are continuously decreased in order to allow more and more specialization and fine tuning of the then increasingly weakly coupled neurons - just as in the standard SOM approach.
After fixed training intervals we repeatedly evaluate for each node an expansion criterion. In our experiments we have so far used the node's quantization error as the growth criterion. If a given threshold Θ_{QE} for a node is exceeded, that node is expanded as described in step 2 above. After the expansion step where all nodes meeting the growth criterion were expanded, all reference vectors from the previous hierarchies become fixed and adaptation "moves" to the nodes of the new structural level.

3.3 Fast Best Match Search

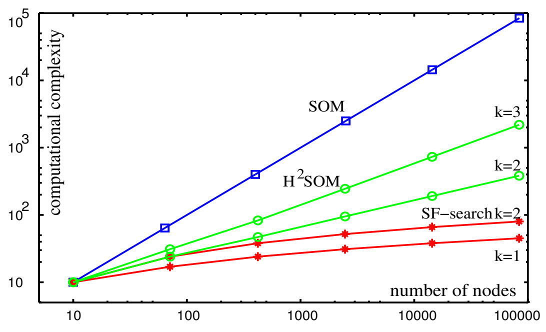

Figure 3: Scaling behaviour for $n_b = 10$.

The peculiar, intrinsically "uniformly hierarchical" structure of the hyperbolic grid offers an intriguing possibility to significantly accelerate the most time-consuming step in a SOM: we can approximate the global search for the winner unit a^* by *a fast tree search*, taking as the search root the initial center node of the growth process and following then the "natural" hierarchical structure in the hyperbolic grid: starting from this node, we recursively determine the k best-matching nodes among its n_b neighbors until we reach the periphery. For $k = 1$, this will generate a path with $\mathcal{O}(\log_{n_b} N)$ comparisons, instead of $\mathcal{O}(N)$ for a global search. For $1 < k \leq n_b$ we asymptotically must search $\mathcal{O}(N^p)$ nodes, with exponent $p = \log_{n_b} k \leq 1$

(restituting a full search with $p = 1$ for $k = n_b$). Fig. 3 shows, for $n_b = 10$, that the resulting scaling behaviour permits speed-ups of several orders of magnitude, as compared with a global (standard SOM) search.

Both, the geometry of the hyperbolic lattice, together with the hierarchical growth scheme, tend to organize the prototype vectors $\mathbf{w}_a$ in such a manner that the above search scheme provides a very good approximation to global search. In fact, our experiments indicate that we may even truncate the tree branching factor to $k = 1$ for all search steps beyond the innermost ring, leading to a "super-fast" search scheme ("SF-search") scaling as $\mathcal{O}(k \cdot \log_{n_b} N)$ (lower curves in Fig. 3). For instance, in the test problem reported below we found that for $k = 2$ ($k = 1$) SF-search led to the correct best match unit or the very vicinity of it in 92% (65%) of all cases, leading to maps that were on par with or even significantly outperformed euclidian SOMs constructed with global search.

3.4 Visualization of Hierarchical Hyperbolic Maps

The distinctive difference of the H^{2}SOM to other hierarchical SOM variants such as the Tree-Structured SOM (TS-SOM) [4], the Hierarchical SOM [11], the Self-Organizing Tree Algorithm (SOTA) [2] or the Evolving Tree by Pakkanen [9] is that the complete hierarchy is embedded within a continuous, browsable space. When selecting a deeper level within the hierarchy the user does not need to carry out a discrete "jump", where the surrounding context might be lost, but instead can traverse the complete hierarchy in a smooth way. We believe that this is a very important property for visualization and have developed a framework using the open source visualization library VTK[1] to display a 3D scene where the user can interact with the Poincaré Disk in two ways: *(i)* The disk can be "grabbed" with the mouse and freely moved in 3D space, such that a suitable viewpoint might be chosen. *(ii)* With a drag operation, the focus of the map can be moved continuously in hyperbolic space, such that a certain level of detail can be selected in a "focus & context"-like manner. Depending on the underlying data set, graphical attributes such as glyph type, color, size or even texture can be used to map selected attributes of the prototype vectors on the visualized nodes. In the next section we give an example how we can move along a learned hierarchy from a coarse overview into finer grained details of the data.

4 Application to the MNIST database

In order to benchmark the H^{2}SOM we have chosen the MNIST database[2] of handwritten digits as an example data set that features a large collection of rather high-dimensional patterns. It consists of 60.000 training samples from approximately 250 writers and 10.000 test samples from a disjoint set of 250 other writers. We used the original 784-dimensional dataset which resembles 28x28 pixel grey level images of the handwritten digits. Since we used the scalar product as our data metric, all samples were normalized to length one.

[1]`http://public.kitware.com/VTK/`
[2]`http://yann.lecun.com/exdb/mnist/`

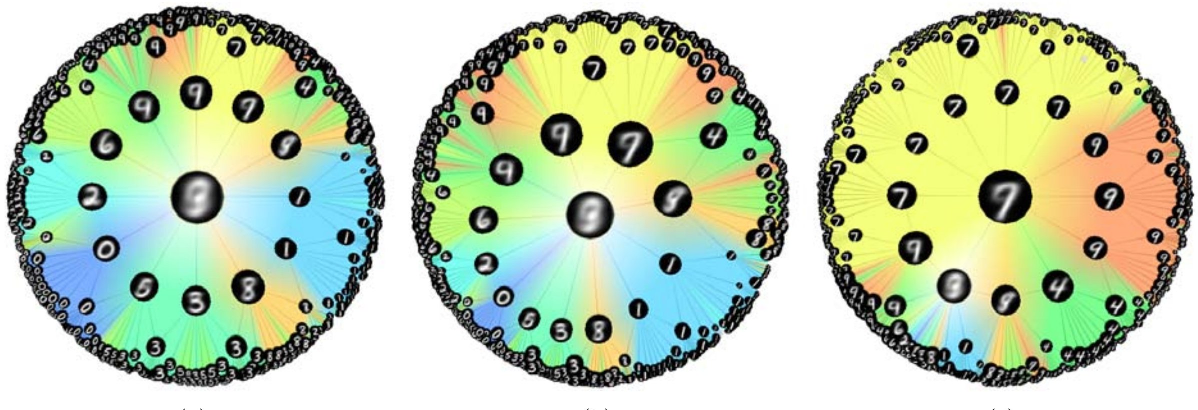
(a) (b) (c)

Figure 4: An example navigation along the learned structure of the MNIST database. The neuron's prototype vectors are visualized as textures overlayed on the node's glyphs. In (a) the overall coarse structure of the dataset is shown. In (b) and (c) the user moved the focus to the "7", "9", "4" region of the map, where the next structural level in the data can be observed.

4.1 Structuring of the data

In Figure 4(a) a H^2SOM with a branching factor of $n_b = 12$ is shown in a centered view, such that the top-level structure of the dataset is visible as the innermost ring of nodes. The prototype vectors are overlayed as textures on the node's glyphs. The colors on the map are just a visual hint to indicate the class to which the majority of training samples belong to in the corresponding region of the map. The H^2SOM can be seen to have learned the following top level structure from the data: The upper three nodes resemble mixtures between "4"s, "9"s and "7"s. Clockwise follows a node with a prototype looking like a blurred slanted "9", then two different orientated "1"s follow. At the bottom, three prototypes similar to an "8", "3" and "5" are shown, and then an articulated "0", "2" and "6" appear. In Figure 4(b) the user is moving the focus towards the one o'clock node which is then centered in Figure 4(c). Here it can be seen, that at this next structural level the data splits up into equally slanted "7"s at the top, "9"s to the right and "4"s at the bottom of the map.

4.2 Comparison to standard SOM

In order to assess the quality of the H^2SOM, we have conducted several numerical experiments and compared the data to the classical SOM. We have trained two standard SOMs of the sizes of 13x13 (169 nodes) and 48x48 (2304 nodes) and two similar sized H^2SOMs with 161 and 2461 neurons, respectively. In all cases 600.000 training steps were performed. The results averaged over 10 runs (the large SOM was just trained twice) are given in Table 1. The most prominent difference is the time needed for the training of the networks. The large SOM took more than 18 hours to compute, while the large H^2SOM using the "super-fast" SF-search was finalized in 13 minutes. Despite using a full search for the SOM, the mean quantization error of the maps with respect to the training data is comparable, with the H^2SOM slightly in advantage. When using the SOMs as a classification tool, we used (a) the SF-search with $k = 2$, and (b) a slower global search to find the best match nodes for the 10.000 test samples

	SOM		H^{2}SOM			
	13x13	48x48	$n_b = 8$, 3 rings		$n_b = 10$, 4 rings	
nodes	N=169	N=2304	N=161		N=2461	
QE	0.2094	0.1510	0.1993		0.1307	
t_{train}	1:07h	18:34h	0:09h		0:13h	
t_{test}	7.8s	181s	(a) 1.8s	(b) 8.4s	(a) 3.0s	(b) 110s
Class	classification performance [%]					
0	93.9	98.3	96.0	**98.1**	**98.3**	**99.1**
1	98.3	98.6	**98.3**	98.1	98.5	**98.9**
2	86.6	94.6	**89.1**	**92.4**	92.4	**94.8**
3	80.2	91.3	76.1	79.5	90.0	**92.7**
4	69.0	88.3	**73.2**	**76.3**	**90.4**	**92.2**
5	66.9	90.0	**83.5**	**89.1**	87.4	**91.6**
6	93.9	97.1	89.7	92.7	96.0	**97.5**
7	81.2	91.0	**81.7**	**85.9**	**91.4**	**92.3**
8	76.4	88.8	59.1	67.6	88.1	**92.5**
9	59.8	88.0	55.9	57.8	**88.3**	**89.8**
total	81.0	92.7	80.5	**85.3**	92.2	**94.6**

Table 1: Comparison of the hierarchically grown HSOM (H^{2}SOM) to similar sized standard SOMs. The table shows the training times in hours and minutes for the map formation of the 60000 training samples and the seconds for the best match lookups for the 10000 test samples of the MNIST database, respectively. For the H^{2}SOM the test runs were performed with (a) the rapid SF-search with $k = 2$ and (b) a slower global search. (All results were obtained on a standard laptop running a Pentium-M processor with 1.5 GHz).

and their attached labels from the training data. In the first case, the overall performance of the SOM is with 0.5% slightly better, though for several classes the H^{2}SOM achieves the same or better results. When using the slower global search only for retrieval *after the fast training* of the H^{2}SOMs, the classification performances for the latter become considerably better and now clearly outperform the SOM.

5 Conclusions

In this paper we have presented the Hierarchically Growing Hyperbolic SOM (H^{2}SOM), a new extension to the hyperbolic SOM. In contrast to other, previous approaches the H^{2}SOM manages to combine *(i)* a growing scheme allowing for an incremental training of an adaptive lattice structure to achieve a prescribed quantization error, *(ii)* a hierarchical data organization both yielding an excellent scaling behaviour and the possibility to explore the underlaying structures in data and *(iii)* last but not least a continuous, map-based browsing offering a natural “focus & context” behaviour. Experiments with the MNIST database indicate, that our proposed SF-search utilizing the intrinsically hierarchical structure of the hyperbolic grid allows to train self-organized maps with a quality on par or even better than standard euclidian SOMs but several magnitudes faster.

References

[1] H. S. M. Coxeter. *Non Euclidean Geometry.* Univ. of Toronto Press, Toronto, 1957.

[2] J. Herrero, A. Valencia, and J. Dopazo. A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics*, 17(2):126–136, 2001.

[3] T. Kohonen. *Self-Organizing Maps.* Springer Series in Information Sciences. 3rd edition, 2001.

[4] P. Koikkalainen and E. Oja. Self-organizing hierarchical feature maps. In *Proc. of the IJCNN 1990*, volume II, pages 279–285, 1990.

[5] J. Lamping and R. Rao. Laying out and visualizing large trees using a hyperbolic space. In *ACM Symposium on User Interface Software and Technology*, pages 13–14, 1994.

[6] F. Morgan. *Riemannian Geometry: A Beginner's Guide.* Jones and Bartlett Publishers, Boston, London, 1993.

[7] T. Munzner. Exploring large graphs in 3D hyperbolic space. *IEEE Computer Graphics and Applications*, 18(4):18–23, July/August 1998.

[8] J. Ontrup and H. Ritter. Text categorization and semantic browsing with self-organizing maps on non-euclidean spaces. In *Proceedings of PKDD-01, 5th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 338–349, 2001.

[9] J. Pakkanen. The Evolving Tree, a new kind of self-organizing neural network. In *Proceedings of the Workshop on Self-Organizing Maps '03*, pages 311–316, Kitakyushu, Japan, September 2003.

[10] J. Pakkanen, J. Iivarinen, and E. Oja. The evolving tree – a novel self-organizing network for data analysis. *Neural Processing Letters*, 20(3):199–211, December 2004.

[11] A. Rauber, D. Merkl, and M. Dittenbach. The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, 13(6):1331–1341, 2002.

[12] H. Ritter. Self-organizing maps in non-euclidian spaces. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 97–110. Amer Elsevier, 1999.

[13] J. Walter, J. Ontrup, D. Wessling, and H. Ritter. Interactive visualization and navigation in large data collections using the hyperbolic space. In *Proceedings of the Third IEEE International Conference on Data Mining*, Nov. 2003.