Hyperbolic Geometry for Hierarchical Data Visualization and Interpretable ML

Matthew Danbury

November 1, 2021

We began our investigation into hyperbolic geometry by considering how circumferences and areas in the Poincaré disk model vary as a function of radius, first by counting hyperbolic triangles and making conjectures (Homework 3), then by doing a more systematic calculation using the scale factor $\frac{2}{1-r^2}$. The result we found was that the hyperbolic area of a circle whose radius has hyperbolic length r is

$$4\pi \sinh^2\left(\frac{r}{2}\right) = \pi(e^r + e^{-r}) - \frac{\pi}{2}$$

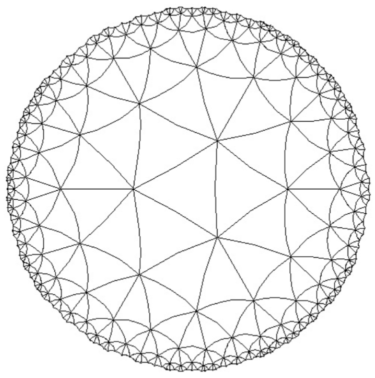
Figure 1: A triangular tiling of the Poincaré disk model of hyperbolic space.

Since e^{-r} is small compared to e^r for large r, it is reasonable to say that hyperbolic areas scales exponentially with hyperbolic radii. Most applications of hyperbolic geometry in data science use this fact as a jumping off point. Seeing an "area" that scales exponentially with its "radius" is a cue invoke hyperbolic geometry.

A cute demonstration of this strategy comes from drawing hierarchical data structures known as “trees”. As a primary school student, you might have had to sketch out your family tree on construction paper, an evolutionary tree for a biology assignment, or a probability tree for the likelihoods of outcomes for a sequence of tasks. If you did, you will remember the sinking feeling you get around layer three when you realize that you are quickly running out of room. You would inevitably realize that the only way to salvage your project is to make your handwriting smaller and smaller, squishing your nodes closer and closer together, and shrinking the edges that connect them. Though you might not have realized it at the time, by doing this you were performing an rudimentary embedding of your tree into hyperbolic space! Indeed, if you consider the number of nodes in your tree to be an "area" - a reasonable interpretation if you are going to be drawing (embedding) them on paper - and the depth of the tree (number of edges from root to leaf node) to be the associated "radius", then the total number nodes for a tree with k child nodes per parent as a function of its depth r is exponential, namely k^r.

With the advent of personal computers with ever more performant processing capabilities and pixel-dense displays, the good folks over at the Xerox Palo Alto Research Center realized that you could formalize this procedure into an interactive information display. In a 1994 paper[1], a pair of HCI (human-computer interaction) re-

[1] [1]J. Lamping and R. Rao, Laying Out and Visualizing Large Trees Using a Hyperbolic Space, Proceedings of the ACM Symposium on User Interface Software and Technology, 1994

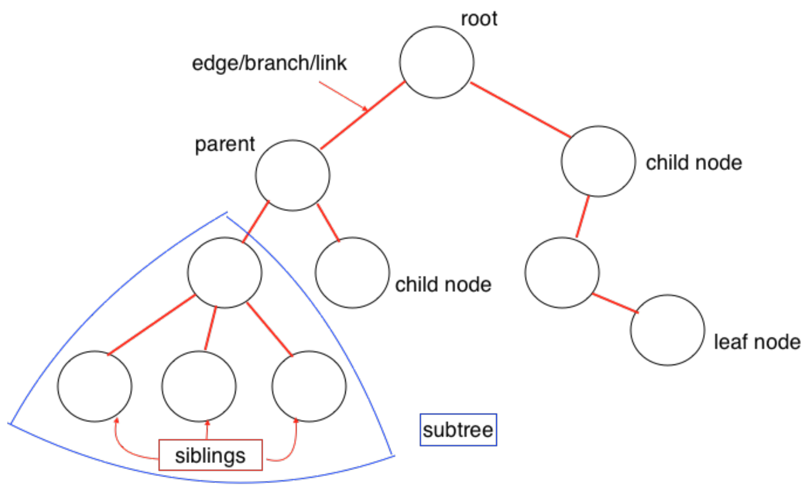

Figure 2: A diagram of a tree data structure with labels explaining the associated jargon.

searchers describe a scheme for visualizing tree data structures by embedding them in the Poincaré disc model of hyperbolic space. The

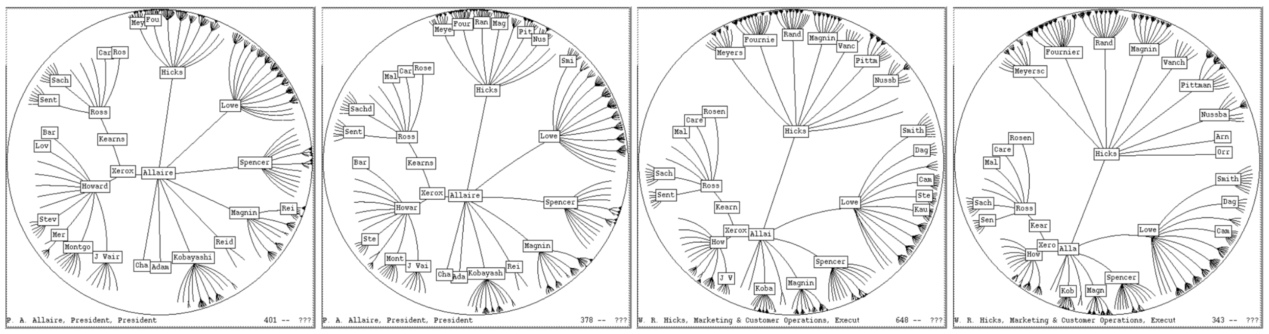
P. A. Allaire, President, President 401 -- ???
P. A. Allaire, President, President 378 -- ???
W. R. Hicks, Marketing & Customer Operations, Execut 648 -- ???
W. R. Hicks, Marketing & Customer Operations, Execut 343 -- ???

Figure 3: An example of a click-and-drag interaction to bring the node "Hicks" into the center of the Poincaré disc model, adapted from 'Laying Out and Visualizing Large Trees Using a Hyperbolic Space'. At each step, the translation isometry is followed up with a "counter-rotation" to preserve a canonical orientation between nodes.

appeal of such an information interface is that since the disc contains all of hyperbolic space, the user can see the entire tree at all times, no matter its depth or number of nodes. Moreover, every point in the Poincaré disc can be mapped to the origin with an appropriate isometry. Therefore, with a click-and-drag interaction (illustrated in Figure 3), the user can focus on any node in the tree by bringing it into the center of the disk where the scale is larger while preserving the tree structure and the contextual information it contains.

If one were to naively implement this scheme, a peculiarity arises that a student of hyperbolic geometry might anticipate, but a lay-user who is used to interacting with Euclidean click-and-drag displays might find confusing: when one translates an arbitrary point in the Poincaré disc, nearby points appear rotated compared to their original positions relative to the dragged point. Therefore, the authors suggest composing each translation with a rotation in order to preserve orientation between nodes before and after the translation. For an an interactive example of this type of tree visualization where this wasn't done, I would encourage the reader to go to https://hyperbolic-tree-of-life.github.io

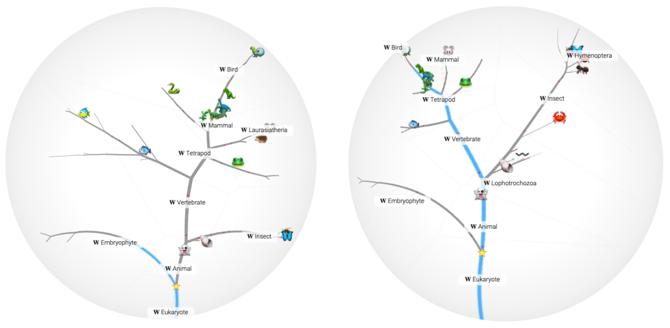

Figure 4: The tree of life visualized with the Poincaré disk model, similar to the scheme proposed in 'Laying Out and Visualizing Large Trees Using a Hyperbolic Space', but with no counter rotation. Notice how by dragging Lophotrochozoa into the center, the vertabrates track counterclockwise about the origin. Adapted from https://hyperbolic-tree-of-life.github.io

Somehow, despite the careful consideration that the authors put in, this kind of tree visualization never really caught on. Our computers' file directories are displayed much like their physical filing cabinet predecessors - as nested file folders. While student of mathematics might bemoan this missed opportunity to put hyperbolic geometry into daily use, I would argue that from a design perspective, the nested folder structure adapts better to standard use cases of file directories. This is largely because search algorithms make digging through deep file trees - whether they are visually represented by lists or hyperbolic space - largely obsolete! On your computer, you are never more than a few keystrokes (cmd + space on the Mac) away from pulling up the exact file you need, no digging required!

But all is not lost! As we saw in the tree of life example, this type of visualization excels when users are actually trying to explore the and understand different relation between nodes. One field that I could see greatly benefiting from this sort of visualization is Machine Learning. For those unfamiliar, at its essence the objective of machine learning is simply function approximation. Given a collection of potentially correlated variables (the jargon being "features" for input variables/the functions domain and "targets" for the output variables/the codomain of the function) and a family of functions of those variables, a machine learning algorithm "learns" which member of the family is the best by iteratively calculating how well a member function maps the input variables of a dataset to their outputs, and adjusting the function's parameters.

A central issue in ML is the interpretability of the resulting model. In general, these are nonlinear functions of many input variables, making them inherently difficult to visualize. One of the most interpretable classes of machine learning models are decision trees. They get their name from the fact that they assign input data output labels based on a series of decisions about the values of the features. Geometrically, we can think of this as recursively partitioning the input space into axes-aligned patches and assigning each patch a particular output value (see Figure 5). As these trees accumulate more

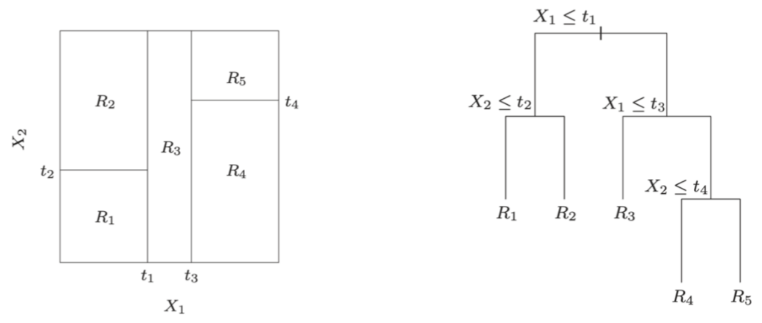

Figure 5: A decision tree of two input variables.

and more nodes, they become more difficult to visualize. For example Figure 6 shows the decision tree for a model trained to predict whether a passenger of the Titantic survived as a function of their age, sex, fare, which deck of the ship they were on, and the number of people in their party (which it achieves with 76 percent accuracy).

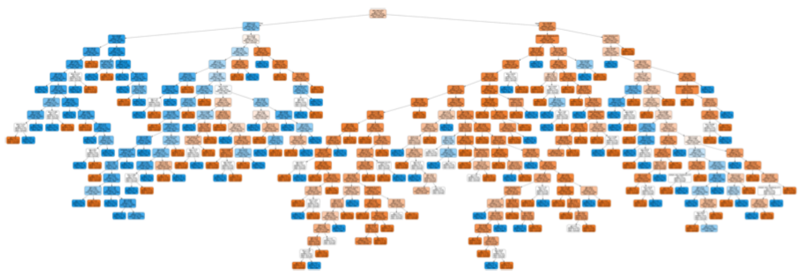

Figure 6: A decision tree which kinds of passengers survived the titantic (blue leaves) and which did not (red leaves).

Unlike other machine learning models, you have a means of visualizing the function obtained, but in its current state, it is fairly hard to explore. One could imagine applying the scheme proposed in 'Laying Out and Visualizing Large Trees Using a Hyperbolic Space', to create an interactive visualization of the decision rule. What is more, a recent thrust in interpretable ML has been to find ways of interpreting complex models by training simpler models on the predictions they make. This is known as model distillation [2] and can be used to

[2] Coincidentally, last Friday's Cornell Center for Applied Mathematics colloquium speaker Lester Mackey just so happens to be a leading expert in model distillation :)

generate "effective decision trees". Therefore is easy to imagine a implementing a general purpose workflow for interpretable ML where one trains a complex model, creates an effective decision tree, and then embeds it into the Poincaré disk model of hyperbolic space to be explored. In practice, this would mean better strategies for peeking into the "black box" of machine learning to create data products that can be explained to domain experts, audited for bias and unfairness, and more readily debugged and improved.