Link Prediction on Synthetic Graphs in Hyperbolic Space

Damir Vrabac and Evan Laksono
CS 229 Final Project, Autumn 2019

Stanford University

Graph Embedding in Hyperbolic Space

Embedding in a non-Euclidean space has been demonstrated to be a viable alternative for graph representation learning [1, 2]. Hyperbolic space is particularly known for its capacity to encode tree-like graphs with minimal distortion, as its volume increases exponentially with radius. Poincare ball is one particular manifold in which the distance between two points grows to infinity as we edge closer to the rim ($R_P = 1$):

$$d_P(\vec{x}, \vec{y}) = \text{arccosh}\left(1 + \frac{2||\vec{x} - \vec{y}||^2}{(1 - ||\vec{x}||^2)(1 - ||\vec{y}||^2)}\right) \tag{1}$$

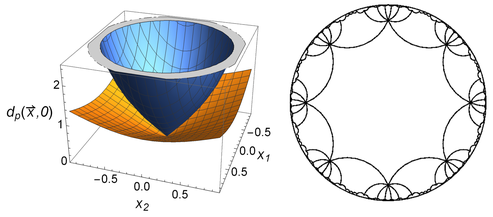

Fig. 1: (Left) The distance of a point $\vec{x}$ from the origin in a Poincaré disk (blue) and Euclidean space (orange). (Right) Tree embedding in a Poincare disk.

Scheme to Generate Synthetic Graphs

- Our scheme is similar to the approach in Ref. [3], which reflects some properties of real world data, e.g. scale-free topology and strong clustering of social networks.
- We used synthetic random graphs as our learning inputs. N nodes were distributed inside a unit circle according to the following probability density function:

$$f(r) = \frac{\alpha \sinh \alpha r}{\cosh \alpha R - 1}, \quad 0 \leq r \leq R, \quad \alpha > 0, \tag{2}$$

  with the angle uniformly distributed.
- The distance function was defined either in an Euclidean space or a Poincare ball with $R_P = 1$.
- An edge between a pair of nodes (i, j) was generated with the probability $p((i,j) \in \varepsilon) = [\exp(\eta(d(i,j) - r_0))]^{-1}$. For $\eta \gg 1$, this effectively means that only nodes separated by $d(i,j) \leq r_0$ are connected to each other.
- The node features were defined by their positions, which we encoded in n_r, n_θ uniform bins in radial and angular directions respectively. In our current experiment, we set $n_r = n_\theta = 20$.

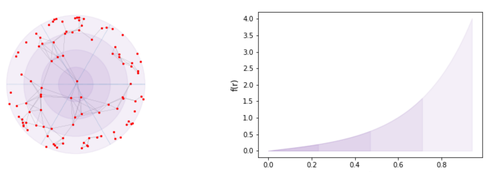

Fig. 2: (Left) A random graph with $N = 100$, $\alpha = 4$, $R_0 = 0.95$, $n_\theta = 6$, and $n_r = 4$. (Right) The corresponding probability distribution function in the radial direction, which is divided into bins of equal spans.

Hyperbolic Graph Convolutional Network (HGCN) Architecture

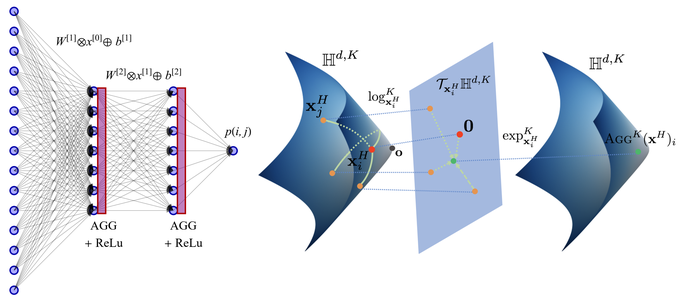

Fig. 3: (Left) HGCN architecture schematics. The number of node features define the number of input nodes while the embedding dimension determines the number of nodes in the hidden layers. In the link prediction task, we calculate the probability of edges between (i, j)-node pairs. (Right) The aggregation operation is performed upon projection to Euclidean space.

- The operations in each HGCN layer ℓ are [1]:

$$\mathbf{h}_i^{\ell,H} = \left(W^\ell \otimes^{K_{\ell-1}} \mathbf{x}_i^{\ell-1,H}\right) \oplus^{K_{\ell-1}} \mathbf{b}^\ell \tag{3}$$

$$\mathbf{y}_i^{\ell,H} = \text{AGG}^{K_{\ell-1}}\left(\mathbf{h}^{\ell,H}\right)_i \tag{4}$$

$$\mathbf{x}_i^{\ell,H} = \sigma^{\otimes^{K_{\ell-1},K_\ell}}\left(\mathbf{y}_i^{\ell,H}\right) \tag{5}$$

  All mathematical operations, e.g. addition $\oplus$, multiplication $\otimes$ and ReLu activation σ, are defined on the Poincare ball. The curvature of the embedding space for each layer $\{K_\ell\}$ was also learned in our experiment.
- In HGCN, the synthetic graphs were embedded on a D-dimensional Poincare ball. The neighborhood information of each node was encoded through aggregation operation.
- HGCN was used for link prediction task. The probability for edges $p(i,j)$ was defined using the Fermi-Dirac decoder:

$$p\left((i,j) \in \mathcal{E} | \mathbf{x}_i^{L,H}, \mathbf{x}_j^{L,H}\right) = \left[\exp\left(\left(d_{\mathcal{L}}^{K_L}\left(\mathbf{x}_i^{L,H}, \mathbf{x}_j^{L,H}\right)^2 - r\right)/t\right) + 1\right]^{-1} \tag{6}$$

  where distance is also defined in a Poincare ball. We used binary cross entropy with respect to $p(i,j)$ as our loss function. The hyperparameters r and t were assumed to be constant throughout our experiment. We used the following hyperparameters: $r = 2$ and $t = 1$.

Baseline Model and Experiments

- Our baseline model is Graph Convolution Network (GCN) which operates in the Euclidean space [1].
- In addition to our edge data, our full dataset also consisted of equal number of false edges, i.e. node pairs which are not connected to each other. For our training, we masked 15% of the positive and false edges separately. Our training/dev/test split was given by 85/5/10.
- We scanned the hyperparameter space and chose the best combinations according to the ROC value for a particular embedding dimension D. In particular, we looped through different dropout rates and learning rates.

Results and Discussions

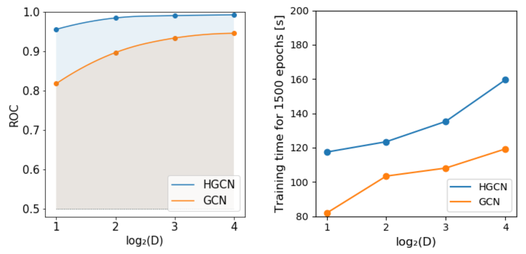

Fig. 4: HGCN and GCN were trained to predict links on randomly generated graphs with $N = 1000$, $\alpha = 10$, $R = 0.95$, $r_0 = 1$, and $\eta = 100$. In this scenario, the distance between nodes is defined on a Poincare disk. (Left) GCN and HGCN performances (ROC values) are plotted for various embedding dimensions D. (Right) The average timing to train for 1500 epochs for each network.

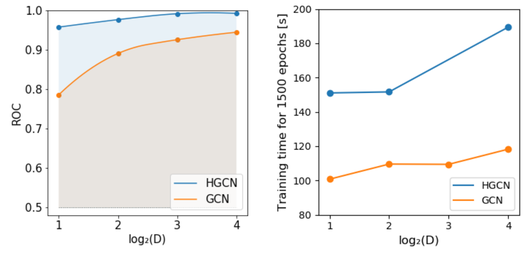

Fig. 5: HGCN and GCN were trained to predict links on randomly generated graphs with $N = 1000$, $\alpha = 0.1$, $R = 0.2$, $r_0 = 1, and \eta = 100$. Here, we used Euclidean distance to determine the edge connections between nodes. We presented similar plots as Fig. 4.

- The quality of the model might change dramatically for different combinations of hyperparameters.
- HGCN boosts the performance of link prediction task compared to GCN on graphs with both underlying Poincare Ball and Euclidean geometries.
- The accuracy of link prediction is largely determined by how close the embedding captures the actual graph structure with its underlying geometry. Our result seems to suggest a greater capacity of HGCN to capture such a structure.
- Training HGCN is slower than training GCN for a fixed D. HGCN is also less stable than GCN as HGCN is more sensitive to hyper-parameter tuning in order to achieve convergence.

Conclusions and Future Directions

We have shown a class of synthetic graphs on which HGCN is superior to GCN in performing link prediction task. Our future work aims to characterize a set of graphs in which one model can be expected to work better than the other model, and how our current synthetic construction relates to a real data set. We also expect to extrapolate the HGCN scheme to producet spaces.

Acknowledgements

We would like to thank Fred Sala, Ines Chami, and Adva Wolf for providing valuable inputs towards the completion of this project.

References

[1] Ines Chami et al. “Hyperbolic Graph Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 32*. 2019, pp. 4869–4880.

[2] Christopher De Sa et al. “Representation tradeoffs for hyperbolic embeddings”. In: *Proceedings of the 35th International Conference on Machine Learning* 80 (2018), pp. 4460–4469.

[3] Fragkiskos Papadopoulos et al. “Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces”. In: *2010 Proceedings IEEE INFOCOM*. 2010, pp. 1–9.