Embedding project

1 Our initial idea (12.06.2019)

Motivation:

- Somehow use our intuition that networks are usually formed from different clusters, which also hierarchically consist of smaller clusters;
- Try to formalize a notion of varying dimension: some parts of a graph can be more complex than others, so we need more dimensions for them;
- Also try to formalize a notion of varying curvature, in order to be not limited by a fixed one.

A possible algorithm:

- Consider the whole graph, choose appropriate curvature and dimension, embed a graph into the corresponding space;
- Compute "residual" graph, find its subgraph with largers error and approximately constant curvature, embed it into the corresponding space;
- Continue the process with other subsets;
- When compute the distance between two nodes, use its common subspace.

Some comments and open questions:

- Global curvature of a graph is not defined and studied. It would be interesting to define it based on some practical metric. (And it is not clear how it would be connected with local curvatures of nodes/edges);
- Intuitively, it seems to be desirable to move from larger subsets of nodes in the algorithm to smaller ones;
- Maybe we can use community detection at some point of the algorithm.

2 Our current simple plan (28.06.2019)

We think that we will start our experiment with the following simple approach, in order to see whether we can benefit from our general idea.

- Take our graph and apply a community detection algorithm to it. There are two options which we discussed: 1) apply some hierarchical algorithm in order to get several levels of granularity as an output; 2) get only one level of communities, but with some overlaps. The further ideas can be applied to both cases, but we like the second one more, so, let me describe this one.
- Assume that we have some budget for the overall dimension. Then, we split this budget into two parts. The first we use to embed the whole graph, using some standard graph embedding approach (using an existing algorithm to choose the curvature). The second part we use in the following way: we embed each cluster separately in its own space (choosing curvature individually for clusters).
- When we have to measure the distance between two elements, if the belong to the same cluster, then we use the corresponding cluster embedding, if they belong to different clusters, then we use the embedding of the whole graph.
- One more idea that we discussed: when we embed the whole graph we care more about long distances, since short ones will be improved at the cluster embedding stage. Therefore, in the learning procedure we can sample only the pairs which are far away from each other.

Speaking about the practical realization, we decided to do the following:

- We want to improve the results of [4] using their algorithm and their datasets.
- We take the code from their github: `https://github.com/HazyResearch/hyperbolics`.
- Using this, it seems to be fairly easy to make some starting experiments.

Also, we discussed the following possible directions for theoretical (or algorithmic) research:

- As we discussed before, it would be very useful to estimate optimal curvature and dimension given a graph (optimal in terms of some target metric, e.g., distortion).
- It would be interesting to understand the optimal way to split the overall maximum dinension into two parts: for global graph embedding and local cluster embeddings. On the one hand, global graph embedding is expected to be coarser. On the other hand, the number of vertices in the whole graph is larger.

- Probably unrelated note: it is always useful to understand which algorithm is better at which distances (longer or shorter ones).

3 Measuring quality of embeddings

Mean Average Precision (MAP) Let us consider the node v. $N_i(v)$ be the number of nodes closest to i we have to consider to cover i neighbors of i. Note that in the ideal situation we have $N_i(v) = i$. Then MAP is defined as

$$\frac{1}{n}\sum_{v}\frac{1}{deg(v)}\sum_{i=1}^{deg(v)}\frac{i}{N_i(v)}.$$

Distortion This is a standard metric for graph embeddings.

$$D(f) = \frac{1}{\binom{n}{2}}\sum_{u\neq v}\frac{|d(f(u),f(v)) - d_G(u,v)|}{d_G(u,v)}.$$

There is also a variant called worst-case distortion:

$$D_{WC}(f) = \frac{\max_{u\neq v} d(f(u),f(v))/d_G(u,v)}{\min_{u\neq v} d(f(u),f(v))/d_G(u,v)}$$

4 Curvature in graphs

There are many different notions of graphs curvature. Here we describe a few of them, starting with the most prominent one Ollivier-Ricci curvature.

4.1 Ollivier-Ricci curvature

To define this curvature we first need to introduce the Wasserstein metric.

Let $(\mathcal{X}, d)$ be a metric space and consider two probability measures μ_1 and μ_2 on this space. Recall that a coupling between μ_1 and μ_2 is a joint probability measure μ whose marginals are μ_1 and μ_2. Let $\Gamma(\mu_1, \mu_2)$ denote the set of all such couplings. The Wasserstein metric (Kantorovich-Rubenstein distance of order one) between μ_1 and μ_2 is then given by

$$W_1(\mu_1,\mu_2) = \inf_{\mu\in\Gamma(\mu_1,\mu_2)}\int_{\mathcal{X}\times\mathcal{X}} d(x,y)\,d\mu(x,y) \tag{1}$$

Due to a duality theorem by Kantorovich and Rubenstein we have the following equivalent definition of $W_1(\mu_1, \mu_2)$

$$W_1(\mu_1,\mu_2) = \sup_{f\in L_1}\int_{\mathcal{X}} f(x)\,d\mu_1(x) - \int_{\mathcal{X}} f(y)\,d\mu_2(y), \tag{2}$$

where the supremum is taken over all Lipschitz continuous function on (X, d) with Lipschitz constant 1, i.e.

$$|f(x) - f(y)| \leq d(x, y).$$

For a graph G we consider the shortest path metric on G, denoted by d_G, and let W_1^G denote the Wasserstein metric with respect to the metric space (G, d_G). Furthermore, for each node v let m_v denote the uniform probability measure on the neighbors of v, i.e.

$$m_v(u) = \frac{1_{u \sim v}}{\deg(v)},$$

where $\deg(v)$ denotes the degree of v. Then the classic definition of Ollivier-Ricci curvature between two neighboring nodes $v \sim u$ in G is defined as

$$\kappa_G(u, v) = 1 - W_1^G(m_v, m_u). \tag{3}$$

It is important to note that Ollivier-Ricci curvature is defined in much more generality in terms of metrics and random walks, see [8]. Thus different version on graphs can be considered. Definition (3) corresponds to the classical choices of graph distance and random walk on the graph.

The strength of Ollivier-Ricci curvature lies in the fact that when we consider its continuous version on Riemannian manifolds than this converges to the Ricci curvature, see [8, Example 7]. It turns out that this also holds when we consider dense random geometric graphs on Riemannian manifolds (upcoming paper). However, for this one needs to deviate from the classical version and consider random-walks on larger neighborhoods. Still these results highlights that Ollivier-Ricci curvature can properly encode the curvature of the underlying manifold.

Below we list some simple examples of Ollivier-Ricci curvature in graphs.

Circle Consider a circle on n nodes and let $v \sim u$ be two neighbors. Then $W_1^G(m_u, m_v) = 1$ and hence $\kappa_G(u, v) = 0$.

Trees Consider any tree graph T, let $v \sim u$ be two neighbors. Then Proposition 2 in [5] states that

$$\kappa_G(m_x, m_y) = -2\left(1 - \frac{1}{\deg(v)} - \frac{1}{\deg(u)}\right)_+,$$

where $(t)_+ = \max\{0, t\}$. In particular, if either $\deg(v) = 1$ or $\deg(u) = 1$ then it follows that $\kappa_G(u, v) = 0$. However, if $\deg(v) \geq 3$ for all nodes v then $\kappa_G(v, u) < 0$. Thus trees are intrinsic examples of negatively curved graphs, according to Ollivier-Ricci curvature.

Lattice Consider the 2-dimensional lattice $\mathbb{Z}^2$ and let $v \sim u$ be two neighbors. Then the optimal transportation from m_u to m_v is given by a translation by the vector $y - x$ and hence $W_1^G(m_v, m_u) = d_G(u, v) = 1$, see also [8, Example 5], and thus $\kappa_G(u, v) = 0$.

Complete graph Consider a complete graph on n nodes. Then, for any two nodes u and v it follows from Example 1 in [5] that $\kappa_G(m_v, m_u) = \frac{n-2}{n-1} \to 1$ as $n \to \infty$.

5 Practical tasks

5.1 Network reconstruction and link prediction

[3] uses some standard complex networks: Blogcatalog, YouTube, HepTh, AstroPh, protein-protein interactions. As metrics they use Precision at k and Mean Average Precision (MAP).

[6] uses the following networks: AstroPh, CondMat, GrQc, HepPh. Each dataset is split into train, validation, and test sets. Parameters are tuned on validation, MAP is measured on test.

[6] also uses transitive closure of the WordNet noun hierarchy. Tasks: reconstruction and link prediction. Measure mean rank and (MAP). The same dataset is used in [9], where MAP is measured. Similarly, [2] uses this, treats link prediction as classifiaction task and measures precision, recall, F1.

[9] also uses fully-balanced trees along with phylogenetic trees expressing genetic heritage (of mosses growing in urban environments), a graph of Ph.D. advisor-advisee relationships, biological sets involving disease relationships, protein interactions in yeast bacteria, collaboration network Gr-QC. Measure MAP and distortion.

[7] uses the following taxonomies: WordNet (noun and verb hierarchy), EuroVoc, ACM, MeSH. Measures MR and MAP. In addition, they also evaluate how well the norm of the embeddings correlates with the ground-truth ranks in the embedded taxonomy: they measure the Spearman rank-order correlation of the normalized rank with the norm of the embedding.

[7] also embeds Enron Email Corpus and measures Spearman correlation of the norms of the embedding with the organizational rank.

5.2 Lexical Entailment

[6] uses HyperLex datasets. Uses embedding of WordNet. Records Spearman's rank correlation with the ground-truth ranking.

5.3 Node classification

[3] uses Blogcatalog and protein-protein interactions. Use embeddings as features (as input to a one-vs-rest logistic regression).

6 Selected relevant papers

Graph embedding techniques, applications, and performance: A survey [3] A survey of graph embedding techniques, covers many different methods, but all based on euclidean space (with euclidean distance or dot product). The code is available at `https://github.com/palash1992/GEM`.

What relations are reliably embeddable in Euclidean space? [1] About embeddings of directed graphs to euclidean space. Also contains a short background on undirected graphs (section 1.1), in particular cite some known theoretical results on the subject. Also, focus on precision of embeddings. Maybe we can use something as a starting point for the theory.

Poincare embeddings for learning hierarchical representations [6] Influential paper - they were the first to use hyperbolic embeddings in computer science. Embed in multidimensional Poincaré ball. Propose embedding algorithm based on Riemannian optimization. The code is available at `https://github.com/facebookresearch/poincare-embeddings`.

Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry [7] Follow-up for the previous work, show that it is better to embed to Lorentx model of hyperbolic space.

Learning Mixed-Curvature Representations in Product Spaces [4] Embed graphs to a product manifold combining multiple copies of spherical, hyperbolic, and Euclidean spaces. The code is available at `https://github.com/HazyResearch/hyperbolics`

Representation Tradeoffs for Hyperbolic Embeddings [9] Interesting observation that not only dimensionality, but also precision is important. Propose a new algorithm for graph embedding: first, embed a graph into a weighted tree, and then embed that tree into the hyperbolic disk. The code is available at `https://github.com/HazyResearch/hyperbolics`.

Hyperbolic Entailment Cones for Learning Hierarchical Embeddings [2] Propose a new method for embedding directed acyclic graphs (DAGs). Use nested geodesically convex cones, which improve embeddings in both euclidean and hyperbolic spaces.

References

[1] Robi Bhattacharjee and Sanjoy Dasgupta. What relations are reliably embeddable in euclidean space? *arXiv preprint arXiv:1903.05347*, 2019.

[2] Octavian-Eugen Ganea, Gary Becigneul, and Thomas Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. In *International Conference on Machine Learning*, pages 1632–1641, 2018.

[3] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[4] Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. Learning mixed-curvature representations in product spaces. *ICLR*, 2019.

[5] Jürgen Jost and Shiping Liu. Ollivier's ricci curvature, local clustering and curvature-dimension inequalities on graphs. *Discrete & Computational Geometry*, 51(2):300–322, 2014.

[6] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, pages 6338–6347, 2017.

[7] Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning*, pages 3776–3785, 2018.

[8] Yann Ollivier. Ricci curvature of markov chains on metric spaces. *Journal of Functional Analysis*, 256(3):810–864, feb 2009.

[9] Frederic Sala, Chris De Sa, Albert Gu, and Christopher Re. Representation tradeoffs for hyperbolic embeddings. In *International Conference on Machine Learning*, pages 4457–4466, 2018.