Group meeting - January 10, 2020

Paper review & Research progress

Truong Son Hy *

*Department of Computer Science
The University of Chicago

Ryerson Physical Lab

Paper review (1)(a)

Title: Efficient Graph Generation with Graph Recurrent Attention Networks (NeurIPS'19)

Summary:

1. The framework better captures the **auto-regressive** conditioning between the already-generated and to-be-generated parts of the graph using GNNs with attention.
2. Parameterizing the output distribution per block using a mixture of Bernoulli that captures the correlations among generated edges within the block.

Paper review (1)(b)

What do they mean:

1. Before with RNN-based graph generative models, people generate graph vertex-by-vertex and edge-by-edge sequentally. Now, the authors generate the graph block-by-block.
2. Within a block, they apply mixture of Bernoulli to generate the edges.

Graph G, adjacency A and vertex ordering π:

$$p(G) = \sum_{\pi} p(G, \pi) = \sum_{\pi} p(A^{\pi})$$

Lower triangle part of A^{π} as L^{π}. We generate 1 block b_t of L^{π} (that is a group of rows in L^{π}) at a time t:

$$p(L^{\pi}) = \prod_{t=1}^{T} p(L^{\pi}_{b_t} | L^{\pi}_{b_1}, .., L^{\pi}_{b_{t-1}})$$

Paper review (1)(c)

Datasets:

1. **Grid**: 100 generated standard 2D grid graphs with $100 \leq |V| \leq 400$.
2. **Protein**: 918 protein graphs with $100 \leq |V| \leq 500$. Each protein is represented by agraph, where nodes are amino acids and two nodes are connected by an edge if they are less than 6 Angstroms away.
3. **Point Cloud**: 41 simulated 3D point clouds of household objects with an average graph size of over 1k nodes, and maximum graph size over 5k nodes.

Metrics:

1. Total variation (TV) distance. Permutation invariant? Graph matching algorithm?
2. Compare the spectra of the graphs by computing the eigenvalues of the normalized graph Laplacian.

Observation: Ambiguous metrics! No guarantee to generate good molecules!

Paper review (2)(a)

Title: Generative models for graph-based protein design (NeurIPS'19)

Summary:

1. An alternative, top-down framework for protein design that directly learns a **conditional generative model** for protein sequences given a specification of the target structure, which is represented as a graph over the amino-acids $\rightarrow$ **language modeling**.
2. Graph Attention Networks augmented with edge features and an auto-regressive decoder.
3. What is the graph here? $\rightarrow$ Nearest-neighbor graph.

Paper review (2)(b)

For a rigid-body design problem, given a fixed set of **back-bone** coordinates $\mathcal{X} = \{x_i \in \mathbb{R}^3 : 1 \leq i \leq N\}$. We need to find which amino-acid s_i corresponds to the coordinate x_i?

Auto-regressive decomposition:

$$p(s|x) = \prod_i p(s_i|x, s_{<i})$$

with the conditionals are parameterized in terms of two sub-networks:

- **Encoder**: computes node embeddings from structure-based node features of x and edge feature of $x \rightarrow$ Structured Transformer.
- **Decoder**: auto-regressively predicts letter s_i given the preceding sequence and structural embeddings from the encoder.

Paper review (2)(c)

Relative spatial encodings $\leftarrow$ I think we can use it as input to Cormorant! Augment the points x_i with orientation O_i:

$$O_i = [b_i, n_i, b_i \times n_i]$$

where b_i is the negative bisector of angle between the rays $(x_{i-1} - x_i)$ and $(x_{i+1} - x_i)$, and n_i is a unit vector normal to that plane:

$$u_i = \frac{x_i - x_{i-1}}{|x_i - x_{i-1}|} \qquad b_i = \frac{u_i - u_{i+1}}{|u_i - u_{i+1}|} \qquad n_i = \frac{u_i \times u_{i+1}}{|u_i \times u_{i+1}|}$$

Spatial edge features:

$$e_{ij} = \left(r(|x_j - x_i|), O_i^T \frac{x_j - x_i}{|x_j - x_i|}, q(O_i^T O_j) \right)$$

where $r(.)$ and $q(.)$ are encondings done by neural networks.

Paper review (3)(a)

Title: N-Gram Graph: Simple Unsupervised Representation for Graphs, with Applications to Molecules (NeurIPS'19)

Summary:

1. Constructs a compact representation for the graph by assembling the vertex embeddings in short random walks in the graph.
2. The whole computation is just a simple graph neural network that needs no training.

Paper review (3)(b)

What do they mean: Efficiently compute the embedding by **dynamic programming** that can also be written in **message passing** scheme.

The embedding f_p of an n-gram p is simply the element-wise (Hadamard) product of the vertex embeddings in that walk:

$$f_p = \prod_{i \in p} f_i$$

Embedding of **all** n-gram walks defined as the sum:

$$f_{(n)} = \sum_{p:|p|=n} f_p$$

The final n-gram graph representation up to length T as a concat:

$$f_G = [f_{(1)}; ..; f_{(T)}]$$

Cormorant-related projects

1. Force learning:
   - Compare $f = \partial E/\partial p$ to the MD-17 ground truth (only with energy E training, without any force training).
   - Optimize $|E - \hat{E}| + \lambda|\partial E/\partial p - \hat{f}|$ by letting PyTorch done the second-order computation in the background.
   - If it does not work then compute manually block $\partial^2 E/\partial p \partial w$ of the Hessian.
2. 3D structure generation:
   - Design a **decoder** based on Cormorant. The **encoder** is just the original Cormorant.
   - Try original VAE $\rightarrow$ Brute-force search for useful structures.
   - Conditional/Goal-oriented VAE trained along with energy prodiction.
   - Erik suggests **Boltzmann distribution**.
3. Graph structure generation: The first two papers presented today are pretty good reference.

Learnable/Differentiable MMF & GNNs

Before: Finding the sequence of rotations of MMF is done by heuristics.

Now: Differentiable way of doing so using GNNs.

Reference: Wavelets on Graphs via Deep Learning (Guibas' paper).