Learning Deep Generative Models of Graphs

(ICML 2018)

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, Peter Battaglia

DeepMind UK

https://qdata.github.io/deep2Read

Presenter: Arshdeep Sekhon

Fall 2018

Motivation

- knowledge graphs
- social interaction nets
- physical world
- molecules

capturing the distribution of a particular family of graphs is important

- drug discovery: dsicover new structures after smpling from decoder model
- priors for bayesian structure learning
- semantic graph representations for natural language sentences

Motivation

Old approaches

- based on random graph models: independece assumptions
- grammar based

Motivation

- this generative model makes no structural assumptions (independence)
- avoids the brittleness of grammar-based techniques.

Overview

- add new structure to existing graph
- probability of addition event depends on history of graph derivation
- represent graph using GNN

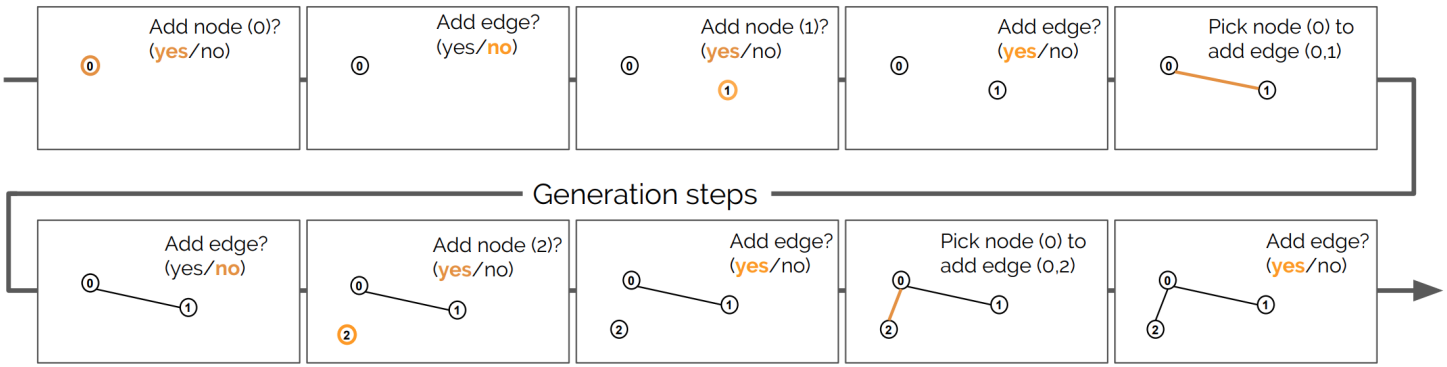
Method

- Sequential Generation process
- generate one node at a time
- connect node to current partial graph by creating edges one by one

Method

In Each Iteration:

1. sample whether to add a new node of a particular type or terminate
2. if a node type is chosen, we add that type node
3. check if any further edges are needed to connect the new node to the existing graph
4. if yes, select a node in the graph and add an edge connecting the new node to the selected node.
5. The algorithm goes back to step (3)
6. repeat until the model decides not to add another edge.
7. go back to step (1) to add subsequent nodes.



Graph generation process

sequence of structure building actions:

- (1) add a new node or not (with probabilities provided by an $f_{addnode}$ module),
- (2) add a new edge or not (probabilities provided by $f_{addedge}$), and
- (3) pick one node to connect to the new node (probabilities provided by f_{nodes}).

Learning Graph Generative Models

- Can use a number of different generative models to model it
- treat the sequences as sentences in natural language: use LSTM
- This paper: use graph nets to model this sequential decision process instead
- $f_{addnode}$, $f_{addedge}$ and f_{nodes} are GNNs

Learning the Generative Model

Propagation Process $prop(h_V, G)$:

- $h_v \in R^H$ with each node $v \in V$
- $a_v = \Sigma_{u:(u,v)\in E} f_e(h_u, h_v, x_{u,v})$
- $h'_v = f_n(a_v, h_v)\ \forall v \in V$

Graph Representation$h_G = R(h_V, G)$:

- $h_G = \Sigma_{v\in V} h_v^G$
- $h_G = \Sigma_{v\in V} g_v^G \odot h_v^G$

Probabilities of Structure Building Decisions

$$\mathbf{h}_V^{(T)} = \text{prop}^{(T)}(\mathbf{h}_V, G)$$

$$\mathbf{h}_G = R(\mathbf{h}_V^{(T)}, G)$$

$$f_{addnode}(G) = \text{softmax}(f_{an}(\mathbf{h}_G))$$

$$f_{addedge}(G, v) = \sigma(f_{ae}(\mathbf{h}_G, \mathbf{h}_v^{(T)}))$$

$$s_u = f_s(\mathbf{h}_u^{(T)}, \mathbf{h}_v^{(T)}), \quad \forall u \in V$$

$$f_{nodes}(G, v) = \text{softmax}(\mathbf{s})$$

a) *faddnode*(*G*)

- take an existing graph G and h_V as input
- make the decision whether to terminate the algorithm or add another node
- first run T rounds of propagation to update node vectors,
- compute a graph representation vector and predict an output from there through a standard MLP followed by softmax or logistic sigmoid.
- After the predictions are made, the new node vectors h_V^T are carried over to the next step
- the same carry-over is applied after each and any decision step.
- This makes the node vectors recurrent, across both the propagation steps and the different decision steps

b) $f_{addedge}(G, v)$ and c) $f_{nodes}(G, v)$

- to get the probability of adding an edge to the newly created node v through a different MLP f_{ae}, after getting the graph representation vector h_G
- $f_{nodes}(G, v)$: after T rounds of propagation, we compute a score for each node, (Eq. 9)
- which is then passed through a softmax to be properly normalized (Eq. 10) into a distribution over nodes.
- f_s maps pairs h_u and h_v to a score s_u for connecting u to the new node v

Initializing Node States

- $h_v = f_{init}(R_{init}(h_V, G), x_v)$

the model can also be made conditional, by adding conditioning information in one of the mlps

Training and Evaluation

- graph generative model defines a joint distribution $p(G, \pi)$ over graphs G and node and edge ordering π
- interested in the marginal: $p(G) = \Sigma_{\pi \in P(G)} p(G, \pi)$

$$p(G) = \sum_{\pi} p(G, \pi) = \mathbb{E}_{q(\pi|G)} \left[\frac{p(G, \pi)}{q(\pi \mid G)} \right].$$

- $q(\pi|G)$ is any proposal distribution over permutations
- get estimate by generating a few samples from $q(\pi|G)$ and then average $\dfrac{p(G, \pi)}{q(\pi|G)}$

Ttraining and Evaluation

- direct optimization of $log(P(G))$ is intractable
- learn $p(G, \pi)$ by maximizing the expected joint likelihood

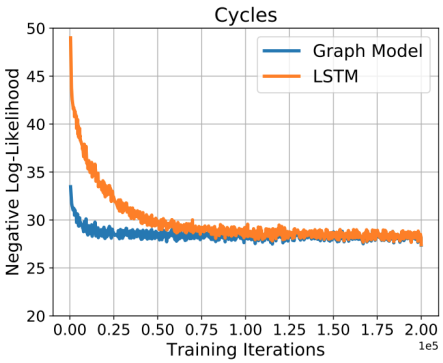
$$\mathbb{E}_{p_{data}(G,\pi)}[\log p(G,\pi)] = \mathbb{E}_{p_{data}(G)}\mathbb{E}_{p_{data}(\pi|G)}[\log p(G,\pi)].$$

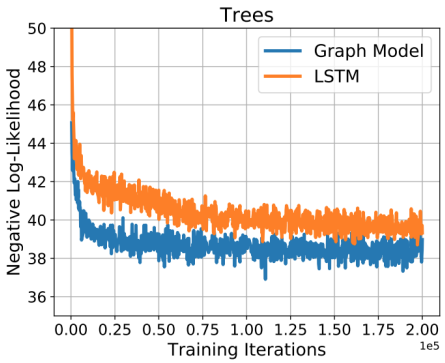
in the experiments: this paper always use a fixed ordering or uniform random ordering for training

Experiments : Synthetic

- (1) cycles, (2) trees, and (3) graphs generated by the BarabasiAlbert model
- generate data on the fly during training
- all cycles and trees have between 10 to 20 nodes
- the BarabasiAlbert model is set to generate graphs of 15 nodes
- each node is connected to 2 existing nodes when added to the graph.
- baselines: Erdos Renyi and LSTM baseline

Results: Synthetic





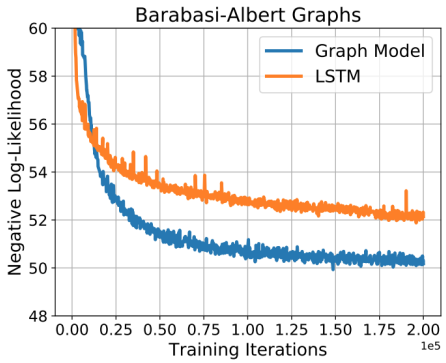


Table 1. Percentage of valid samples for three models on cycles and trees datasets, and the KL-divergence between the degree distributions of samples and data for Barabasi–Albert graphs.

Dataset	Graph Model	LSTM	E–R Model
Cycles	**84.4%**	48.5%	0.0%
Trees	**96.6%**	30.2%	0.3%
B–A Graphs	**0.0013**	0.0537	0.3715

Molecule Generation

- ChemBL Database: A manually curated database of bioactive molecules with drug-like properties. It brings together chemical, bioactivity and genomic data to aid the translation of genomic information into effective new drugs.
- restricted the dataset to molecules with at most 20 heavy atoms, and used a training / validation / test split of 130,830 / 26,166 / 104,664 examples each

Table 2. Molecule generation results. N is the number of permutations for each molecule the model is trained on. Typically the number of different SMILES strings for each molecule < 100.

Arch	Grammar	Ordering	N	NLL	%valid	%novel
LSTM	SMILES	Fixed	1	21.48	93.59	81.27
LSTM	SMILES	Random	< 100	**19.99**	93.48	83.95
LSTM	Graph	Fixed	1	22.06	85.16	80.14
LSTM	Graph	Random	$O(n!)$	63.25	91.44	91.26
Graph	Graph	Fixed	1	20.55	**97.52**	90.01
Graph	Graph	Random	$O(n!)$	58.36	95.98	**95.54**

Table 3. Negative log-likelihood evaluation on small molecules with no more than 6 nodes.

Arch	Grammar	Ordering	N	Fixed	Best	Marginal
LSTM	SMILES	Fixed	1	17.28	15.98	15.90
LSTM	SMILES	Random	< 100	**15.95**	15.76	15.67
LSTM	Graph	Fixed	1	16.79	16.35	16.26
LSTM	Graph	Random	$O(n!)$	20.57	18.90	15.96
Graph	Graph	Fixed	1	16.19	**15.75**	15.64
Graph	Graph	Random	$O(n!)$	20.18	18.56	**15.32**

Conditional generation results

a subset of the ChEMBL training set used in the previous section that contains molecules of 0, 1 and 3 aromatic rings.

Table 4. Conditional generation results.

Arch	Grammar	Condition	Valid	Novel	Atom	Bond	Ring	All
LSTM	SMILES	Training	84.3	82.8	71.3	70.9	**82.7**	**69.8**
LSTM	Graph	Training	65.6	64.9	63.3	62.7	50.3	48.2
Graph	Graph	Training	**93.1**	**92.1**	**81.7**	**79.6**	76.4	66.3
LSTM	SMILES	2-rings	64.4	61.2	7.1	4.2	43.8	0.5
LSTM	Graph	2-rings	54.9	54.2	23.5	21.7	23.9	9.8
Graph	Graph	2-rings	**91.5**	**91.3**	**75.8**	**72.4**	**62.1**	**50.2**
LSTM	SMILES	4-rings	71.7	69.4	46.5	3.7	1.3	0.0
LSTM	Graph	4-rings	42.9	42.1	16.4	10.1	3.4	1.8
Graph	Graph	4-rings	**84.8**	**84.0**	**48.7**	**40.9**	**17.0**	**13.3**

Issues

- Ordering
- Long Sequences : The generation process used by the graph model is typically a long sequence of decisions
- scalability: T
- found that training such graph models is more difficult than training typical LSTM models. The sequences these models are trained on are typically long, and the model structure is constantly changing, which leads to unstable training.