Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation

(NIPS 2018)

Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, Jure Leskovec
Stanford University

https://qdata.github.io/deep2Read

Presenter: Arshdeep Sekhon

Fall 2018

Motivation

- Generating novel graph structures that optimize given objectives while obeying some given underlying rules
- Challenge: optimize desired properties while incorporating highly complex and non-differentiable rules

Overview

- Graph Convolutional Policy Network (GCPN): generate molecules where the generation process can be guided towards specified desired objectives,
- Restrict the output space based on underlying chemical rules
- graph representation, reinforcement learning and adversarial training in a single unified framework.

Overview

- RL: non differentiable rules and exploration
- Adversarial Training: Incorporating prior knowledge specified by a dataset of example molecules
- "GCPN is designed as a RL agent that operates within a chemistry aware graph generation environment."

Method: Notations

- $G : (A, E, F)$
- Adjacency Matrix: $A \in \{0,1\}^n$
- Node Feature Matrix $F :\in R^{n \times D}$
- Edge Conditioned Adjacency Tensor $E \in \{0,1\}^{b \times n \times n}$
- $A = \Sigma_{i=1}^{b} E_i$
- Goal to generate graphs that maximize a given property function $S(G) \in R$

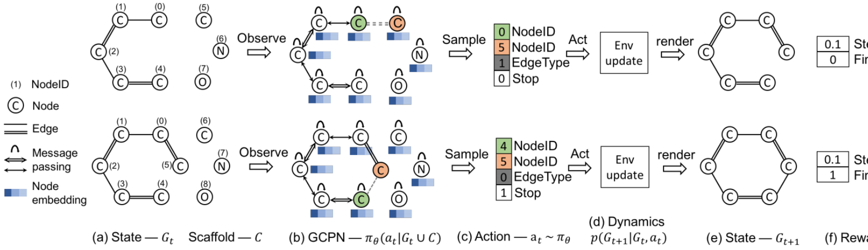
$$max_{G'} E(S(G')) \tag{1}$$

- Prior Knowledge:
  - constraints
  - training data distribution

Method: (a) Graph Generation as Markov Decision Process

- MDP =(S,A,p,R,γ)
- $S = \{s_i\}$: e set of states that consists of all possible intermediate and final graphs
- $A = \{a_i\}$: set of actions or the modification made to current graph at each t
- P(transition dynamics): $p(s_{t+1}|s_t, \ldots, s_0, a_t)$
- $R(s_t)$: reward at s_t
- $p(s_{t+1}|s_t, \ldots, s_0) = \Sigma_{a_t} p(s_{t+1}|s_t, \ldots, s_0, a_t) p(a_t|s_t, \ldots, s_0)$
- policy $\pi_\theta = p(a_t|s_t, \ldots, s_0)$

Method: Overview



Method: (b) Molecule Generation Environment

- State Space: s_t intermediate generated graph G_t at time step t
- Scaffold Subgraphs$\{C_0, \ldots, C_S\}$: (Example: C consists of all b different single node graphs/atom types)
- Action Space: connecting a new subgraph C_i to a node in G_t or connecting existing nodes within graph G_t

Method: (b) Molecule Generation Environment (contd)

- State Transition Dynamics: Domain-specific rules
- Infeasible actions proposed by the policy network are rejected and the state remains unchanged.
- Example : Valency Check
- the environment updates the (partial) molecule according to the actions

Reward Design

- Two types of Rewards:
  - Intermediate
  - Final Rewards

Reward Design

- Two types of Rewards:
  - Intermediate
  - Final Rewards
- Final rewards: domain-specific rewards + adversarial rewards
- domain-specific rewards: (combination of) final property scores, : octanol-water partition coefficient (logP), druglikeness (QED) and molecular weight (MW),penalization of unrealistic molecules

Reward Design

- Two types of Rewards:
  - Intermediate
  - Final Rewards
- Final rewards: domain-specific rewards + adversarial rewards
- domain-specific rewards: (combination of) final property scores, : octanol-water partition coefficient (logP), druglikeness (QED) and molecular weight (MW),penalization of unrealistic molecules
- Intermediate: Adversarial + step wise validity
- A small positive reward is assigned if the action does not violate valency rules, otherwise a small negative reward is assigned.
- When the environment updates according to a terminating action, both a step reward and a final reward are given, and the generation process terminates.

Adversarial Rewards

$$\min_{\theta}\max_{\phi} V(\pi_\theta, D_\phi) = \mathbb{E}_{x\sim p_{data}}[\log D_\phi(x)] + \mathbb{E}_{x\sim\pi_\theta}[\log D_\phi(1-x)]$$

- x represents an input graph,
- p_{data} is the underlying data distribution defined either over final graphs (for final rewards) or intermediate graphs (for intermediate rewards).
- Only D can be trained with stochastic gradient descent, as x is a graph object that is non-differentiable with respect to parameters ϕ.
- use $-V(\pi_\theta, D_\phi)$ as an additional reward together with other rewards, and optimize the total rewards with policy gradient methods

Graph Conv Policy Network

Node Embeddings: perform message passing over each edge type for a total of L layers

$$H^{(l+1)} = \text{AGG}(\mathbf{ReLU}(\{\tilde{D}_i^{-\frac{1}{2}}\tilde{E}_i\tilde{D}_i^{-\frac{1}{2}}H^{(l)}W_i^{(l)}\}, \forall i \in (1, ..., b)))$$

- $D_i = \Sigma_k \tilde{E}_{ijk}$
- $\tilde{E}_i = E_i + I$

Graph Convolutional

Graph Conv Policy Network

Action Prediction

$$a_t = \text{CONCAT}(a_{\text{first}}, a_{\text{second}}, a_{\text{edge}}, a_{\text{stop}})$$

$$\begin{aligned}
f_{\text{first}}(s_t) &= \text{SOFTMAX}(m_f(X)), \\
f_{\text{second}}(s_t) &= \text{SOFTMAX}(m_s(X_{a_{\text{first}}}, X)), \\
f_{\text{edge}}(s_t) &= \text{SOFTMAX}(m_e(X_{a_{\text{first}}}, X_{a_{\text{second}}})), \\
f_{\text{stop}}(s_t) &= \text{SOFTMAX}(m_t(\text{AGG}(X))),
\end{aligned}$$

$$\begin{aligned}
a_{\text{first}} &\sim f_{\text{first}}(s_t) \in \{0,1\}^n \\
a_{\text{second}} &\sim f_{\text{second}}(s_t) \in \{0,1\}^{n+} \\
a_{\text{edge}} &\sim f_{\text{edge}}(s_t) \in \{0,1\}^b \\
a_{\text{stop}} &\sim f_{\text{stop}}(s_t) \in \{0,1\}
\end{aligned}$$

Policy Gradient Training

Proximal Policy Optimization

$$\max L^{\mathrm{CLIP}}(\theta) = \mathbb{E}_t[\min(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)], r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

- pretraining the policy net: any ground truth objective
- $L^{EXPERT}(\theta) = -log(\pi_\theta(a_t|s_t))$
- randomly sample a molecular graph G, and randomly select one connected subgraph G_0 of G as the state s_t.

Experiments

- Property Optimization : molecules with high property score
- Property Targeting : molecules with a pre-specified range of target property score
- Constrained Property Optimization : molecules containing a specific substructure while having high property score

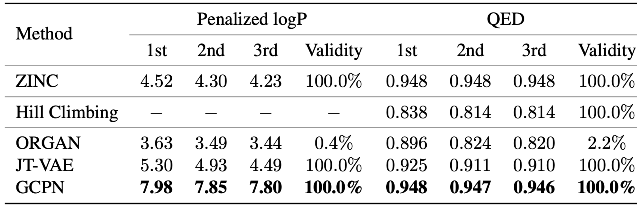
Experimental Setup

- Dataset: Zinc250k
- 9 atom types and 3 edge types
- Baselines:
  - State of the art Junction Tree VAE
  - ORGAN: RL based from text sequence representation

Results:Property optimization.

- Goal: highest possible penalized logP and QED scores.
- Penalized logP : logP score that also accounts for ring size and synthetic accessibility
- QED : indicator of drug-likeness.

Table 1: Comparison of the top 3 property scores of generated molecules found by each model.

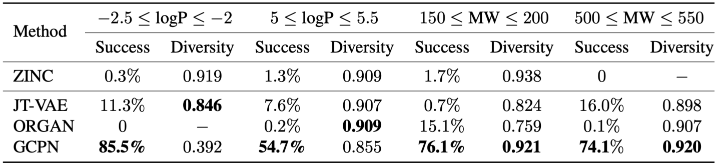


Method	Penalized logP				QED			
	1st	2nd	3rd	Validity	1st	2nd	3rd	Validity
ZINC	4.52	4.30	4.23	100.0%	0.948	0.948	0.948	100.0%
Hill Climbing	–	–	–	–	0.838	0.814	0.814	100.0%
ORGAN	3.63	3.49	3.44	0.4%	0.896	0.824	0.820	2.2%
JT-VAE	5.30	4.93	4.49	100.0%	0.925	0.911	0.910	100.0%
GCPN	**7.98**	**7.85**	**7.80**	**100.0%**	**0.948**	**0.947**	**0.946**	**100.0%**

Results: Property Targeting

- range of Mol Weight and QED
- The RL reward for this task is the L1 distance between the property score of a generated molecule and the range center.

Table 2: Comparison of the effectiveness of property targeting task.



Method	$-2.5 \le \text{logP} \le -2$		$5 \le \text{logP} \le 5.5$		$150 \le \text{MW} \le 200$		$500 \le \text{MW} \le 550$	
	Success	Diversity	Success	Diversity	Success	Diversity	Success	Diversity
ZINC	0.3%	0.919	1.3%	0.909	1.7%	0.938	0	–
JT-VAE	11.3%	**0.846**	7.6%	0.907	0.7%	0.824	16.0%	0.898
ORGAN	0	–	0.2%	**0.909**	15.1%	0.759	0.1%	0.907
GCPN	**85.5%**	0.392	**54.7%**	0.855	**76.1%**	**0.921**	**74.1%**	**0.920**

Results: Constrained Property Optimization

- Optimize the penalized logP while constraining the generated molecules to contain one of the 800 ZINC molecules with low penalized logP, following the evaluation in JT-VAE.
- Since JT-VAE cannot constrain the generated molecule to have certain structure, : the constraint is relaxed such that the molecule similarity $sim(G, G_0)$ between the original and modified molecules is above a threshold δ.

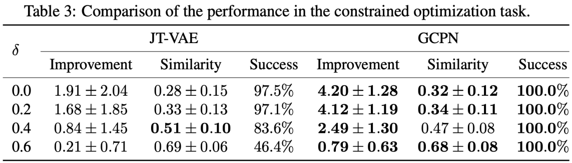


Table 3: Comparison of the performance in the constrained optimization task.

δ	JT-VAE			GCPN		
	Improvement	Similarity	Success	Improvement	Similarity	Success
0.0	1.91 ± 2.04	0.28 ± 0.15	97.5%	$\mathbf{4.20 \pm 1.28}$	$\mathbf{0.32 \pm 0.12}$	**100.0%**
0.2	1.68 ± 1.85	0.33 ± 0.13	97.1%	$\mathbf{4.12 \pm 1.19}$	$\mathbf{0.34 \pm 0.11}$	**100.0%**
0.4	0.84 ± 1.45	$\mathbf{0.51 \pm 0.10}$	83.6%	$\mathbf{2.49 \pm 1.30}$	0.47 ± 0.08	**100.0%**
0.6	0.21 ± 0.71	0.69 ± 0.06	46.4%	$\mathbf{0.79 \pm 0.63}$	$\mathbf{0.68 \pm 0.08}$	**100.0%**