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 $E \in \{0, 1\}^{b \times n \times n}$
- $A = \sum_{i=1}^{b} E_i$
- Goal to generate graphs that maximize a given property function $S(G) \in R$

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

- Prior Knowledge:
  - constraints
  - training data distribution
Method: (a) Graph Generation as Markov Decision Process

- MDP = (S, A, P, R, γ)
- S = {s_i} : 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): \( p(s_{t+1}|s_t, ..., s_0, a_t) \)
- R(s_t): reward at s_t
- \( p(s_{t+1}|s_t, ..., s_0) = \sum_{a_t} p(s_{t+1}|s_t, ..., s_0, a_t) p(a_t|s_t, ..., s_0) \)
- policy \( \pi_\theta = p(a_t|s_t, ..., s_0) \)
Method: Overview

Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation

(NIPS 2018)
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

  Final rewards: domain-specific rewards + adversarial rewards
  - Domain-specific rewards: (combination of) final property scores, 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.
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, \( \text{octanol-water partition coefficient (logP)} \), 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 \(\phi\).
- Use \(-V(\pi_\theta, D_\phi)\) as an additional reward together with other rewards, and optimize the total rewards with policy gradient methods.
Node Embeddings: perform message passing over each edge type for a total of $L$ layers

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

- $D_i = \sum_k \tilde{E}_{ijk}$
- $\tilde{E}_i = E_i + 1$
Graph Conv Policy Network

Action Prediction

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

\[ f_{\text{first}}(s_t) = \text{SOFTMAX}(m_f(X)), \quad a_{\text{first}} \sim f_{\text{first}}(s_t) \in \{0, 1\}^n \]
\[ f_{\text{second}}(s_t) = \text{SOFTMAX}(m_s(X_{a_{\text{first}}}, X)), \quad a_{\text{second}} \sim f_{\text{second}}(s_t) \in \{0, 1\}^{n+c} \]
\[ f_{\text{edge}}(s_t) = \text{SOFTMAX}(m_e(X_{a_{\text{first}}}, X_{a_{\text{second}}})) \quad a_{\text{edge}} \sim f_{\text{edge}}(s_t) \in \{0, 1\}^b \]
\[ f_{\text{stop}}(s_t) = \text{SOFTMAX}(m_t(\text{AGG}(X))) \quad a_{\text{stop}} \sim f_{\text{stop}}(s_t) \in \{0, 1\} \]
Policy Gradient Training

Proximal Policy Optimization

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

- pretraining the policy net: any ground truth objective
- $$L^{\text{EXPERT}}(\theta) = -\log(\pi_\theta(a_t|s_t))$$
- 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Penalized logP</th>
<th>QED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1st</td>
<td>2nd</td>
</tr>
<tr>
<td>ZINC</td>
<td>4.52</td>
<td>4.30</td>
</tr>
<tr>
<td>Hill Climbing</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ORGAN</td>
<td>3.63</td>
<td>3.49</td>
</tr>
<tr>
<td>JT-VAE</td>
<td>5.30</td>
<td>4.93</td>
</tr>
<tr>
<td>GCPN</td>
<td>7.98</td>
<td>7.85</td>
</tr>
</tbody>
</table>
Results: Property Targeting

- range of Mol Weight and QED
- The RL reward for this task is the $L_1$ distance between the property score of a generated molecule and the range center.

<table>
<thead>
<tr>
<th>Method</th>
<th>$-2.5 \leq \log P \leq -2$</th>
<th>$5 \leq \log P \leq 5.5$</th>
<th>$150 \leq MW \leq 200$</th>
<th>$500 \leq MW \leq 550$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Success</td>
<td>Diversity</td>
<td>Success</td>
<td>Diversity</td>
</tr>
<tr>
<td>ZINC</td>
<td>0.3%</td>
<td>0.919</td>
<td>1.3%</td>
<td>0.909</td>
</tr>
<tr>
<td>JT-VAE</td>
<td>11.3%</td>
<td>0.846</td>
<td>7.6%</td>
<td>0.907</td>
</tr>
<tr>
<td>ORGAN</td>
<td>0</td>
<td>–</td>
<td>0.2%</td>
<td>0.909</td>
</tr>
<tr>
<td>GCPN</td>
<td>85.5%</td>
<td>0.392</td>
<td>54.7%</td>
<td>0.855</td>
</tr>
</tbody>
</table>

Table 2: Comparison of the effectiveness of property targeting task.
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 $\text{sim}(G, G_0)$ between the original and modified molecules is above a threshold $\delta$.

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>Improvement</th>
<th>Similarity</th>
<th>Success</th>
<th>Improvement</th>
<th>Similarity</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>1.91 ± 2.04</td>
<td>0.28 ± 0.15</td>
<td>97.5%</td>
<td>4.20 ± 1.28</td>
<td>0.32 ± 0.12</td>
<td>100.0%</td>
</tr>
<tr>
<td>0.2</td>
<td>1.68 ± 1.85</td>
<td>0.33 ± 0.13</td>
<td>97.1%</td>
<td>4.12 ± 1.19</td>
<td>0.34 ± 0.11</td>
<td>100.0%</td>
</tr>
<tr>
<td>0.4</td>
<td>0.84 ± 1.45</td>
<td>0.51 ± 0.10</td>
<td>83.6%</td>
<td>2.49 ± 1.30</td>
<td>0.47 ± 0.08</td>
<td>100.0%</td>
</tr>
<tr>
<td>0.6</td>
<td>0.21 ± 0.71</td>
<td>0.69 ± 0.06</td>
<td>46.4%</td>
<td>0.79 ± 0.63</td>
<td>0.68 ± 0.08</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

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