Graph Convolutional Policy Network

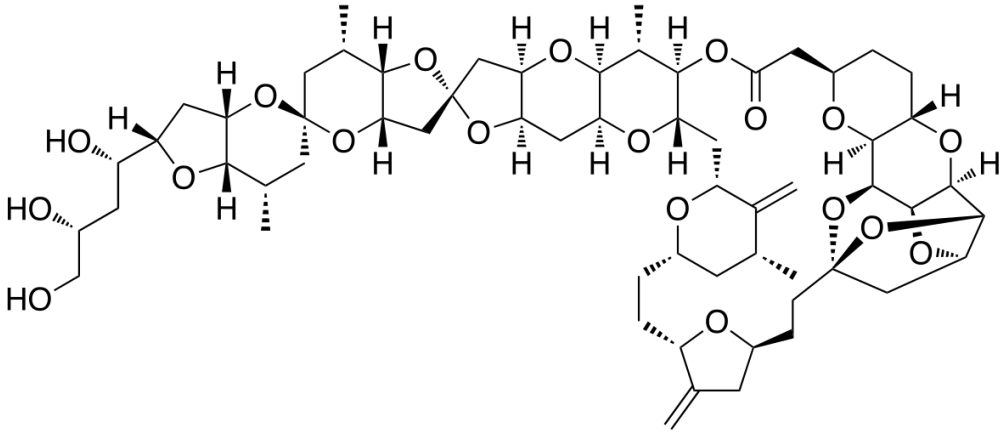
Shanxiu He, Howard Xie, Justin Yi

Original paper by: Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, Jure Leskovec

Contents

Motivation

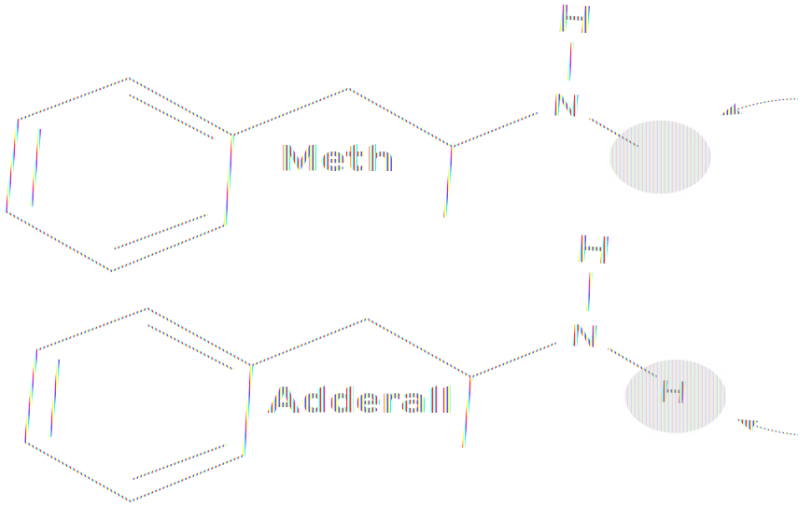
- Many important problems in drug discovery and material science are based on the principle of designing molecular structures with specific desired properties
- However, the space of drug-like molecules is estimated to be in the range of 10^{23} and 10^{60}



E7130

Motivation

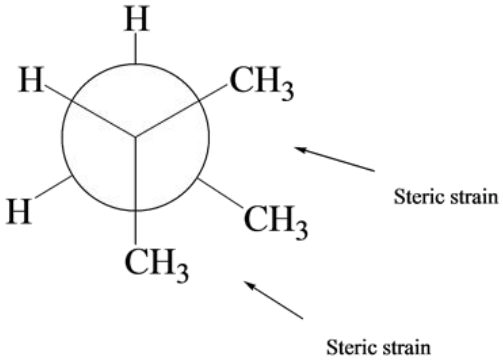
- Additionally, chemical space is discrete, and molecular properties are highly variable to small changes in molecular structure.
- Increased effectiveness of the design of new molecules with application-driving goals would dramatically accelerate the development of novel medicines and materials.

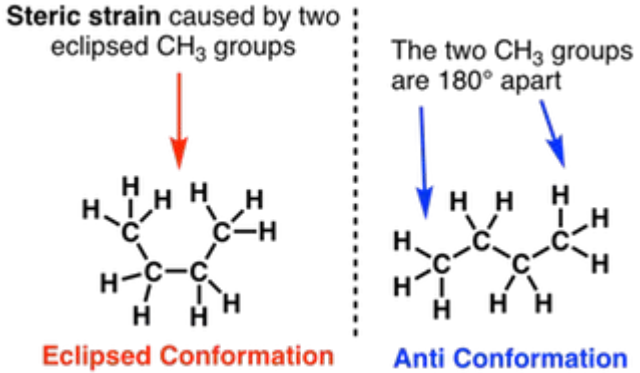


Introduction & Background

Introduction

- Graph Convolutional Policy Network (GCPN)
  - Molecule generation process can be guided towards desiderata while constraining the output space to chemical rules
    - Valency
    - Octanol-water partition coefficient
      - Tendency of chemical to partition itself between organic phase and aqueous phase
    - Druglikeness
    - Molecular weight (MW)
    - Steric Strain





Introduction

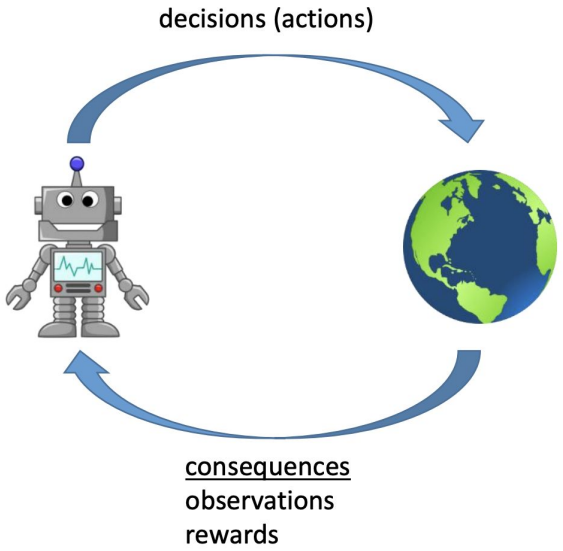
- GCPN employs three well studied concepts
  - Graph Representation
    - Utilizes a Graph Convolutional Network (GCN) to compute node embeddings and a policy on which Markov Decision Process (MDP) is defined
  - Reinforcement Learning (RL)
    - Agent acts on an iterative graph generation process taking the intermediate generated graph as the state and chemical bonds to different atoms as the action space, governed by universal chemical principles via domain specific reward design.
  - Adversarial Training
    - Generative Adversarial Network (GAN) framework to define adversarial rewards to ensure generated molecules resemble expert samples

Introduction

- Evaluation subdomains
  - Property Optimization
  - Property Targeting
    - Specify a target range for particular properties
  - Constrained Property Optimization

Reinforcement Learning Overview

- Agent: takes observations from environment and makes actions
- Environment: the world the agent inhabits
- Desiderata: Agent to take best sequence of actions to complete a task of interest



Markov Decision Processes (MDP)

- $M = \{S, A, T, r\}$

  S : state space (can be continuous or discrete)

  A : action space (can be continuous or discrete

  T : transition function (describes environment dynamics $p(s_{t+1}|s_t, a_t)$

  r : reward function (real valued) $r(s, a)$

- State must contain all information about environment necessary for agent to act optimally
- Markov Property: Next state is conditionally independent of previous states + actions given current state + action

Markov Decision Processes (MDP)

$M = \{S, A, T, r\}$

S : state space (can be continuous or discrete)

A : action space (can be continuous or discrete

T : transition function (describes environment dynamics $p(s_{t+1} | s_t, a_t)$

r : reward function (real valued) $r(s, a)$

RL Objective

Policy $\pi_\theta(a_t|s_t)$

Optimal policy $\pi^*_\theta(a_t|s_t)$

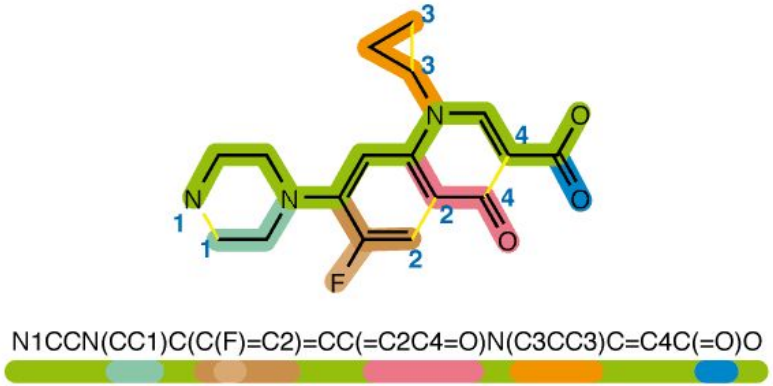
Trajectory distribution

$$p_\theta(\tau) = p_\theta(s_1, a_1, \ldots, s_T, a_T) = p(s_1)\Pi_{t=1}^{T}\pi_\theta(a_t|s_t)\, p(s_{t+1}|s_t, a_t)$$

Find $\theta^* = \max_\theta \mathbb{E}_{\tau\sim p_\theta(\tau)} \sum_{t=1}^{T} r(s_t, a_t)$

Related Work

- Recurrent neural network (RNN) using simplified molecular-input line-entry system (SMILES) string generator using Monte Carlo (MC) tree search and policy gradient.
- Variational Autoencoder (VAE) framework, where molecules are represented as junction trees of small clusters of atoms
- Various sequential graph generation models that are able to specify target molecular properties



High Level Description

Molecule is constructed by

1) connect a new substructure or an atom with an existing molecular graph

2) add bonds to existing atoms

GCPN predicts the action of the bond addition

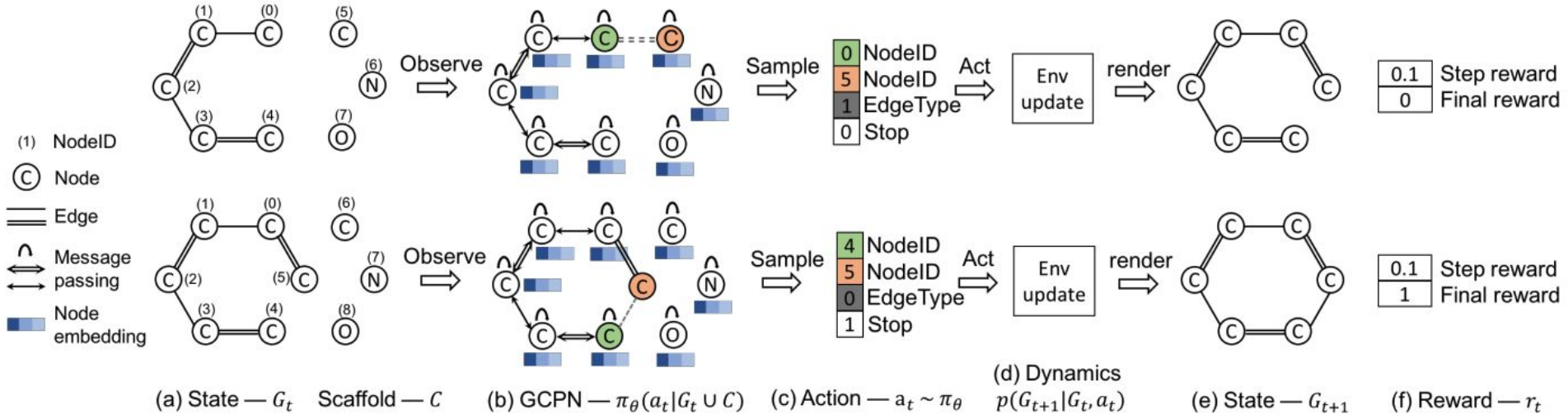
1) train via policy gradient to optimize both property objectives & adversarial loss

Adversarial loss is provided by a graph convolutional network

Methods

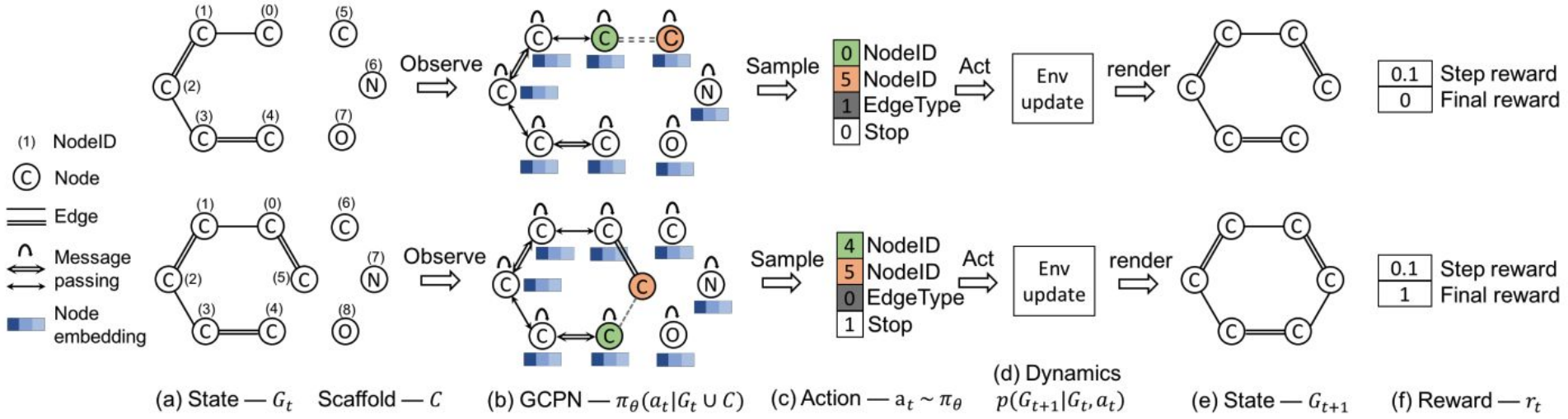
1. Model Introduction & Problem Definition
2. Graph Generation
3. Generation Environment
4. GCPN
5. Policy Gradient Training

Graph Convolutional Policy Network (GCPN)



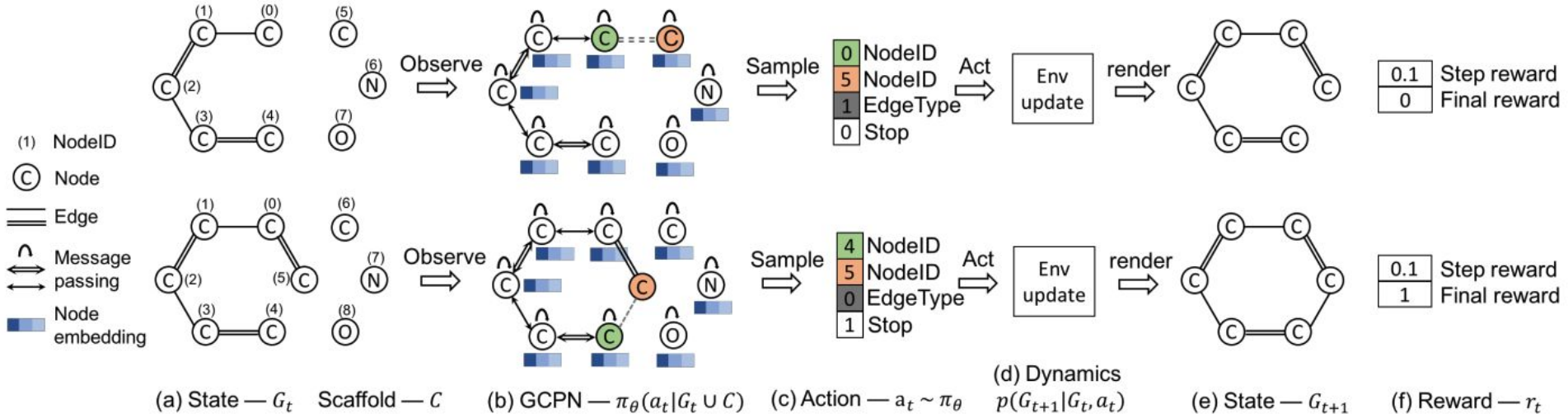
- (a): State is defined as intermediate graph and the set of scaffold subgraphs is appended for GCPN calculation

Graph Convolutional Policy Network (GCPN)



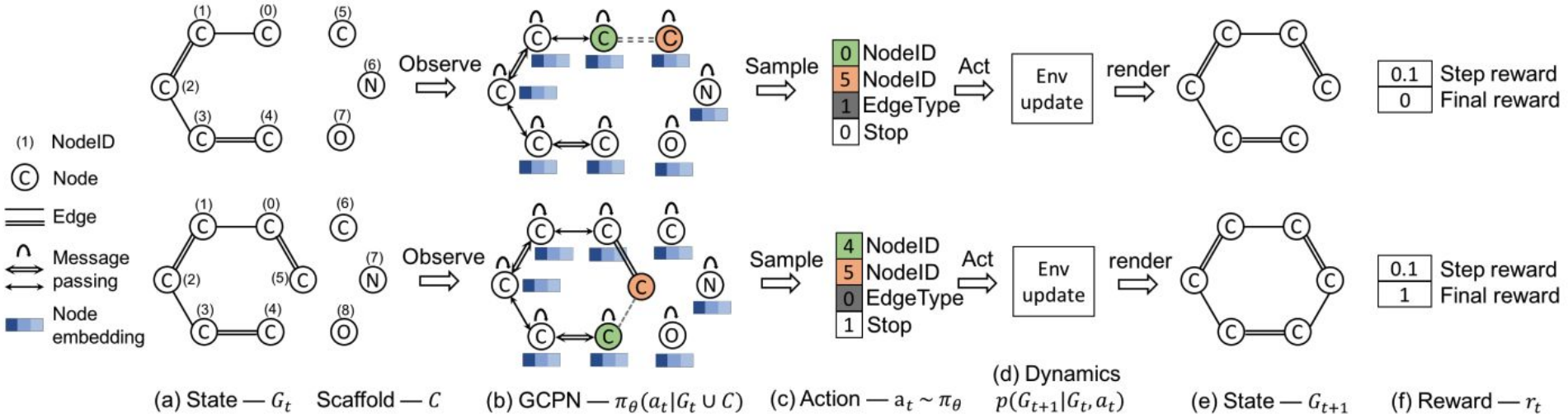
- (b): GCPN conducts message passing to encode state as node embeddings and produces a policy

Graph Convolutional Policy Network (GCPN)



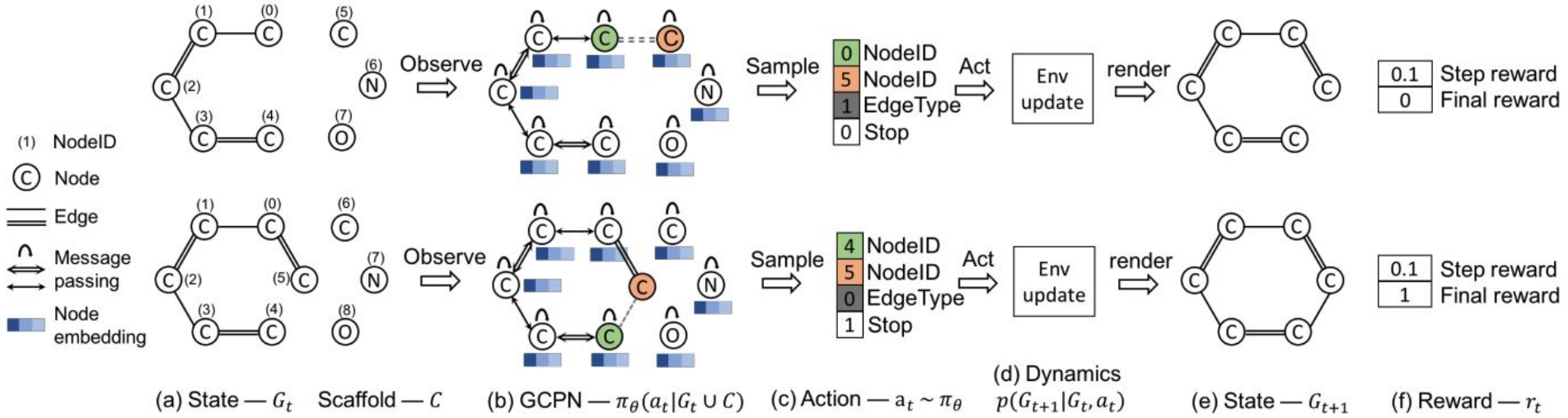
- (c): Four component action is sampled from the policy

Graph Convolutional Policy Network (GCPN)



- (d): Environment performs a valency check on intermediate state for structure validity

Graph Convolutional Policy Network (GCPN)



- (e): Next state is returned
- (f): Associated reward is returned

1. Formal Problem Definition

Represent a graph $G := (A, E, F)$

Adjacency matrix $A \in \{0, 1\}^{n \times n}$

Node feature matrix $F \in \mathbb{R}^{n \times d}$

Discrete edge-conditioned adjacency tensor, b possible edge types $E \in \{0, 1\}^{b \times n \times n}$

$E_{i,j,k} = 1$ if there exists an edge of type i between nodes j and k, $A = \sum_{i=1}^{b} E_i$

Primary objective is to generate graphs that maximize a given property function $S(G) \in \mathbb{R}$

2. Graph Generation as a Markov Decision Process (MDP)

General Decision Process Formulation: $M = \{\mathcal{S}, \mathcal{A}, P, R, \gamma\}$

Set of states (all possible intermediate and final graphs) $\mathcal{S} = \{s_i\}$

Set of actions that describe the modification made to current graph at each time step $\mathcal{A} = \{a_i\}$

Transition dynamics specifying possible outcomes of carry out an action $P = p(s_{t+1}|s_t, \ldots, s_0, a_t)$

Reward function after reaching state s_t, $R(s_t)$

Discount factor γ

2. Graph Generation as a Markov Decision Process (MDP)

Graph generation trajectory with final graph s_n, $(s_0, a_0, r_0, \ldots, s_n, a_n, r_n)$

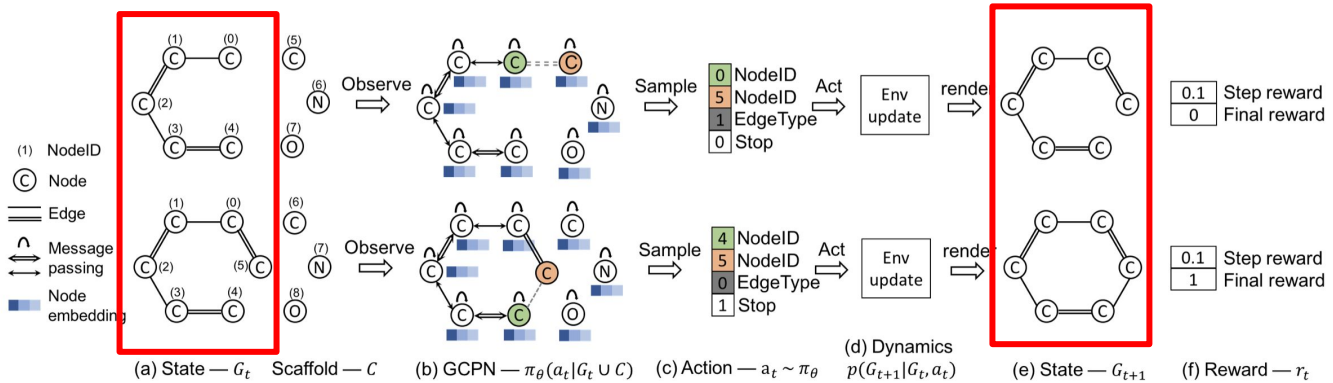
Modification of graph at each time step can be viewed as a state transition distribution:

$$p(s_{t+1}|s_t, \ldots, s_0) = \sum_{a_t} p(a_t|s_t, \ldots, s_0) p(s_{t+1}|s_t, \ldots s_0, a_t)$$

where policy network $\pi_\theta = p(a_t|s_t, \ldots, s_0)$

3. Molecule Generation Environment

1. State Space

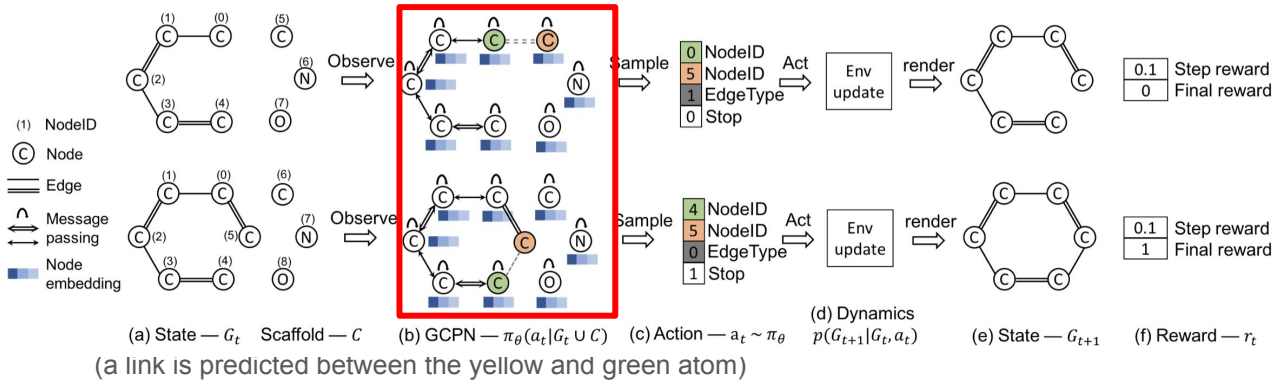


Definition: The state space s_t is the generated graph G_t at time step t. (The boxes represent the state space before and after an action is taken.

Default: G_0 is assumed to be a single node represents a carbon atom.

3. Molecule Generation Environment

2.Action Space



(a link is predicted between the yellow and green atom)

Pre-defined a set of subgraphs C = {C_1, C_2, …, C_s}

Extended Graph at time t: $G_t \cup C$

Action: Connect $C_i \subseteq C$ to G_t or connecting existing nodes within G_t (all disconnected subgraphs are removed once an action is taken)

3. Molecule Generation Environment

3. State Transition Dynamics

Constraints: Infeasible actions proposed by the policy network are rejected

(Graph remains unchanged; Rules such as valency check)

3. Molecule Generation Environment

4. Reward Design

   Final Reward = All Domain Specific Rewards + Adversarial Rewards

Generation stops when termination action is taken; Calculated Final Reward.

<u>Domain Rewards:</u>

Rewards on Octanol-water partition coefficient rewards (logP), druglikeness(QED) and molecular weight (MW);

Penalty on unrealistic molecules (criteria such as excessive steric strain)

3. Molecule Generation Environment

4. Reward Design

Intermediate Rewards:

Step-wise Validity Rewards (valency) + Adversarial Rewards

3. Molecule Generation Environment

4. Reward Design

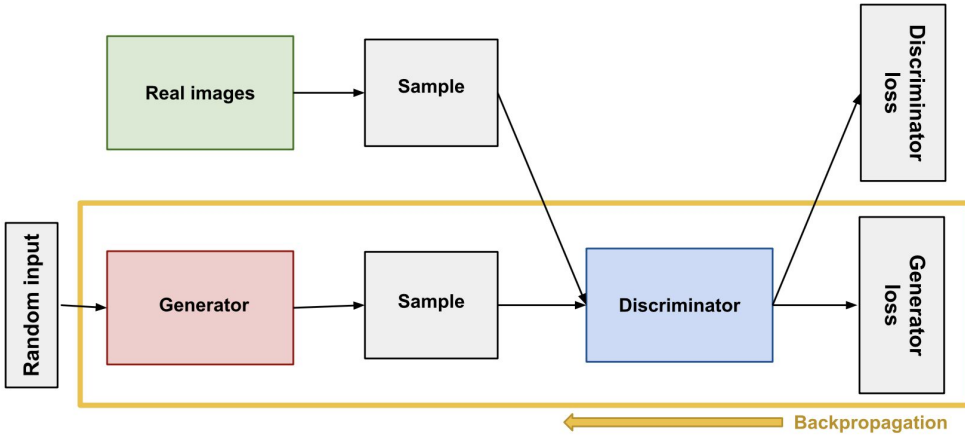
GAN introduction

1. Discriminator & Generator train separately (different objectives)
2. Train until discriminator reaches about 50% accuracy

In this use case, we wish to trair the network via

Discriminator Generator

$$E_x[log(D(x))] + E_z[log(1 - D(G(z)))]$$



Source: https://developers.google.com/machine-learning/gan/generator

3. Molecule Generation Environment

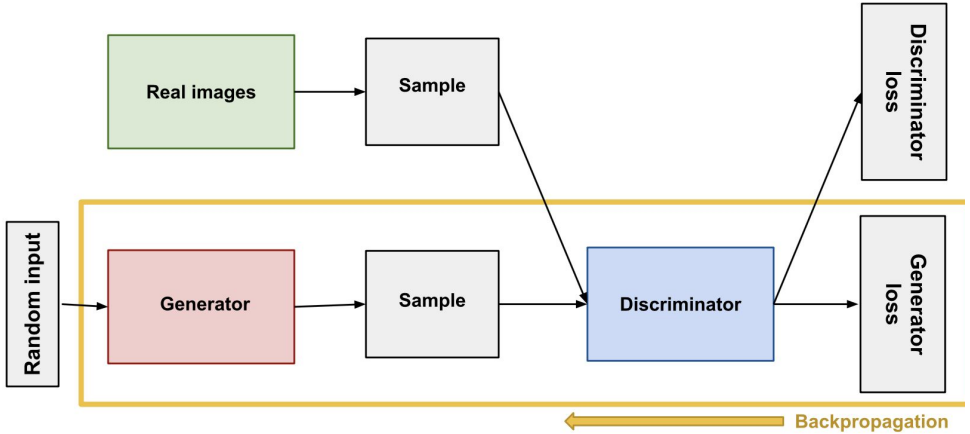
4. Reward Design

GAN introduction

In this use case, we wish to train the network so that our model could generate molecules that are similar to that from the actual data input

Discriminator Generator

$$E_x[log(D(x))] + E_z[log(1 - D(G(z)))]$$



Source: https://developers.google.com/machine-learning/gan/generator

3. Molecule Generation Environment

4. Reward Design

Adversarial Rewards

$$E_x[log(D(x))] + E_z[log(1 - D(G(z)))]$$

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

π_θ is the policy network
D_φ is the discriminator network
x represents an input graph
p_{data} is the underlying data distribution

(Takeaway: This is the rewards for discriminator; Generative Model is benefited from validity + final rewards)

4. GCPN

Input: intermediate graph G_t and the collection of subgraphs C

Output: action a_t, predicts a new link to be added

Node Embedding on high level:

To compute $G_t \cup C$, first compute the node embedding of an input graph thorough GCN.

The authors adopt a variant that, in high level, **perform message passing over each edge type for a total of L layers.**

4. GCPN

Node Embedding:

Baseline GCN: $H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$

At the l_{th} layer of the GCN, we aggregate all messages from different edge types to compute the next layer node embedding $H^{(l+1)} \in R^{(n+c)\times k}$, where n, c are the sizes of G_t and C respectively, and k is the embedding dimension.

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

E_i is the i slice of edge-conditioned adjacency tensor E, and $\sim E_i = E_i + I$; $\sim D_i = \sum_k E_{ijk}$.

$W_i^{(l)}$ is a trainable weight matrix for the i^{th} edge type, and $H_i^{(l)}$ is the node representation i learned in the l^{th} layer.

AGG(.) denotes functions which would be one of {MEAN, MAX, SUM, CONCAT}

We apply L layer GCN to compute the final node embedding matrix $H^{(L)}$

4. GCPN

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$$

Node Embedding:

At the l_{th} layer of the GCN, we aggregate all messages from different edge types to compute the next layer node embedding $H^{(l+1)} \in R^{(n+c)\times k}$, where n, c are the sizes of G_t and C respectively, and k is the embedding dimension.

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

E_i is the i slice of edge-conditioned adjacency tensor E, and $\sim E_i = E_i + I$; $\sim D_i = \sum_k E_{ijk}$.

$W_i^{(l)}$ is a trainable weight matrix for the i^{th} edge type, and $H_i^{(l)}$ is the node representation i learned in the l^{th} layer.

AGG(.) denotes functions which would be one of {MEAN, MAX, SUM, CONCAT}

We apply L layer GCN to compute the final node embedding matrix $H^{(L)}$

4. GCPN

2. Action Prediction

Action at at time step t: a concatenation of four components: selection of two nodes, prediction of edge type, and prediction of termination.

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

4. GCPN

2. Action Prediction

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

$$f_{\text{first}}(s_t) = \text{SOFTMAX}(m_f(X)), \qquad a_{\text{first}} \sim f_{\text{first}}(s_t) \in \{0,1\}^n$$

$$f_{\text{second}}(s_t) = \text{SOFTMAX}(m_s(X_{a_{\text{first}}}, X)), \qquad a_{\text{second}} \sim f_{\text{second}}(s_t) \in \{0,1\}^{n+c}$$

$$f_{\text{edge}}(s_t) = \text{SOFTMAX}(m_e(X_{a_{\text{first}}}, X_{a_{\text{second}}})), \qquad a_{\text{edge}} \sim f_{\text{edge}}(s_t) \in \{0,1\}^b$$

$$f_{\text{stop}}(s_t) = \text{SOFTMAX}(m_t(\text{AGG}(X))), \qquad a_{\text{stop}} \sim f_{\text{stop}}(s_t) \in \{0,1\}$$

High Level: The action is sampled from the distribution.

4. GCPN

2. Action Prediction

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

$$\begin{aligned}
f_{\text{first}}(s_t) &= \text{SOFTMAX}(m_f(X)), & a_{\text{first}} &\sim f_{\text{first}}(s_t) \in \{0,1\}^n \\
f_{\text{second}}(s_t) &= \text{SOFTMAX}(m_s(X_{a_{\text{first}}}, X)), & a_{\text{second}} &\sim f_{\text{second}}(s_t) \in \{0,1\}^{n+c} \\
f_{\text{edge}}(s_t) &= \text{SOFTMAX}(m_e(X_{a_{\text{first}}}, X_{a_{\text{second}}})), & a_{\text{edge}} &\sim f_{\text{edge}}(s_t) \in \{0,1\}^b \\
f_{\text{stop}}(s_t) &= \text{SOFTMAX}(m_t(\text{AGG}(X))), & a_{\text{stop}} &\sim f_{\text{stop}}(s_t) \in \{0,1\}
\end{aligned}$$

M_f denotes multilayer perceptron, where it represents the probability distribution of selecting each node.

The selection of the first node passes to the second by concatenate its probability with the node in extended graph

4. GCPN

2. Action Prediction

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

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

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

M_s maps to selection of second node

M_e predicts the edge type

M_t maps the embedding to a scalar

5. Policy Gradient Training

GCPN predicts the action of the bond addition

GCPN is trained via policy gradient to optimize a reward composed of molecular property objectives and adversarial loss. The adversarial loss is provided by a graph convolutional network based discriminator trained jointly on a dataset of example molecules.

Adopted Proximal Policy Optimization (PPO)

- Policy Gradient
- PPO with Clipped Objective

5. Policy Gradient Training

Adopted Proximal Policy Optimization (PPO)

- Policy Gradient
- PPO with Clipped Objective

Policy Gradient

Necessity of Policy Gradient:

- cannot undo the rewards
- step size matters

Trust Region Approach:

- Determine the maximal step we want to take first
- Locate the optimal point within trust region

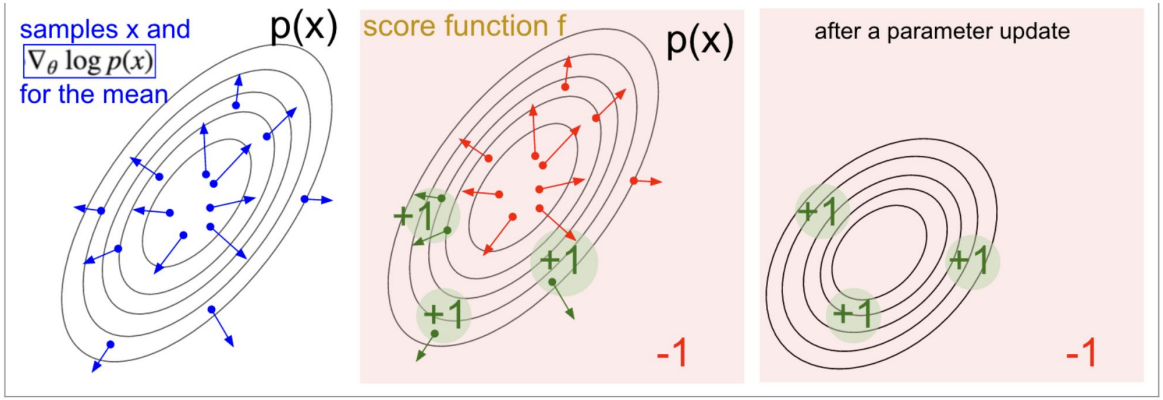
Source: https://jonathan-hui.medium.com/rl-proximal-policy-optimization-ppo-explained-77f014ec3f12

Policy Gradient

If we were to nudge θ in the direction $\nabla_\theta \log p(x;\theta)$
We are to expect increase in x.

Necessity of Policy Gradient: cannot undo the rewards; step size matters.

$$
\begin{aligned}
\nabla_\theta E_x[f(x)] &= \nabla_\theta \sum_x p(x) f(x) && \text{definition of expectation} \\
&= \sum_x \nabla_\theta p(x) f(x) && \text{swap sum and gradient} \\
&= \sum_x p(x) \frac{\nabla_\theta p(x)}{p(x)} f(x) && \text{both multiply and divide by } p(x) \\
&= \sum_x p(x) \nabla_\theta \log p(x) f(x) && \text{use the fact that } \nabla_\theta \log(z) = \frac{1}{z} \nabla_\theta z \\
&= E_x[f(x) \nabla_\theta \log p(x)] && \text{definition of expectation}
\end{aligned}
$$



Source: http://karpathy.github.io/2016/05/31/rl/

PPO with clipped distance

Maintain 2 policy networks

$$\pi_\theta(a_t|s_t)$$

The second is the policy that we last used to collect samples.

$$\pi_{\theta_k}(a_t|s_t)$$

With the idea of importance sampling, we can evaluate a new policy with samples collected from an older policy. This improves sample efficiency.

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right]$$

Source: https://jonathan-hui.medium.com/rl-proximal-policy-optimization-ppo-explained-77f014ec3f12

PPO with clipped distance

New objective:

But as we refine the current policy, the difference between the current and the old policy is getting larger. The variance of the estimation will increase and we will make bad decision because of the inaccuracy. So, say for every 4 iterations, we synchronize the second network with the refined policy again.

$$\pi_{\theta_{k+1}}(a_t|s_t) \longleftarrow \pi_\theta(a_t|s_t)$$

With clipped objective, we compute a ratio between the new policy and the old policy:

$$r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_k}(a_t|s_t)$$

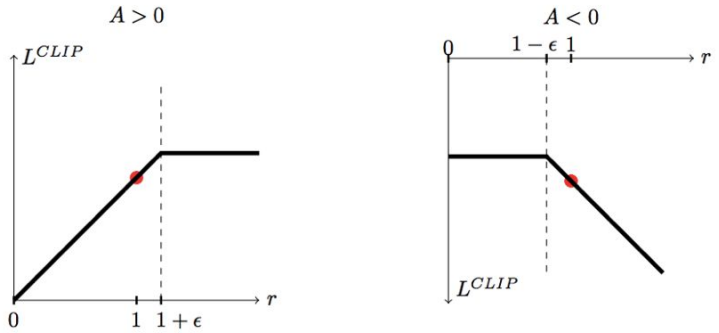
This ratio measures how difference between two policies. We construct a new objective function to clip the estimated advantage function if the new policy is far away from the old policy. Our new objective function becomes:

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau\sim\pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

Source: https://jonathan-hui.medium.com/rl-proximal-policy-optimization-ppo-explained-77f014ec3f12

PPO with clipped distance

If the probability ratio between the new policy and the old policy falls outside the range (1 — ε) and (1 + ε), the advantage function will be clipped. ε is set to 0.2 for the experiments in the PPO paper.



Source

Source: https://jonathan-hui.medium.com/rl-proximal-policy-optimization-ppo-explained-77f014ec3f12

5. Policy Gradient Training

Adopted Proximal Policy Optimization (PPO)

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

5. Policy Gradient Training

Adopted Proximal Policy Optimization (PPO)

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

where $r_t(\theta)$ is the probability ratio that is clipped to the range of $[1-\epsilon, 1+\epsilon]$, making the $L^{\text{CLIP}}(\theta)$ a lower bound of the conservative policy iteration objective [17], $\hat{A}_t$ is the estimated advantage function which involves a learned value function $V_\omega(\cdot)$ to reduce the variance of estimation. In GCPN, $V_\omega(\cdot)$ is an MLP that maps the graph embedding computed according to Section 3.4.

Experiments

Experimental Setup

- ZINC250k molecule dataset
  - Contains over 250,000 drug like commercially available molecules
- RDKit OpenAI Molecule environment
  - To check validity of generated molecules
- GCPN setup
  - 3-layer defined GCPN
  - 64 dimensional node embedding in all hidden layers
  - Batch normalization after each layer

Baselines

- Junction Tree Variational Autoencoder (JT-VAE)
  - State-of-the-art algorithm for molecular generation
- Objective-Reinforced Generative Adversarial Networks (ORGAN)
  - State-of-the-art algorithm for molecular generation using RL and text-based representation of molecules

Tasks

1. Property Optimization
    a. Generate molecules while optimizing a score
2. Property Targeting
    a. Generate molecules within a targeted score
3. Constrained Property Optimization
    a. Generate new molecules on top of a given substructure while optimizing a score

Experiment Results

- Task 1: Property Optimization
  - Generating molecules with greatest penalized logP and QED scores
    - logP is the logarithm of the octanol-water partition coefficient
      - Score for how soluble a molecule is in water and fat
      - Penalized logP also takes into account ring size and synthetic accessibility
    - QED is an indicator of druglikeness from [0-1]

Task 1: Property Optimization Results

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

Experiment Results

- Task 2: Property Targeting
  - Set a targeted range of molecular weight (MW) and logP
  - Measure success rate and diversity
  - Success is the percentage of generated molecules that fall within the targeted range
  - Diversity is the average pairwise Tanimoto distance between the Morgan fingerprints of the molecules

Task 2: Property Targeting Results

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

Method	$-2.5 \leq \text{logP} \leq -2$		$5 \leq \text{logP} \leq 5.5$		$150 \leq \text{MW} \leq 200$		$500 \leq \text{MW} \leq 550$	
	Success	Diversity	Success	Diversity	Success	Diversity	Success	Diversity

Experiment Results

- Task 3: Constrained Property Optimization
  - Generate molecules that contain a base substructure/molecule (one of 800 ZINC molecules)
  - Base substructure/molecule has a low penalized logP so they measure the improvement of the penalized logP of the generated molecule
  - Penalized logP is being optimized
  - The baseline JT-VAE algorithm cannot constrain generated molecules to have a certain substructure. Therefore, they relaxed the constraint so the generated molecule must have a similarity score above a threshold δ.

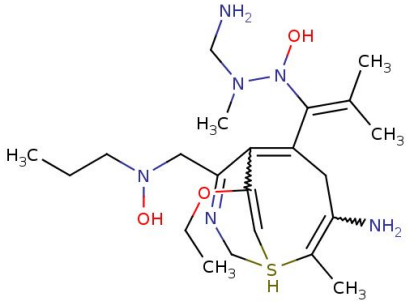
Task 3: Constrained Property Optimization

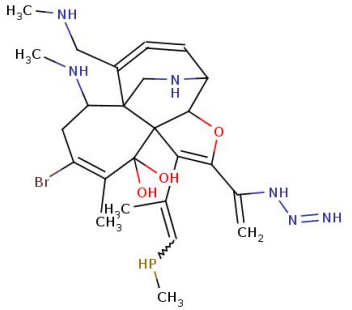
Table 3: Comparison of the performance in the constrained optimization task.

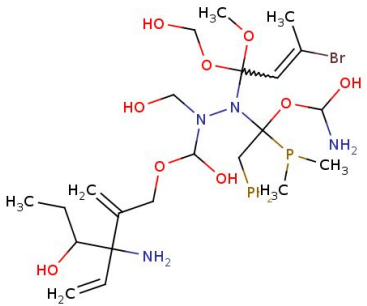
δ	JT-VAE			GCPN		
	Improvement	Similarity	Success	Improvement	Similarity	Success
0.0						
0.2						
0.4						
0.6						

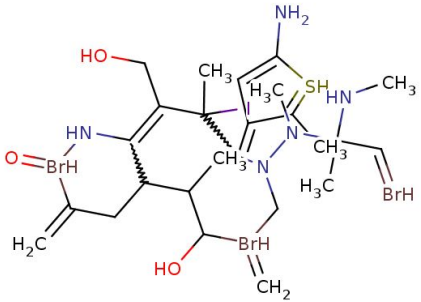
Demo

Example Molecules Generated









Discussion

Pros

Graph Representation

GCPN directly models molecules as molecular graphs.

- The text based representations such as SMILES (simplified molecular-input line-entry system) are vulnerable to even simple perturbation.
- Partially generated molecular graphs can be interpreted as substructures, whereas partially generated text representations might not be meaningful.

Reinforcement Learning

Better than generative model since

- It could model hard constraints and chemical properties which are not differentiable
- It allows active exploration outside datasets

Pros

Optimization:

This approach allows direct optimization of application-specific objectives, while ensuring that the generated molecules are realistic and satisfy chemical rules.

Performance:

In all tasks, GCPN achieves state-of-the-art results.

GCPN generates molecules with property scores 61% higher than the best baseline method, and outperforms the baseline models in the constrained optimization setting by 184% on average.

Cons

No guarantees of synthesizability

- Simply provides a chemically valid graph, but fails to have a documented synthesis path

Scaling

- Since GCPN operates iteratively, scaling graph generation may prove difficult since each action only contributes a single edge

Conclusions

Closing Remarks

- GCPN is a graph generation policy network using graph state representation and adversarial training, applied to the task of molecular graph generation
- SOTA in molecular property optimization and targeting, while maintaining integrity to realistic molecules.
- GCPN can be modified with other domain specific rewards to be applied to other fields

Q & A

Thanks for Listening!

Reference

RL, Proximal Policy Optimization (PPO) Explained.https://jonathan-hui.medium.com/rl-proximal-policy-optimization-ppo-explained-77f014ec3f12

Generative Adversarial Network. https://developers.google.com/machine-learning/gan/generator

Deep Reinforcement Learning: Pong from Pixels. http://karpathy.github.io/2016/05/31/rl/

Proximal Policy Optimization. https://openai.com/blog/openai-baselines-ppo/