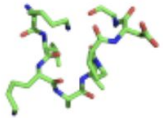
Tutorial: Graph generative models for drug discovery

https://github.com/hohyun312/molgen-tutorial/

RDKit
PyG
GNN
…

Molecular Representations

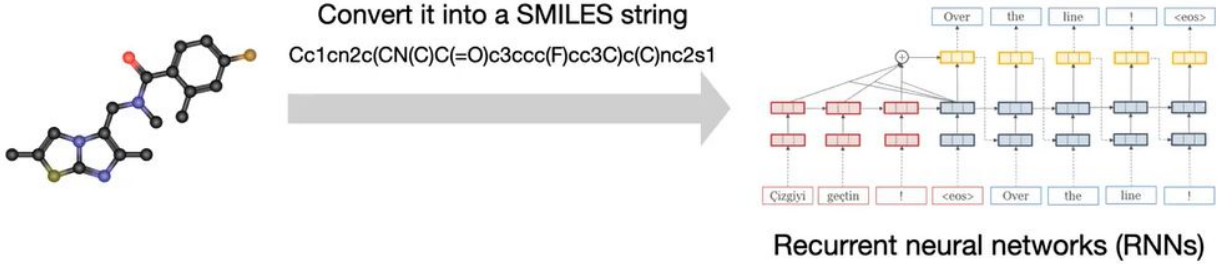
Table 1: Representation choice for molecules. Adding in 3D information consistently improves performance. The depicted 1D representations are the amino acid sequences and SMILES strings [Weininger, 1988] for proteins and small molecules, respectively.

Dimension	Representation	Examples	
		Proteins	Small Molecules
1D	linear sequence	KVKALPDA	CC(C)CC(C)NO
2D	chemical bond graph		
3D	atomistic geometry		

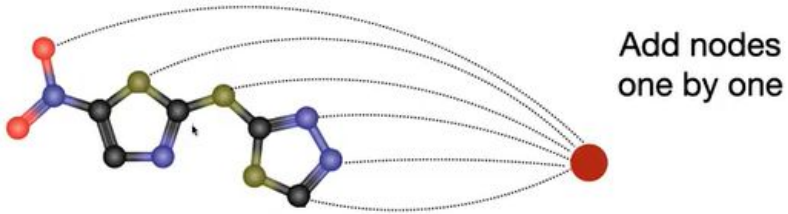
Townshend et al. ATOM3D: Tasks On Moleculs in Three Dimensions. NeurIPS. 2021.

Molecular Representations

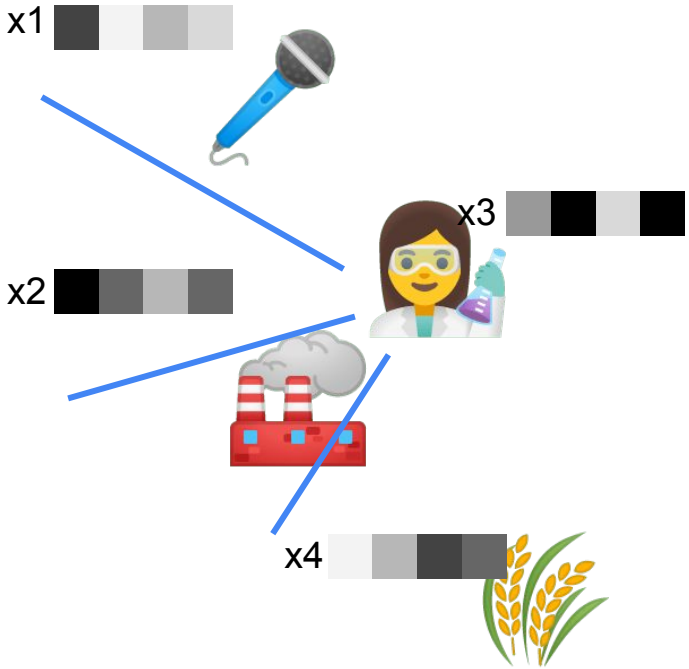
- String-based approach



- Graph-based approach



Graph Neural Network

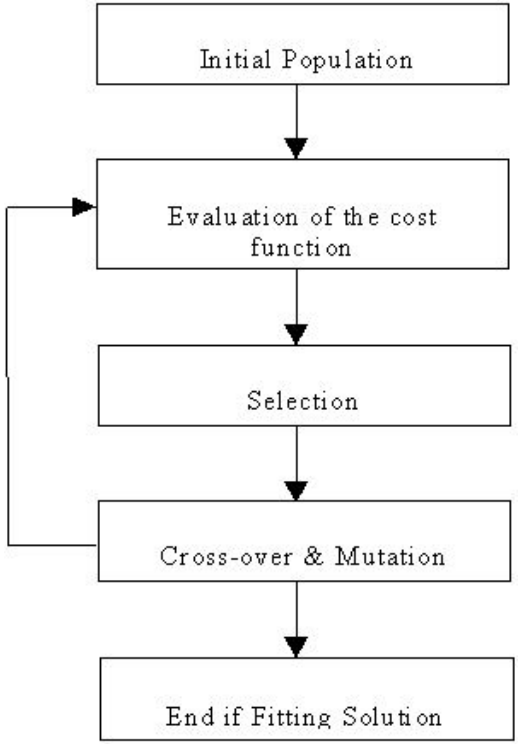


$$\mathbf{X} = \begin{matrix} x1 \\ x2 \\ x3 \\ x4 \end{matrix}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

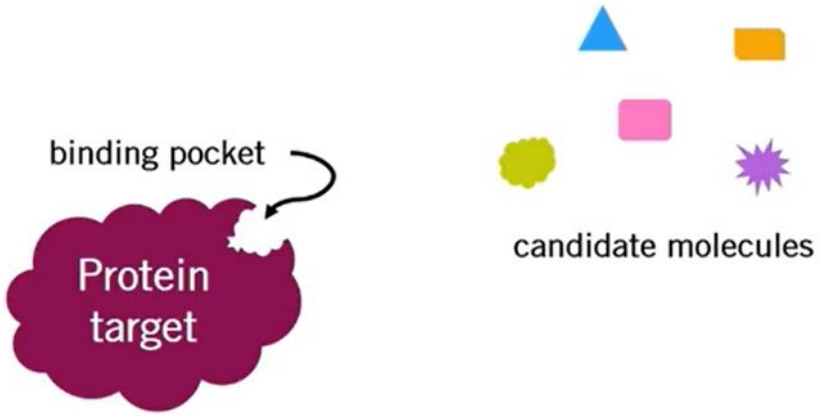
Genetic Algorithm



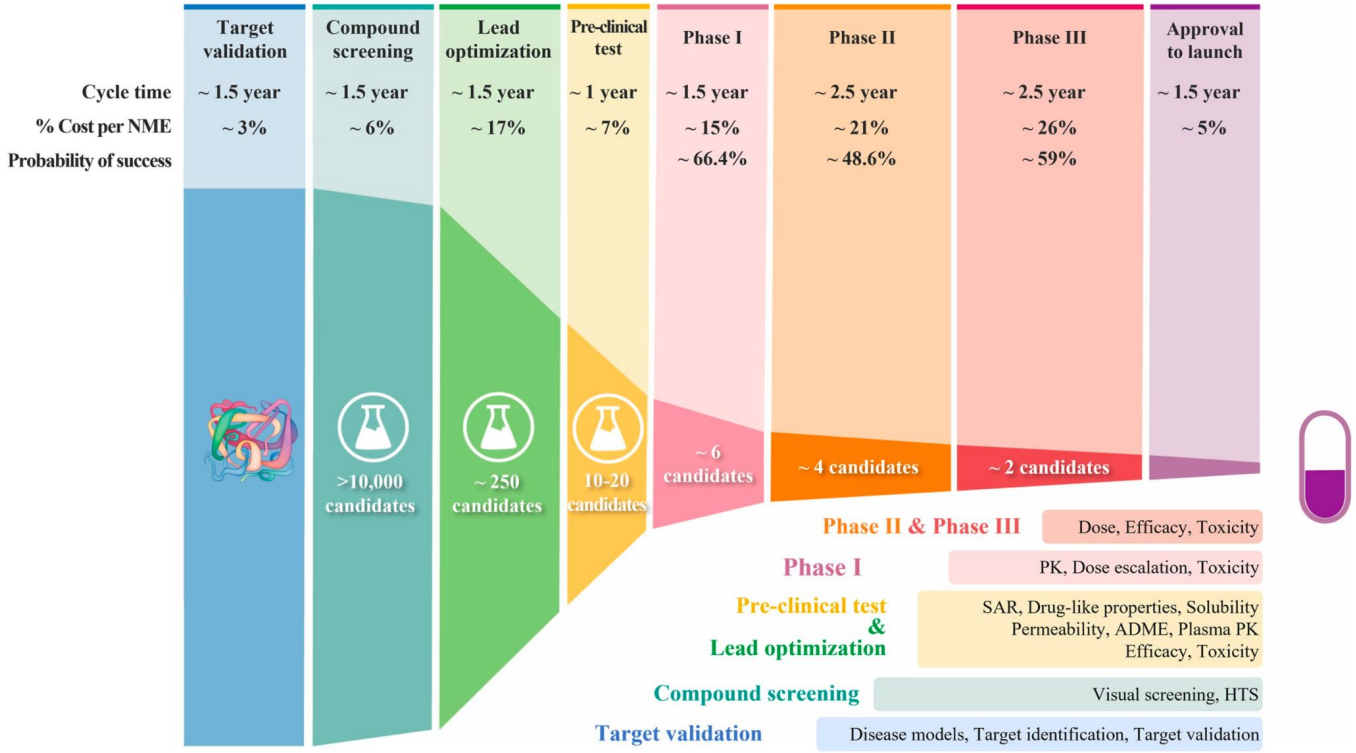
Drug Discovery

Goal: find molecules with desired properties

- Binding affinity
- Selectivity
- Solubility
- Toxicity
- Synthetic accessibility
- ...

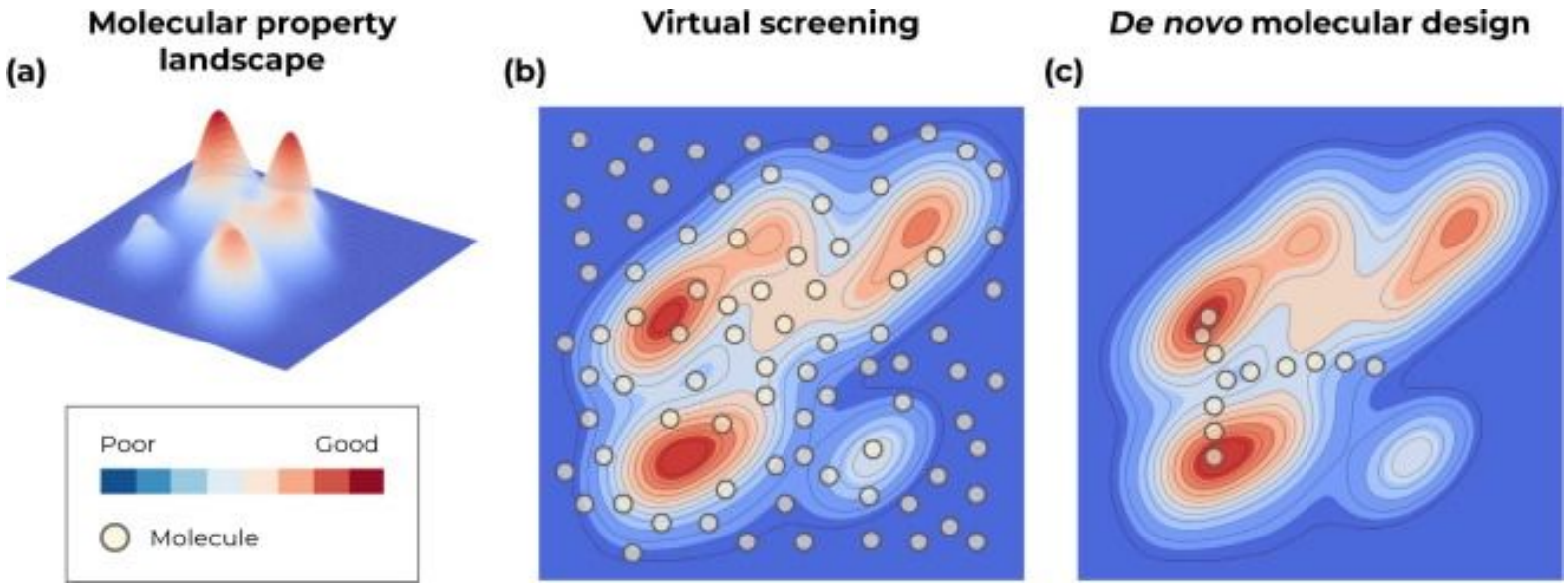


Drug Discovery



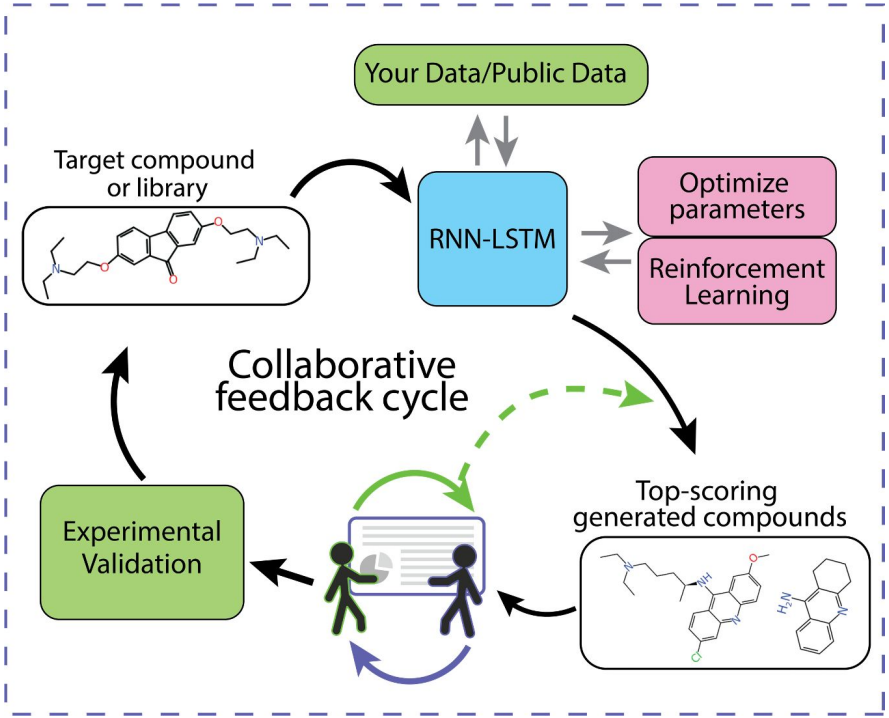
Source: https://www.sciencedirect.com/science/article/pii/S2211383522000521

Drug Discovery



Source: https://www.sciencedirect.com/science/article/pii/S1359644621002531

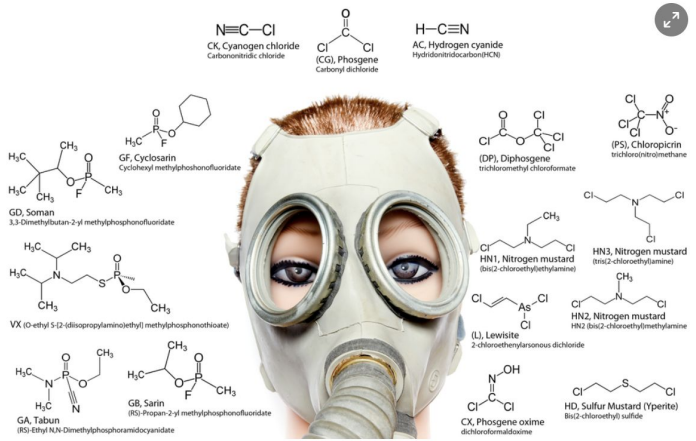
Drug Discovery



의약품 개발 AI, 독성 물질 개발 AI로 변신

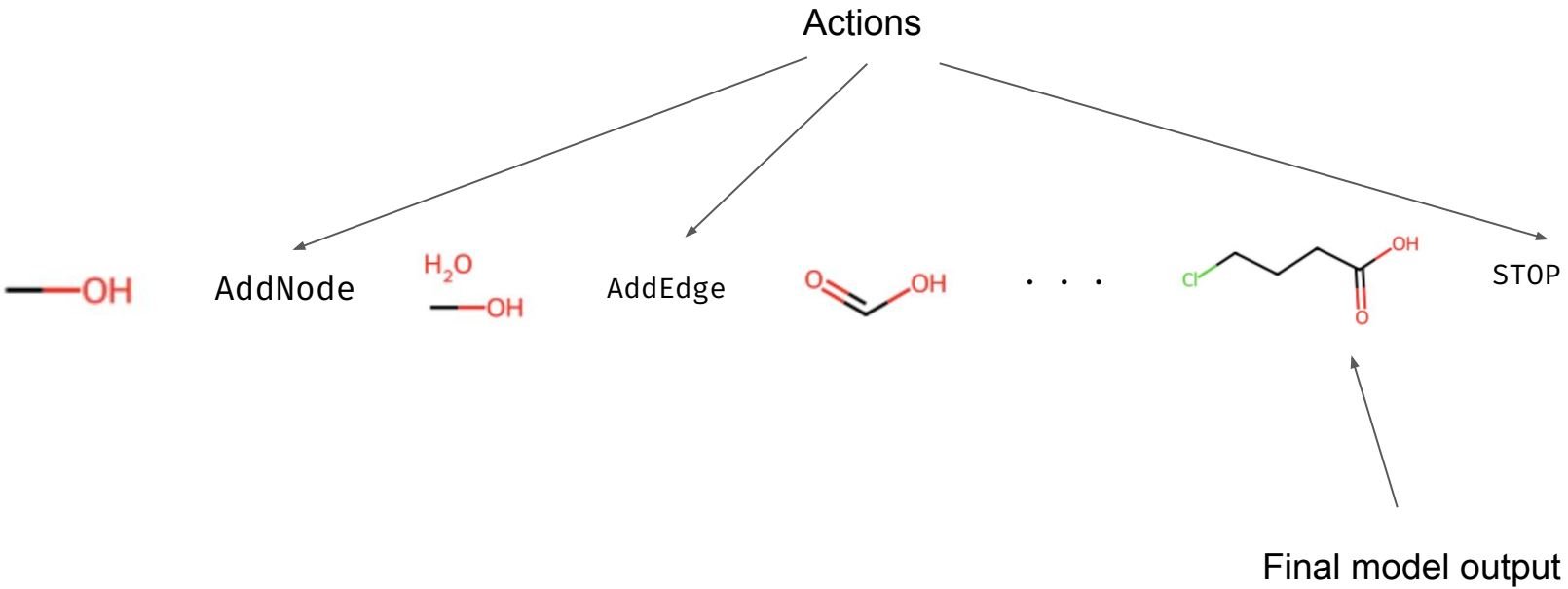
박찬 위원 입력 2022.03.22 17:10 댓글 0 좋아요 0

의약품 개발 AI를 독성물질 개발로 용도 변경
6시간 만에 40,000종의 독성 화학물질을 특정
약간의 지식으로 손 쉽게 독성 물질 제조 가능
AI의 오남용이 초래할 수 있는 위험성을 경고



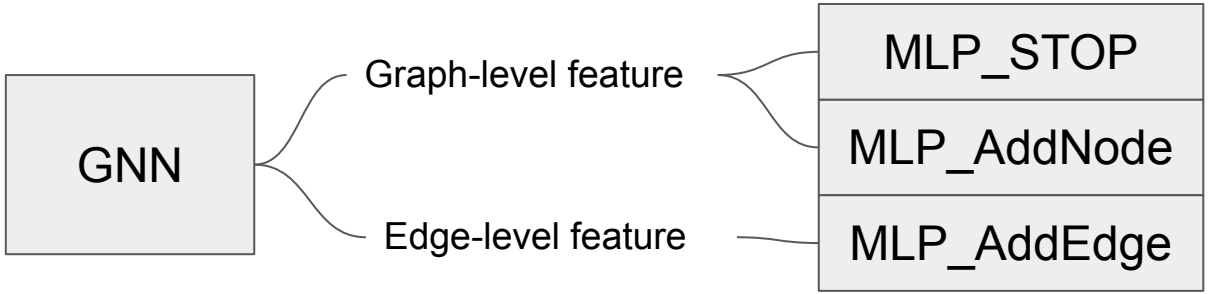
의약품 개발 AI의 용도를 변경해 인간에게 해가 되는 독성물질을 개발해줄 것을 주문했다.(사진=셔터스톡)

Graph Generation Process



Policy


State
Policy
Action


GNN
Graph-level feature
Edge-level feature
MLP_STOP
MLP_AddNode
MLP_AddEdge

Score Function Estimator

- Also called "score function estimator" or "likelihood-ratio estimator"

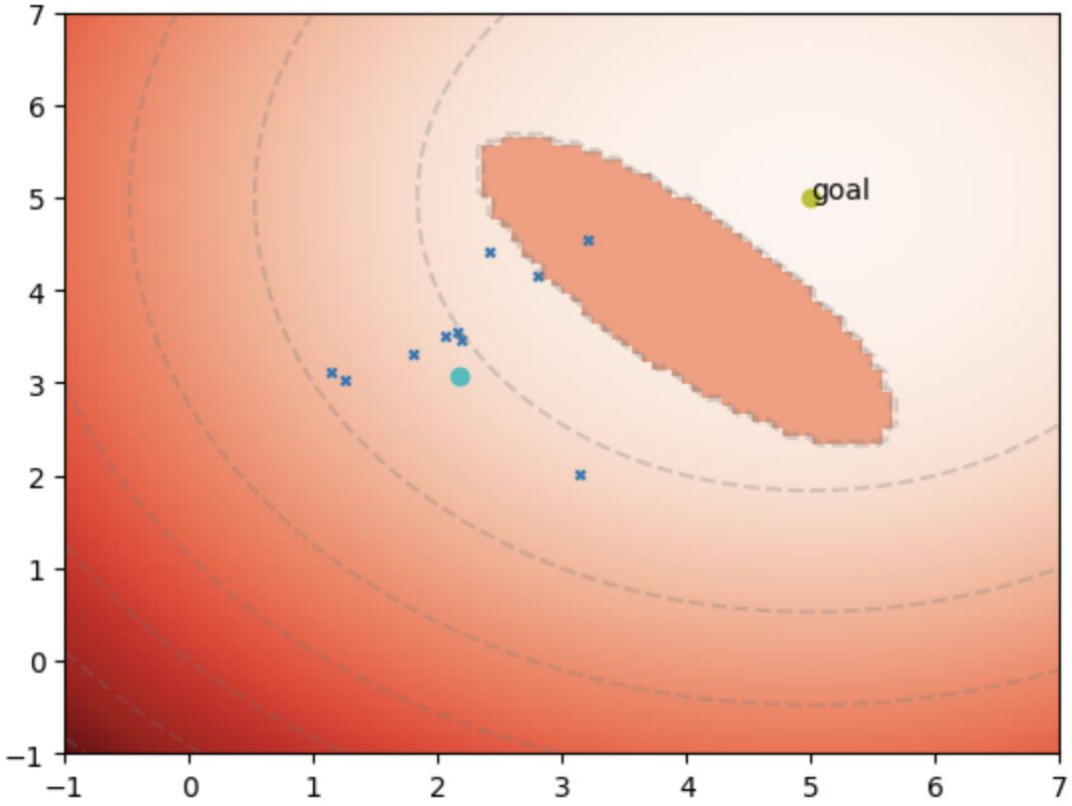
$$
\begin{aligned}
\nabla_\theta E_x[f(x)] &= \nabla_\theta \sum_x p(x) f(x) \\
&= \sum_x \nabla_\theta p(x) f(x) \\
&= \sum_x p(x) \frac{\nabla_\theta p(x)}{p(x)} f(x) \\
&= \sum_x p(x) \nabla_\theta \log p(x) f(x) \\
&= E_x[f(x) \nabla_\theta \log p(x)]
\end{aligned}
$$

```
loss = torch.mean(- log_probs * rewards)
optimizer.zero_grad()
loss.backward()
```

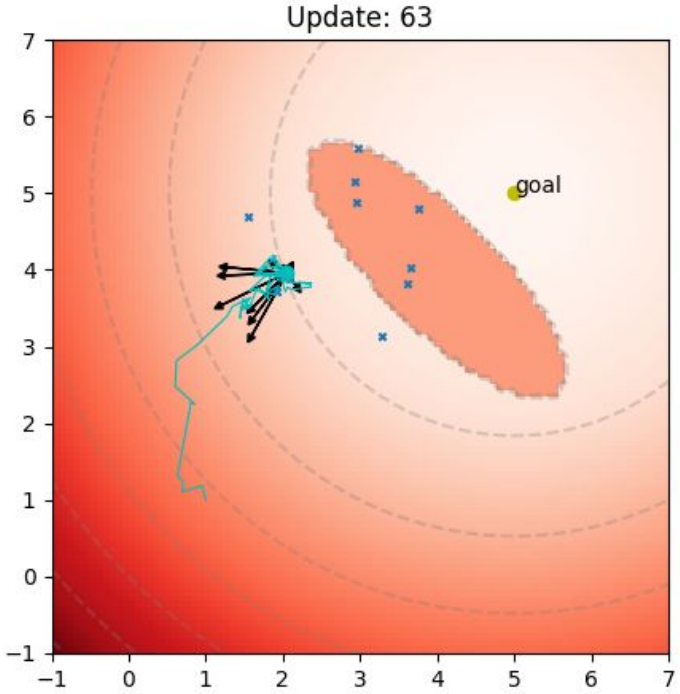
Score Function Estimator

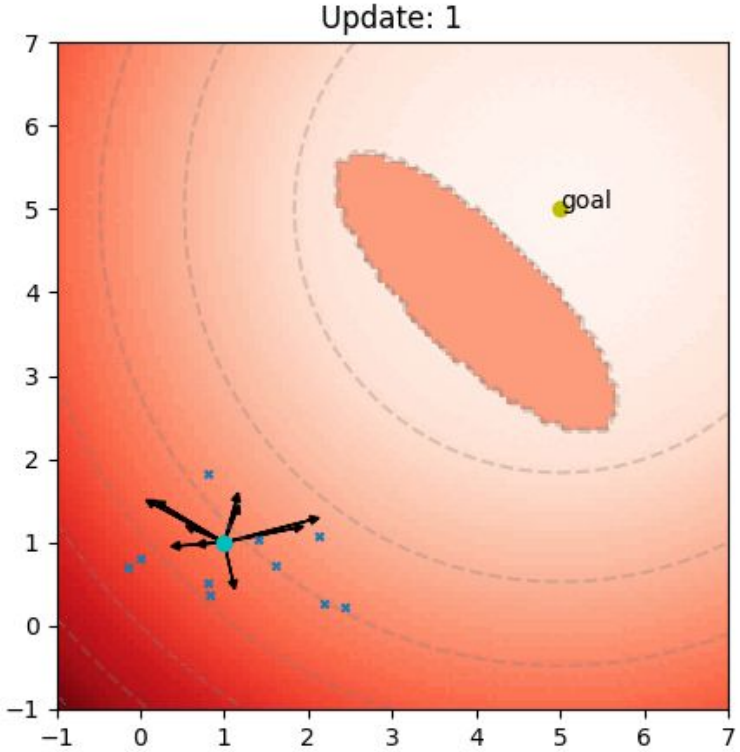
Learn $\mu \in \mathbb{R}^2$ to minimize the objective below to reach the goal.

$$\mathcal{L} = \mathbb{E}_{x \sim N(\mu, I)}[R(x)]$$



Score Function Estimator





Environment


```python
@dataclass
class State:
    node_types
    edge_types
    edge_list:

    num_node:
    num_edge:
```

```python
class ActionType(enum.Enum):
    AddNode = enum.auto()
    AddEdge = enum.auto()
    STOP = enum.auto()


@dataclass
class Action:
    type: ActionType = None
    source: int = None
    target: int = None
    node_type: NodeType = None
    edge_type: EdgeType = None
```

Environment


```python
def step(state: State, action: Action):
    next_state, done = deepcopy(state), False

    if action.type == ActionType.AddNode:
        next_state.add_node(action.node_type)

    elif action.type == ActionType.AddEdge:
        next_state.add_edge(action.source, action.target, action.edge_type)

    else:  # ActionType.STOP
        done = True

    return next_state, done
```

Environment