Learning Transferable Graph Exploration

Hanjun Dai, Yujia Li, Chenglong Wang, Rishabh Singh, Po-Sen Huang, Pushmeet Kohli

33rd Conference on Neural Information Processing Systems, Vancouver, Canada.

November 15, 2019
Goal: given an environment, efficiently reach as many distinct states as possible.

Examples:

- model checking: design test inputs to expose as many potential errors as possible
- active map building: construct a map of unknown environment efficiently
- exploration in reinforcement learning in general
High-level Idea: randomly choose states to visit / actions to take

Examples:

1. Random Walk on Graph [2]:
   - cover time (expected number of steps to reach every node) depends on graph structure
   - lower-bound on cover time: $O(n \log n)$; upper-bound: $O(n^3)$.

2. $\epsilon$-greedy Exploration:
   - select random action with probability $\epsilon$
   - prevents (to some extent) being locked onto suboptimal action

3. Learning to Prune: more on this later!
Common Approaches: Directed Exploration

**High-level Idea:** optimize objective that encourages exploration / coverage (usually some kind of “quantified uncertainty”)

**Examples:**

1. **UCB for Bandit Problems:**
   - in addition to maximizing the reward, encourage exploring unselected actions by the term \( \sqrt{\frac{\ln t}{N_t(a)}} \)

2. **Intrinsic Motivations in RL:**
   - pseudo-count (similar to UCB): rewards change in state density estimates
   - information gain: take actions from which you learn about the environment (reduces entropy)
   - predictive error: encourage actions that lead to unpredictable outcome (for instance unseen states)

Reference: Sergey Levine’s Deep Reinforcement Learning Course 2017, Lecture 13
Exploration on Graphs

- goal is to efficiently reach as many vertices as possible
- effectiveness of random walk greatly depends on the graph structure

Motivation: given the distribution of graphs in training time, can the algorithm learn efficient covering strategy [1]?
Problem Setup

**Environment:** Graph-structured state-space

- at time $t$, the agent observes a graph $G_{t-1} = \{E_{t-1}, V_{t-1}\}$, and a coverage mask $c_{t-1} : V_{t-1} \rightarrow \{0, 1\}$ indicating the nodes explored so far
- the agent takes an action $a_t$ and receives a new graph $G_t$
- number of steps / actions can be seen as budget for exploration (to be minimized)

**Goal of Learning:**

- learn exploration strategy such that given an unseen environment (from the same distribution as training environment), the agent can efficiently visit as many unique states as possible
Defining the Reward

Maximize the number of visited nodes:

\[
\max_{\{a_1, a_2, \ldots, a_t\}} \sum_{v \in V_t} \frac{c_t(v)}{|V|}; \text{ equivalently, } r_t = \sum_{v \in V_t} \frac{c_t(v)}{|V|} - \sum_{v \in V_{t-1}} \frac{c_{t-1}(v)}{|V|}.
\]

Objective:

\[
\max_{\{\theta_1, \theta_2, \ldots, \theta_t\}} \mathbb{E}_{G \sim \mathcal{D}} \left[ \sum_{t=1}^{T} \mathbb{E}_{a_t^G \sim \pi(a|h_t^G, \theta_t)} \left[ r_t^G \right] \right],
\]

- \(h_t = \{(a_i, G_i, c_i)\}_{i=1}^{t}\) is the exploration history
- \(\pi(a|h_t, \theta_t)\) is an action policy at time \(t\) parameterized by \(\theta_t\)
- \(\mathcal{D}\) is the distribution of environments

Agent trained with the advantage actor critic algorithm (A2C) [3]
Representing the Graph:

- use graph neural networks to learn a representation $g : (G, c) \rightarrow \mathbb{R}^d$ (node features are concatenated with the one-bit information $c_t$)
- starting from node $\mu_v^{(0)}$, update representation via message passing: $\mu_v^{(l+1)} = f(\mu_v^{(l)}, \{e_{uv}, \mu_u^{(l)}\}_{u \in \mathcal{N}(v)})$, where $\mathcal{N}$ is the neighbor nodes of $v$ and $f(\cdot)$ is parameterized by MLP
- apply attention weighted-sum to aggregate node embedding
- graph representation learned via unsupervised link prediction
Representing the Exploration History (continued)

Representing the History (graph external memory):

- summarize representation up to the current step via auto-regressive aggregation parameterized as
  \[ F(h_t) = \text{LSTM}(F(h_{t-1}, g(G_t, c_t))). \]
Toy Problem: Erdos-Renyi Random Graph

- blue node indicates starting point; darker colors represent more visit counts
- the proposed algorithm explores the graph more efficiently
Toy Problem: 2D Maze

- given fixed budget ($T = 36$), the agent is trained to traverse the 6x6 maze as much as possible
- test on held-out mazes from the same distribution
• data generated by program synthesizer
• learned exploration strategy is comparable or better than expert-designed heuristic algorithm
Limitation and Future Directions

Limitation:

- cannot scale to large programs
- requires reasonable large amount of training data

Possible Extensions:

- reuse computation for efficient representation
- RL-based approximation for other NP-complete problems
