Multi-Path Policy Optimization

Ling Pan¹, Qingpeng Cai², Longbo Huang¹

¹ IIS, Tsinghua University
² Alibaba Group
Policy-based Methods in Reinforcement Learning

• Vanilla policy gradient methods
  • May incur large policy changes → performance collapse due to unlimited updates

• Trust-region Policy Optimization\[1\] & Proximal Policy Optimization\[2\]
  • Pros: stable performance
  • Cons: suffer from insufficient exploration

On-policy
Exploration

• Deep reinforcement learning (DRL)
  • Count-based methods [3]
  • Intrinsic motivation [4]
  • Noisy networks [5]
  • Diversity regularization
    • Limits the applicability to trust-region methods

• Evolutionary algorithms (EA) [6]
  • Population-based methods

- Sensitive parameters
- Learn complex structures
- Better exploration ability
- Better stability
- Sample-inefficient
Exploration

• Combining DRL and EA
  • Existing approaches [7, 8, 9]
    • Maintain a population of agents
    • Each of them interact with the environment to collect experiences
    • Store the diverse experiences in a shared experience replay buffer

• Problem
  • Focus on off-policy algorithms
  • Sample-inefficient to apply to on-policy learning

➤ Rollout all policies in the population at each iteration can be inefficient as each policy cannot exploit other policies’ experiences

➤ Each interaction with the environment can be expensive

How to achieve efficient exploration for on-policy algorithms?
Motivating Example

• Environment: 21x21 maze with sparse rewards
  • Reward: +1 when the agent reaches G, and 0 otherwise

• Baselines
  • TRPO: vanilla single-path TRPO
  • Multi-TRPO: simply trains an ensemble of policies and picks the best one
  • MP-TRPO: our MPPO method applied to TRPO

• Performance comparison
  • The agent can fail to reach the goal state under TRPO.
  • The agent can also fail to consistently find the destination that simply trains the policy ensemble and choosing the best (Multi-TRPO).
  • MP-TRPO can always successfully reach the destination after 0.6 million steps.
Motivating Example

• State visitations

- TRPO: The agent can only explore a very limited area in the maze and mainly stays in the left side.
- Multi-TRPO: The agent is able to search a larger region, but still mostly re-explores the left part.
- MP-TRPO: The agent is capable to bypass the wall and explore both sides of the maze.
Multi-Path Policy Optimization

• Method

Main components:
1. Pick and rollout
2. Value function approximation
3. Policy optimization
4. Policy buffer update
Multi-Path Policy Optimization

• Initialization

- $K$ different policies randomly initialized in the policy buffer
  \[ \pi_t = (\pi_{1t}, \ldots, \pi_{Kt}) \]
- A shared value network $V$
Multi-Path Policy Optimization

- **Iteration**: Pick
  - Unnecessary to rollout all policies in the population
  - Pick a candidate policy from the current policy buffer
  - Picking rule: choose the policy $\pi_{it}$ with highest score $f_i$
  - $\forall k, f_k(\pi_t) = (1 - \alpha)\hat{J}_k(\pi_t) + \alpha\hat{H}_k(\pi_t)$
  
  - Pick the policy with good performance and being explorative simultaneously
  
  - $\alpha$ trades-off between exploration and exploitation
Multi-Path Policy Optimization

• Iteration: Pick
  • Theoretical guarantee for MP-TRPO

1. Pick the same policy
2. Pick a more explorative but worse policy

**Theorem 3.1.** Let $i, j$ denote the indexes of policies that are picked at timestep $t$, $t+1$, respectively. Denote the improvement of $J_i(\pi_{t+1})$ over $J_i(\pi_t)$ as $\sigma_t$. Then, the following bound holds for $0 \leq \alpha < 1$:

$$J_j(\pi_{t+1}) - J_i(\pi_t) \geq \frac{\alpha}{1-\alpha} [\max_k J_k(\pi_{t+1}) - \min_k J_k(\pi_{t+1})] + \sigma_t.$$
Multi-Path Policy Optimization

- Iteration: Rollout
  - Only the picked candidate policy interact with the environment by rollouts to collect samples
Multi-Path Policy Optimization

- Iteration: Value network update
  - The shared value network is updated according to samples collected by the candidate policy to approximate the value function
  \[
  \frac{1}{N} \sum_{n=1}^{N} (r_n + \gamma V_{\phi_t}(s_{n+1}) - V_{\phi_t}(s_n))^2
  \]
  - Exploits diverse samples collected by the candidate policy at each iteration
  - Can better estimate the value function
  - Provides more information for \( A \) to distinguish good/bad actions
Multi-Path Policy Optimization

- **Iteration:** Policy optimization
  - Only the candidate policy is optimized with a base policy optimization method
    - Collected samples
    - Shared value network

\[
\frac{1}{N} \sum_{n=1}^{N} \nabla \theta_{it} \log \pi_{\theta_{it}}(a_n | s_n) A_{\pi_{it}}(s_n, a_n)
\]

- With a better value estimation, policy optimization drives each picked policy to explore previously unseen good states and actions
Multi-Path Policy Optimization

- Iteration: Policy buffer update
  - Common way: replace the worst policy
    - Quickly loses the diversity
    - Leads to a low exploration level
  - Update strategy: replace the candidate policy by itself
    \[ \pi_{t+1} = (\pi_{1t}, ..., \pi_{it}', ..., \pi_{Kt}) \]
    - Maintain the diversity
Multi-Path Policy Optimization

Summary

- Maintains $K$ policy paths
- Provides diverse samples for $V$
  - Enable better value estimation
- The updated picked policy by policy optimization are more able to collect trajectories with higher rewards
  - Provide better guidance for the picked policy
- The picking rule does not incur much computational cost
- The policy buffer update strategy ensures diversity of the policy buffer
Experiments

• Ablation study
  • The effect of the number of paths $K$
    - $K$ trades off the diversity of the policy buffer and sample efficiency
  • The effect of the weight $\alpha$
    - $\alpha$ controls the trade-off between exploration and exploitation
Experiments

• Performance comparison

MP-TRPO outperforms single-path policy optimization and state-of-the-art exploration methods in terms of sample efficiency and final performance.
MPPO is readily applicable given a baseline on-policy reinforcement learning algorithm to encourage exploration

Figure 8: Performance comparison of MP-PPO.
Conclusion

• We propose a novel methodology MPPO utilizing a population of policies to tackle the exploration bottleneck of on-policy RL algorithms.
• We give a theoretical guarantee of stable performance of MP-TRPO.
• MPPO can be readily applied given any baseline on-policy algorithm.
• We validate MPPO to two popular on-policy algorithms, TRPO and PPO, and conduct extensive evaluation on MuJoCo tasks. Results show that MPPO outperforms state-of-the-art exploration methods.


Thank you!