Goal: Sample efficient exploration in RL

How can we get reinforcement learning algorithms to explore efficiently when operating in complex environments? Such algorithms would be immensely useful in scaling RL into high stakes scenarios where sample-efficiency is a primary concern. A possible solution is through representation learning, where we discover some simple underlying structure that enables us to efficiently explore and approximate operating in complex environments? Such algorithms would be immensely useful in scaling RL into high stakes scenarios where sample-efficiency is a primary concern.

Question 1: What does it mean to have a good representation?

Question 2: How do we learn one in a sample-efficient manner, while exploring?

We answer these questions in the context of low rank MDPs.

Main result

Assume access to function class \(\Phi, Y\) such that \(\phi \in \Phi, y \in Y\)

Assume computational oracle for optimizing and sampling from \(\Phi, Y\)

Theorem [AKKS20]: FLAMBE learns a low rank MDP model such that

\[
\mathbb{E}_\pi \left[ \phi_{\pi}(s, a, \mu) \right] = \max_{\phi} \mathbb{E}_\pi \left[ \phi_{\pi}(s, a, \mu) \right] \leq \epsilon
\]

With sample complexity:

\[
p \approx \gamma(d, |A|, H, \frac{1}{\epsilon^2} \log(\Phi|Y|/\delta))
\]

FLAMBE runs in polynomial time in oracle model.

Corollary 1: For any reward, near-optimal policy and Q function are linear in \(\phi_{\pi, y}\)

Corollary 2: Can optimize any reward function with no additional experience

Corollary 3: Simpler planner for stochastic factorization, with a much better sample complexity.

Discussion

1. Provably RL with general non-linear function approximation

2. Suggestions for practice: reward bonuses, model architecture, etc.

3. Future work: does it work in practice?

References


