Explicit Explore-Exploit Algorithms in Continuous State Spaces
Mikael Henaff
Microsoft Research

Introduction

- Many RL algorithms have high sample complexity
- This can limit their practical application
- Explicit Explore-Exploit (E³): First provably efficient RL algorithm [1]
- Sample complexity polynomial in |S|
- Model-based:
  - Use current model to plan to reach unknown states
  - Once model is sufficiently accurate, optimize rewards
- Assumes small discrete state space

E³ with continuous states

- Instead of maintaining set of states, maintain set of dynamics models \( \mathcal{M}_i \)
- Treat states which induce high disagreement between models in \( \mathcal{M}_i \) as unknown
- Execute policies which induce high disagreement
- No dependence on number of states

Algorithm

Model class \( \mathcal{M} \), policy class \( \Pi \)
\[ D(\pi, M, M') = \| P_{M}^h(\cdot) - P_{M'}^h(\cdot) \|_{TV} \]

\[ v_{\text{exploit}}(\pi, M) = \max_{\pi' \in \Pi} \sum_{t=1}^{h} D(\pi, M, \pi', t) \]

Algorithm 1 \((\mathcal{M}, \Pi, n, \epsilon, \phi)\)

\[ \mathcal{M}_1 \leftarrow \mathcal{M} \]
Initialize replay buffer \( \mathcal{R} \leftarrow \emptyset \).
for \( t = 1, 2, \ldots \) do

\[ \pi_{\text{exploit}} = \text{argmax}_{\pi \in \Pi} v_{\text{exploit}}(\pi, \mathcal{M}_1) \]
if \( v_{\text{exploit}}(\pi_{\text{exploit}}, \mathcal{M}_1) > \frac{1}{\epsilon} \) then
Add \( n \) trajectories following \( \pi_{\text{exploit}} \) to \( \mathcal{R} \).
\[ \mathcal{M}_{t+1} \leftarrow \text{UpdateModelSet}(\mathcal{M}_t, \mathcal{R}, \phi) \]
else
Choose any \( \tilde{M} \in \mathcal{M}_t \)
\[ \pi_{\text{exploit}} = \text{argmax}_{\pi \in \Pi} v_{\text{exploit}}(\pi, \tilde{M}) \]
Halt and return \( \pi_{\text{exploit}} \)
end if
end for

Analysis

Use rank of model misfit matrices as complexity measure:
\[ d = \max_k \text{rank}(A_k) \]

- Bounded by |S|
- Bounded by rank of transition matrix
- Bounded by # parameters in factored MDPs

Theorem

Assuming that \( M^* \in \mathcal{M} \), for any \( \epsilon, \delta \in (0, 1] \) and appropriate \( n, \phi \), with probability at least \( 1 - \delta \), Algorithm 1 outputs a policy \( \pi_{\text{exploit}} \) such that \( v_{\text{exploit}}(\pi_{\text{exploit}}, \mathcal{M}_t) \geq v_{\text{exploit}}(\pi, \mathcal{M}_t) \). The number of trajectories collected is at most \( O\left(\frac{|S|^2}{\epsilon^2} \log \left(\frac{|S|}{d}m^2\right)\right) \), where \( d \) is the rank of the misfit matrix.

Proof sketch (simplified, errors are 0/1):
- High disagreement between \( M, M' \) implies at least one must have high error
- At any iteration, there exists a model with high error or all models give a good exploitation policy
- At iteration \( t \), row \( \pi_t \) of \( A_t \) linearly independent of rows of previous \( \pi_{t+1} \) at most \( \text{rank}(A_t) \leq d \) iterations

Experiments

Environments tested (require exploration):
- Stochastic combination lock
- Mazes
- Classic Control

References


Links

• Code: https://github.com/mhenaff/neural-e3
• Contact: mhenaff@microsoft.com

Idealized Algorithm
- \( \mathcal{M}_t = \{ \text{all } M \in \mathcal{M} \text{ with error less than } \phi \text{ on } \mathcal{R} \} \)
- Planning: exact search over \( \Pi \)

Practical Algorithm
- \( \mathcal{M}_t = \{ \text{ensemble of models which minimize error on } \mathcal{R} \} \)
- Planning: continuous version of BFS or MCTS