Online Exploration in Least-Squares Policy Iteration

Lihong Li, Michael L. Littman, and Christopher R. Mansley

Rutgers Laboratory for Real-Life Reinforcement Learning (RL³)
Contributions

Reinforcement Learning

Challenge I
Exploration/Exploitation Tradeoff

Rmax [Brafman & Tenneholtz 02]
(provably efficient, finite)

Challenge II
Value-Function Approximation

LSPI [Lagoudakis & Parr 03]
(continuous, offline)

LSPI-Rmax
Outline

• Introduction
  – LSPI
  – Rmax
• LSPI-Rmax
• Experiments
• Conclusions
Basic Terminology

• Markov decision process
  – States: $S$
  – Actions: $A$
  – Reward function: $-1 \leq R(s,a) \leq 1$
  – Transition probabilities: $T(s'|s,a)$
  – Discount factor: $0 < \gamma < 1$

• Optimal value function: $Q^*(s,a)$
• Optimal policy: $\pi^*(s) = \arg \max_a Q^*(s,a)$
• Approximate $Q^*(s,a)$
Linear Function Approximation

\[ Q(s, a) = \sum_{i=1}^{k} w_i \phi_i(s, a) = w \cdot \phi(s, a) \]

• **Features:** \( \phi_i(s, a) \)
  – A.k.a. “basis functions”, and predefined

• **Weights:** \( w_i \)
  – Measures contributions of \( \phi_i \) to approximating \( Q^* \)

• **Learning** = finding \( w \) such that: \( w \cdot \phi(s, a) \approx Q^*(s, a) \)
LSPI [Lagoudakis & Parr 03]

Initialize $\pi$
Evaluate $\pi$: compute $w$
Improve $\pi$: $\pi'(s) = \text{argmax}_a w \cdot \phi(s,a)$

Given samples: $D = \{(s_1, a_1, r_1, s'_1), \ldots, (s_m, a_m, r_m, s'_m)\}$
Approx. Bellman Eqn.: $w \cdot \phi(s_i, a_i) \approx r_i + \gamma w \cdot \phi(s'_i, \pi(s'_i)), \ \forall i$
LSTDQ sets up a least-squares problem
and computes: $w = A^{-1}b$

$$A = \sum_{i=1}^{m} \phi(s_i, a_i)\left(\phi(s_i, a_i) - \gamma \phi(s'_i, \pi(s'_i))\right)^T, \quad b = \sum_{i=1}^{m} \phi(s_i, a_i)r_i$$
But, LSPI does not specify how to collect samples $D$: a fundamental challenge in online reinforcement learning.

An agent only collects samples in states it visits…

Given samples: $D = \{(s_1, a_1, r_1, s'_1), \ldots, (s_m, a_m, r_m, s'_m)\}$

Approx. Bellman Eqn.: $w \cdot \phi(s_i, a_i) \approx r_i + \gamma w \cdot \phi(s'_i, \pi(s'_i)), \ \forall i$

LSTDQ sets up a least-squares problem

and computes: $w = A^{-1}b$

$$A = \sum_{i=1}^{m} \phi(s_i, a_i)\left(\phi(s_i, a_i) - \gamma \phi(s'_i, \pi(s'_i))\right)^T, \quad b = \sum_{i=1}^{m} \phi(s_i, a_i)r_i$$
Exploration/Exploitation Tradeoff

- **Efficient exploration**
- **Inefficient exploration**

- **Optimal policy**

- Time vs. total rewards graph

- Node labels: 1, 2, 3, 98, 99, 100
Rmax [Brafman & Tenenholtz 02]

- Rmax is for finite-state, finite-action MDPs
- Learns $T$ and $R$ by counting/averaging
- In $s_t$, takes optimal action in $\hat{M}_{\text{Known}}$

\[ \hat{M}_{\text{Known}} = \langle S, A, \hat{T}, \hat{R}, \gamma \rangle \]

- "Optimism in the face of uncertainty"

\[ Q_{\text{max}} = \frac{1}{1 - \gamma} \]

- **Either:** explore "unknown" region
- **Or:** exploit "known" region

**Thm:** Rmax is provably efficient
LSPI-Rmax

• Similar to LSPI
• But distinguishes known/unknown \((s,a)\):

- Known state-actions
- Unknown state-actions
  - Treat their Q-value as \(Q_{\text{max}}\)
  - (Like Rmax)
- Modifications of LSTDQ

\[ S \times A \]

Samples in \(D\)
LSTDQ-Rmax

Given samples: \( D = \{(s_1, a_1, r_1, s'_1), \ldots, (s_m, a_m, r_m, s'_m)\} \)

Treat \( Q(s, a) = \frac{1}{1 - \gamma} \) if \((s, a)\) is unknown:

E.g., if \((s_i, a_i)\) is unknown,

change \((s_i, a_i, r_i, s'_i)\) to \((s_i, a_i, Q_{\max}, \Box)\) and

\[
A = \ldots + \phi(s_i, a_i)\phi(s_i, a_i)^T + \ldots
\]

\[
b = \ldots + \phi(s_i, a_i)Q_{\max} + \ldots
\]

Similarly for \((s'_i, a)\).
LSPI-Rmax for Online RL

- \( D = \text{empty set} \)
- Initialize \( w \)
- \textbf{for} \( t = 1, 2, 3, \ldots \)
  - Take greedy action: \( a_t = \arg\max_a w \cdot \phi(s_t, a) \)
  - \( D = D \cup \{(s_t, a_t, r_t, s_{t+1})\} \)
  - Run LSPI using LSTDQ-Rmax
Experiments

• Problems
  – MountainCar
  – Bicycle
  – Continuous Combination Lock
  – ExpressWorld (a variant of PuddleWorld)

Four actions
Stochastic transitions
Reward:
  -1 reward per step
  -0.5 reward per step in “expresslane”
  penalty for stepping into puddles
Random start states
Various Exploration Rules with LSPI

Cumulative Reward in ExpressWorld

- Converges to better policies
A Closer Look

States visited in the first 3 episodes:

- Inefficient exploration
- Efficient exploration

Help discovery of goal and express lane
More Experiments

Cumulative Reward in MountainCar

Rmax (m=7)
counter-based (m=7)
epsilon-greedy (eps=0.2)

Cumulative Reward in ContCombLock

Rmax (m=20)
counter-based (m=3)
epsilon-greedy (eps=0)

Cumulative Reward in Bicycle

Rmax (m=2)
epsilon-greedy (eps=0)
counter-based (m=2)
Effect of $R_{\text{max}}$ Threshold

![Graph showing the effect of $R_{\text{max}}$ threshold on average per-episode reward. The graph displays a decreasing trend in the average reward as the $R_{\text{max}}$ threshold increases.]
Conclusions

• We proposed LSPI-Rmax
  – LSPI + Rmax
  – encourages active exploration
  – with linear function approximation

• Future directions
  – Similar idea applied to Gaussian process RL
  – Comparison to model-based RL
Where are features from?

• Hand-crafted features
  – expert knowledge required
  – expensive and error prone

• Generic features
  – RBF, CMAC, polynomial, etc.
  – may not always work well

• Automatic feature selection using
  – Bellman error [Parr et al. 07]
  – spectral graph analysis [Mahadevan & Maggioni 07]
  – TD approximation [Li & Williams & Balakrishnan 09]
  – $L_1$ Regularization for LSPI [Kolter & Ng 09]
LSPI-Rmax vs. MBRL

• Model-based RL (e.g., Rmax)
  – Learns an MDP model
  – Computes policy with the approximate model
  – Can use function approx. in model learning
    • Rmax w/ many compact representations [Li 09]

• LSPI-Rmax is model-free RL
  – Avoids expensive “planning” step
  – Has weaker theoretical guarantees