

# Active Reinforcement Learning

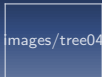
---

images/tree04

**Aijun Bai**

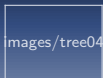
Sep 24, 2011

Multi-Agent Systems Lab., USTC



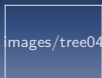
# Outline

- 1 **Preliminaries**
  - Markov Decision Process
  - Reinforcement Learning
- 2 **Active Reinforcement Learning**
  - Active Learning Fashion
  - Sensitivity Analysis
  - Active RL Algorithm
- 3 **Empirical Test**
- 4 **Conclusions**
  - Advantages & Disadvantages
  - Possible Applications in 2D



# Outline

- 1 **Preliminaries**
  - Markov Decision Process
  - Reinforcement Learning
- 2 **Active Reinforcement Learning**
  - Active Learning Fashion
  - Sensitivity Analysis
  - Active RL Algorithm
- 3 **Empirical Test**
- 4 **Conclusions**
  - Advantages & Disadvantages
  - Possible Applications in 2D



## Definition of MDPs

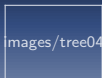
- MDP is a 4-tuple:
  - Finite set of states:  $S = \{s_1, s_2, \dots, s_{|S|}\}$
  - Finite set of actions:  $A = \{a_1, a_2, \dots, a_{|A|}\}$
  - Transition probability function:  $T(s'|s, a) \rightarrow [0, 1]$
  - Immediate reward function:  $R(s, a) \rightarrow \mathbf{R}$
- Define  $Next(s, a) = \{s' : T(s'|s, a) > 0\}$
- Deterministic policy:  $\pi(s) \rightarrow A$

# Properties of policy $\pi$

- Transition probability matrix:  $T^\pi(s, s') = T(s'|s, \pi(s))$
- Reward vector:  $R^\pi(s) = R(s, \pi(s))$
- Value function(vector):  $V^\pi(s) = E \{ r_1 + \alpha r_2 + \alpha^2 r_3 + \dots \}$
- Bellman equation:  $V^\pi = \alpha T^\pi V^\pi + R^\pi$
- Compute  $V^\pi(T, R)$  (Policy evaluation):
  - Direct method:  $V^\pi = (I - \alpha T^\pi)^{-1} R^\pi$
  - Iterative method:  $V^\pi(s) \leftarrow R^\pi(s) + \alpha T^\pi(s) V^\pi$
- Utility function:  $U^\pi(T, R) = E_{s_0 \sim D} V^\pi(s_0; T, R)$

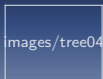
# The optimal policy $\Pi_{T,R}$

- $\Pi_{T,R}$  respects:  $\Pi_{T,R} = \arg \max_{\pi} U^{\pi}(T, R)$
- Can be solved by Dynamic Programming method:
  - Value iteration
  - Policy iteration
- Only if  $T$  and  $R$  is known



# Model-free fashion using RL

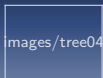
- Find  $\Pi$  when  $T$  or  $R$  is not known
- General processes:
  - 1 Sample Collection: exploration & exploitation
  - 2 Policy Evaluation:  $V \leftarrow V^\pi$
  - 3 Policy Improvement:  $\pi \leftarrow greedy(V)$
- Interact eachother until convergence of policy or it's value function
- Can be proved to be optimal when convergence happens



# Sample Collection

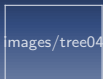
- Samples are sets of  $s, a, r, s'$
- Sampling methods:
  - Interaction with realistic environment
  - Simulation episodes from start state
  - Query a single state/action pair with simulation





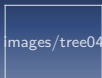
# Outline

- 1 Preliminaries
  - Markov Decision Process
  - Reinforcement Learning
- 2 **Active Reinforcement Learning**
  - Active Learning Fashion
  - Sensitivity Analysis
  - Active RL Algorithm
- 3 Empirical Test
- 4 Conclusions
  - Advantages & Disadvantages
  - Possible Applications in 2D



## Learning with prior knowledge

- Observations:
  - Traditional RL method (MC, TD, LSPI, etc.) learns from 0
  - Near-optimal policy can be got from inaccurate models
- Idea: using prior MDP specification to accelerate learning
- Method: active sample collection with sensitivity analysis of individual state/action pair



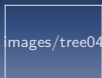
## Replacement Function

- Replace the transition probabilities  $T(\cdot|\hat{s}, \hat{a})$  of a fixed state/action pair with a given distribution  $X \in \Delta(\text{Next}(\hat{s}, \hat{a}))$ :

$$W_{\hat{s}, \hat{a}}[T, X](s'|s, a) = \begin{cases} X(s') & \text{if } s, a = \hat{s}, \hat{a} \\ T(s'|s, a) & \text{otherwise} \end{cases}$$

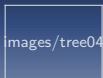
- Similarly, replace the reward of the chosen state/action pair with  $r$ :

$$Y_{\hat{s}, \hat{a}}[R, r](s, a) = \begin{cases} r & \text{if } s, a = \hat{s}, \hat{a} \\ R(s, a) & \text{otherwise} \end{cases}$$



## The Goal of Sensitivity Analysis

- To determine how much  $T(\cdot|\hat{s}, \hat{a})$  can change before the currently optimal policy becomes suboptimal
- More precisely:
  - Let  $\Pi_{T;\hat{s},\hat{a}} = \arg \max_{\pi} U^{\pi}(W_{\hat{s},\hat{a}}[T_0, T], R_0)$
  - Region in  $\Delta(\text{Next}(\hat{s}, \hat{a}))$  space:  
$$C = \{T : U^{\Pi_{T_0;\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, T], R_0) \geq U^{\Pi_{T;\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, T], R_0)\}$$
  - The smaller the region, the more sensitive the station/action pair

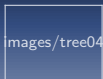


## Taylor's Approximation

- Approximate  $U_{T_1}^\pi(W_{\hat{s}, \hat{a}}[T_0, X])$  around  $T_1$  with Taylor's approximation method:

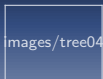
$$\begin{aligned}\hat{U}_{T_1}^\pi(W_{\hat{s}, \hat{a}}[T_0, X]) &\approx U^\pi(W_{\hat{s}, \hat{a}}[T_0, T_1], R_0) \\ &\quad + \nabla_{X|\hat{s}, \hat{a}} U^\pi(W_{\hat{s}, \hat{a}}[T_0, T_1], R_0)(X|\hat{s}, \hat{a} - T_1)\end{aligned}$$

- The gradient of the utility function can be computed from a derived MDP



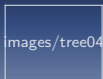
# Newton's method for sensitivity analysis

- 1 Solve optimal policy  $\Pi_{T_0; \hat{s}, \hat{a}}$  for  $T_0$
- 2 Approximate  $\hat{U}_{T_0}^{\Pi_{T_0; \hat{s}, \hat{a}}} (W_{\hat{s}, \hat{a}}[T_0, X])$  around  $T_0$
- 3 Select the starting point  $T'_0 \in \Delta(\text{Next}(\hat{s}, \hat{a}))$  and  $i \leftarrow 0$
- 4 Solve optimal policy  $\Pi_{T'_i; \hat{s}, \hat{a}}$  for  $T'_i$
- 5 Approximate  $\hat{U}_{T'_i}^{\Pi_{T'_i; \hat{s}, \hat{a}}} (W_{\hat{s}, \hat{a}}[T_0, X])$  around  $T'_i$
- 6 Let  $T'_{i+1}$  be the closest to  $T_0$  intersection point of  $\hat{U}_{T_0}^{\Pi_{T_0; \hat{s}, \hat{a}}} (W_{\hat{s}, \hat{a}}[T_0, X])$  and  $\hat{U}_{T'_i}^{\Pi_{T'_i; \hat{s}, \hat{a}}} (W_{\hat{s}, \hat{a}}[T_0, X])$
- 7 Let  $i \leftarrow i + 1$  and repeat steps 4-7 until  $\Pi_{T'_i; \hat{s}, \hat{a}} = \Pi_{T_0; \hat{s}, \hat{a}}$
- 8 Return  $\| T'_i | \hat{s}, \hat{a} - T_0 | \hat{s}, \hat{a} \|$



## General steps for Active RL Algorithm

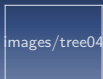
- 1 Specify possibly inaccurate  $T$  &  $R$  of the environment offline
- 2 Analyse sensitivity of individual state/action pair
- 3 Sample state/action pairs based on their sensitivity ordering
- 4 Find the optimal policy in the “corrected” MDP



## Convergence and Complexity

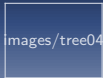
- Convergence can be reached in logarithmic time in MDP which  $|Next(\cdot, \cdot)| \leq 2$
- DAG-structured MDP can be converted in polynomial time into an MDP of above form
- No strong convergence results for arbitrary MDPs
- Works well based on empirical results





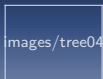
# Outline

- 1 Preliminaries
  - Markov Decision Process
  - Reinforcement Learning
- 2 Active Reinforcement Learning
  - Active Learning Fashion
  - Sensitivity Analysis
  - Active RL Algorithm
- 3 Empirical Test
- 4 Conclusions
  - Advantages & Disadvantages
  - Possible Applications in 2D



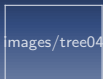
## Experiment Setup

- Test domains:
  - Mountain-Car
  - Cart-Pole
  - Windy Gridworld
  - Pizza Delivery
  - Drunkard's Walk
- Test method:
  - Provide initial description as  $T$
  - Sensitivity analysis and sort state/action pair
  - Randomly perturb  $T$  to  $T'$
  - Sample one state/action for 10000 times
  - Estimate  $T'$  using maximum likelihood methods
  - Find the optimal policy of the “corrected” MDP
  - Evaluate the policy in  $T'$



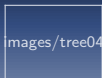
## Experiment Results

- Comparison with:
  - Random RL
  - Omniscient
  - Prior Optimal
  - Full-backups Q-learning
- Active RL outperforms Random RL
- Prior Optimal performs poorly
- Q-learning rarely catches up with Active RL



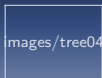
# Outline

- 1 Preliminaries
  - Markov Decision Process
  - Reinforcement Learning
- 2 Active Reinforcement Learning
  - Active Learning Fashion
  - Sensitivity Analysis
  - Active RL Algorithm
- 3 Empirical Test
- 4 Conclusions
  - Advantages & Disadvantages
  - Possible Applications in 2D



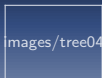
# Advantages & Disadvantages

- Advantages:
  - Use prior knowledge to accelerate learning
  - Provide a good strategy for sample collection avoiding blind sampling
- Disadvantages:
  - Explicitly maintaining of  $T$  need much memory with large state/action space
  - Low order Taylor's Approximation can not approximate well
  - Sensitivity calculation is complicated and time-consuming
  - Hardly to do "query" style sampling with real robots.



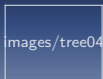
## Hetero Parameters Problem

- Players got hetero parameters determining the basis physical model
- Realistic hetero parameters specified randomly by server when game starts
- Prior optimal policy based on default hetero parameters may not perform well



## Solution with Active RL

- Offline:
  - Learning  $T_0$  for default hetero parameters  $H_0$
  - Sort state/action pairs based on sensitivity analysis
- Online:
  - Use  $T_0$  as prior MDP for realistic hetero parameters  $H_1$
  - Estimate  $T_1$  by sensitivity based samples collection
  - Find the optimal policy of  $T_1$  for online using



## References

- Arkady Epshteyn, **Active Reinforcement Learning**,  
<http://ai.stanford.edu/~acvogel/papers/290.pdf>