Reinformcement Learning Theory Learning

Yue Wang

November 20, 2016

Abstract

a simple note for RL theroy

1 Introduction

RL theory is very long history and these days

- Bullet point one
- Bullet point two

1. Numbered list item one
2. Numbered list item two

2 Notation and Formatting

some notations:

$\mathcal{S}$	the state space
$\mathcal{A}$	the action space
s_t	the state at time t, actual
s	the state
r_t	the reward at time t, actual
$r(s,a,s')$	the reward from stat s take action to stat s';
$v_k(s)$	the value of stat s at k iteration
v_k	the value function at time k iteration
$p(s,a,s')$	the probability of transfer to s' when given the current stat s and action a
$p(\pi,s,a,s')$	the probability of transfer to s' when given the current stat s and policy π
π	the policy
$\pi(s,a)$	the probability of take action a given the current stat s under policy π

3 RL algorithms

4 RL convergence theory

4.1 counterexample

There are many that shows the RL algorithms may not convergence even divergence under some conditions

In [Sutton and Barto, 2011, chap 11.3] the authors give an intuitive conclusion about when these algorithms will divergence :

> *The danger of instability and divergence arises whenever we combine three things:*
>
> 1. *training on a distribution of trainsition other than that naturally generated by the process whose expectation is being estimated(e.g. off-policy learning)*
> 2. *scalable function approximation (e.g. linear semi-gradient)*
> 3. *bootstrapping (eg, DP,TD learning)*

There are many counter example, follows are some of that.

4.1.1 counterexample1

In [Baird, 1995] the author gives an example to show that the TD(0) algorithms with linear function approximation will diverge.

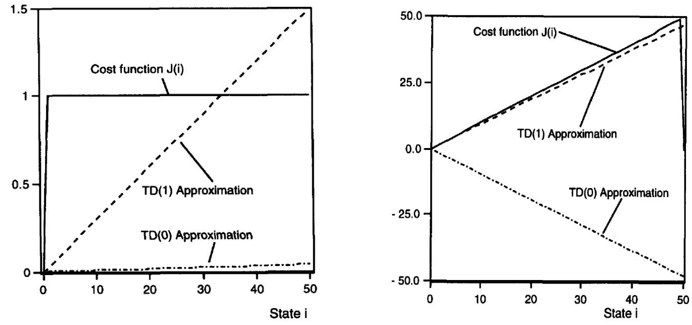

Figure 1: Bertsekas' counterexample

4.1.2 conberexample2

In [Bertsekas, 1995] the author gives an example to show that the **TD(λ) algorithm with linear function approximation** convergence to a very poor approximation to the cost function.

As showed in 1

4.1.3 counterexample3

In Tsitsiklis and Roy [1996] the author gives an example to show that the value iteration with linear function approximation use off policy may diverge

4.2 convergence for look up table

4.2.1 policy iteration with look up table

updata rule of policy iteration:

evaluation: $v_\pi = r(s,a,s') + \gamma \sum_{a,s'} P(\pi,s,a,s')v_\pi(s')$

improvement: $\pi_v(s) = \arg\max_a (r(s,a,s') + \gamma \sum_{s'} P(s,a,s')v_\pi(s'))$

The proof of policy iteration :

First, prove the monotonicity of policy improvement

Second, it is obvious that the number of policy is finite

4.2.2 value iteration with look up table

updata rule of value iteration:

1. $v_{k+1}(s) = \mathcal{T}(v_k)(s)$
2. $\mathcal{T}(v_k)(s) = \max_a [r(s,a,s') + \gamma \sum_{s'} P(s,a,s')v_k(s')]$

The proof of value iterationTsitsiklis and Roy [1996]:
$||v_{k+1} - v^*||_\infty = ||\mathcal{T}(V_k) - \mathcal{T}(v^*)_\infty|| \le \gamma ||v_k - v^*||_\infty \le \cdots \le \gamma^{k+1}||v_0 - v^*||_\infty \to 0$

4.2.3 Q-learning with look up table

updata rule of Q-learning:

- $q_{k+1} = (1-\alpha)q_k(s_t,a_t) + \alpha \max_b [r_t + \gamma q_k(s_{t+1}, b)]$

4.2.4 SARSA iteration with look up table

updata rule of Q-learning:

- $q_{k+1} = (1-\alpha)q_k(s_t,a_t) + \alpha [r_t + \gamma q_k(s_{t+1}, a_{t+1})]$

4.3 convergence for linear function approximation

4.3.1 linear function approximation with prediction problem

4.3.2 linear function approximation with control problem

4.4 convergence for nonlinear function approximation

4.4.1 nonlinear function approximation with prediction problem

4.4.2 nonlinear function approximation with prediction problem

References

Leemon Baird. Residual Algorithms: Reinforcement Learning with Function Approximation. *Icml*, pages 30–37, 1995. ISSN 1098-6596. doi: 10.1017/CBO9781107415324.004. URL `http://kirk.usafa.af.mil/{~}baird`.

Dimitri P Bertsekas. A counterexample to temporal differences learning. *Neural Computation*, 7(2):270–279, 1995.

Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction, 2011.

JN Tsitsiklis and B Van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 1996. URL `http://link.springer.com/article/10.1023/A:1018008221616`.