Introduction to Reinforcement Learning with Function Approximation

Norio Kosaka

January 26, 2019

Contents

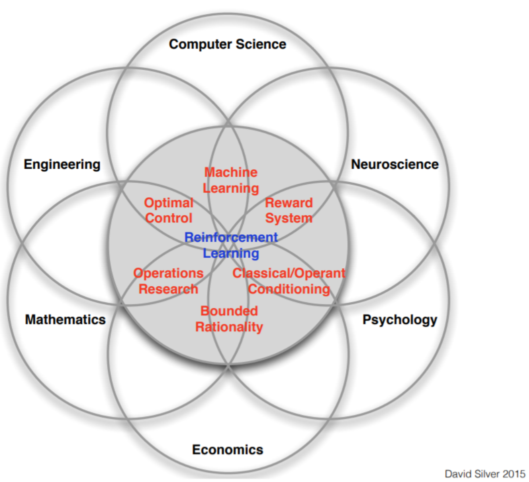
1 Foreword

This is the summary of the great presentation by Richard Sutton's presentation at NIPS 2015. And the original PPT is here:
`http://media.nips.cc/Conferences/2015/tutorialslides/SuttonIntroRL-nips-2015-tutorial.pdf`

2 Introduction to RL: Successes and Challenged

2.1 What is RL?

- Agent-oriented learning - learning by interacting with an environment achieve a goal
- Learning by trial and error, with only delayed evaluative feedback(reward)
- The beginnings of a science of mind that is neither natural science nor applications technology



2.2 Some RL Successes

- Learned the world's best player of Backgammon(Tesauro 1995 [14])
- Learned acrobatic helicopter autopilots(Ng, Abbeel, Coates et al., 2006+ [1])
- Widely used in the placement and selection of advertisements and pages on the web (e.g., A-B tests)
- Used to make strategic decisions in *Jeopardy!*(IBM's Watson 2011)

- Achieved human-level performance on Atari games from pixel-level visual input, in conjunction with deep learning (Google Deepmind 2015 [10], [3])

2.2.1 DQN

- Learned to play 49 games for the Atari 2600 game console, without labels or human input, from self-play and the score alone
- Learned to play better than all previous algorithms and at human level for more than half the games
- mapping raw screen pixels to predictions of final score for each of 18 joystick actions: Same learning algorithm applied to all 49 game without human tuning!!

3 The Formal Problem: Finite Markov Decision Processes

3.1 The Environment: A Finite Markov Decision Process(MDP)

- Proposed by Howard, 1964 [6]
- Discrete time: $t = 1, \cdots, T$
- A finite set of **states**
- A finite set of **actions**
- A finite set of **rewards**
- Life is a trajectory: $S_0, A_0, S_1, A_1, \cdots, S_t, A_t$
- with arbitrary Markov(stochastic, state-dependent) dynamics:

$$p(r, s'|s, a) = P[R_{t+1} = r, S_{t+1} = s'|S_t = s, A_t = a]$$

- **Policies**: deterministic policy: $a = \pi(s)$
- Agent learns to find the best policy which maximise the return
- **Action-value functions**: it says how good it is to be in a state, take an action, and thereafter follow a policy

$$q(s, a) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots |S_t = s, A_t = a, A_{t+1:\infty} \sim \pi]$$

- **Optimal Policies**(π^*): maximise the action-value function:

$$q_{\pi^*}(s, a) = \max_{\pi} q_\pi(s, a) = q_*(s, a)$$

$$\pi^*(s) = \arg\max_a q_*(s, a)$$

4 Exact Solution Methods (tabular methods)

4.1 Q-Learning, the simplest model-free RL algorithm

It is introduced by Watkins&Dayan 1992 [16]

- Initialise an array $Q(s,a)$ arbitrarily
- Choose actions in any way, perhaps based on Q, such that all actions are taken in all states
- On each time step, change one element of the array

$$\Delta Q(s,a) = \alpha\Big(R_{t+1} + \gamma \max_a Q(S_{t+1},a) - Q(S_t,A_t)\Big)$$

- If desired, reduce the step-size parameter α over time

4.2 Policy Improvement Theorem

Given the value function for any policy π: $q_\pi(s,a)\ \forall s,a$, it can always be greedified to obtain a better policy

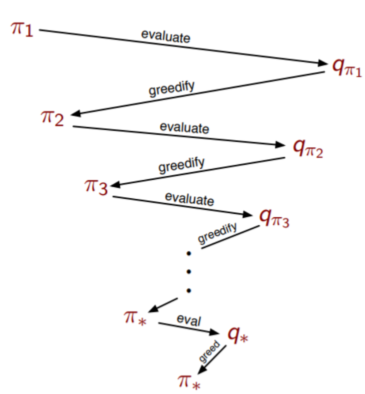
$$\pi'(s) = \arg\max_a q_\pi(s,a)$$

where better means

$$q_{\pi'}(s,a) \geq q_\pi(s,a)\ \forall s,a$$

4.3 Policy Iteration

- Policy Evaluation
- Policy Improvement



4.4 Exploration/Exploitation Dilemma

- You can't do the action that you think is best all the time
- You can't explore all the time
- Thus you must both explore and exploit, but neither to excess. What is the right balance?

4.5 Bootstrapping

Bootstrapping is the key idea underlying both **dynamic programming (DP)** and all **temporal-difference (TD)** learning. It updates an estimate from an estimate, it is like a guess from a guess. The concept itself is derived from the **Bellman Expectation Equation**:

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}\Big[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s, A_t = a, A_{t+1:\infty} \sim \pi\Big] \\ &= \mathbb{E}\Big[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1} | S_t = s, A_t = a, A_{t+1} \sim \pi\Big] \end{aligned}$$

Or the Bellman optimality equation:

$$q_*(s,a) = \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') | S_t = s, A_t = a\Big]$$

4.6 Summary

Off-policy learning is learning about the value of a policy other than the policy being used to generate the trajectory. Q-learning learns about the value of its deterministic greedy policy—which gradually become optimal—from data while behaving in a more exploratory manner. Thus Q-learning is off-policy and this is essential to its strategy for escaping the **explore/exploit** dilemma

- the target policy is the policy being learned about
- the behavior policy is the policy generating the trajectory data
- on-policy learning is when the two policies are the same

5 Approximate Solution Methods (function approximation)

So far, we have confirmed that RL finds optimal policies for finite state-space environment, if the value functions and policies can be exactly represented in tables. But the real world is too large and complex for tables; continuous state-space. So, the question we will look at in this section is;

- Will RL work with approximations?
- Will RL work with function approximators?

5.1 Function Approximations

- Represent the action-value function by a **parameterised function approximator** with parameter θ

$$q(s, a, \theta) \approx q_*(s, a)$$

- The approximator could be a **deep neural network**, with the weights being the parameter

Q-leaning works well with function approximation(Watkins 1989 [17])

- Semi-gradient Q-learning(Watkins 1989 [17])

$$L(\theta) = \mathbb{E}\Big[\big(R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \theta) - \hat{q}(S_t, A_t, \theta)\big)^2\Big] \tag{1}$$

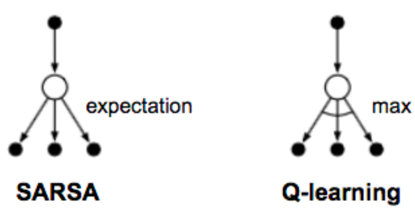
$$\Delta\theta_t = \alpha\Big(R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \theta) - \hat{q}(S_t, A_t, \theta)\Big)\frac{\partial \hat{q}(S_t, A_t, \theta_t)}{\partial \theta_t} \tag{2}$$

- Semi-gradient Sarsa(Rummery 1994 [11], Sutton 1988 [12])

$$L(\theta) = \mathbb{E}\Big[\big(R_{t+1} + \gamma \hat{q}(S_{t+1}, a, \theta) - \hat{q}(S_t, A_t, \theta)\big)^2\Big] \tag{3}$$

$$\Delta\theta_t = \alpha\Big(R_{t+1} + \gamma \hat{q}(S_{t+1}, a, \theta) - \hat{q}(S_t, A_t, \theta)\Big)\frac{\partial \hat{q}(S_t, A_t, \theta_t)}{\partial \theta_t} \tag{4}$$

Figure 1: Backup: SARSA VS Q-LEARNING



5.2 Problem of instability with semi-gradient Q-learning

As an on-policy method, Semi-gradient Sarsa has good convergence properties(Sutton 1988 [12], Dayan 1992 [4], Tsitsiklis&Van Roy 1997 [15]) But when it comes to Q-learning, the risk of divergence arises whenever we combine three things below;

- Function approximation: significantly generalizing from large numbers of examples

- Bootstrapping: learning value estimates from other value estimates, as in dynamic programming and temporal-difference learning. but it introduces **biases** in learning
- Off-policy learning: learning about a policy from data not due to that policy, as in Q-learning, where we learn about the greedy policy from data with a necessarily more exploratory policy

So, this problem is called **The deadly triad**.

5.3 Workaround for Deadly Triad

- Tsitsiklis&Van Roy 1997 [15] proposed: the degree of bootstrapping: λ, from $\lambda = 0$(full bootstrapping) to $\lambda = 1$(no bootstrapping)
- Double Q-learning, van Hasselt 2010 [5], but it is too soon to be sure
- least-squares methods like off-policy LSTD(λ), (Yu 2010 [18], Mahmood et al. 2015 [9]), but their computational costs scale with the square of the number of parameters
- Gradient-TD (Maei, 2011 [8]) and proximal-gradientTD (Mahadevan et al., 2015 [7]), These seem to me(R.Sutton) to be the best attempts to make TD methods with the robust convergence properties of stochastic gradient descent.
- Residual gradient methods (Baird 1999 [2])
- Emphatic-TD methods (Sutton, White&Mahmood 2015 [13], Yu 2015 [19])

6 Miscellany and closing remarks

6.1 Many dimensions of RL

6.1.1 Problems

- prediction vs control
- MDPs vs Bandits

6.1.2 Methods

- Tabular vs function approximation
- On-policy vs off-policy
- Bootstrapping vs Monte Carlo
- Model-based vs Model-free
- value-based vs policy-based

6.2 Policy-gradient actor-critic methods

- Policy is explicitly represented with its own parameters independent of any value function, so it is easy and simpler than dealing with the value function
- Policy parameters are updated by stochastic gradient ascent in a performance measure such as average reward per step
- A state-value function (critic) is optional but can significantly reduce variance
- Good convergence properties (on-policy)

References

[1] Pieter Abbeel, Adam Coates, and Andrew Y Ng. “Autonomous helicopter aerobatics through apprenticeship learning”. In: *The International Journal of Robotics Research* 29.13 (2010), pp. 1608–1639.

[2] Leemon Baird. “Residual algorithms: Reinforcement learning with function approximation”. In: *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 30–37.

[3] Marc G Bellemare et al. “The arcade learning environment: An evaluation platform for general agents”. In: *Journal of Artificial Intelligence Research* 47 (2013), pp. 253–279.

[4] Peter Dayan. “The convergence of TD (λ) for general λ”. In: *Machine learning* 8.3-4 (1992), pp. 341–362.

[5] Hado V Hasselt. “Double Q-learning”. In: *Advances in Neural Information Processing Systems*. 2010, pp. 2613–2621.

[6] Ronald A Howard. “Dynamic programming and Markov processes”. In: (1964).

[7] Bo Liu et al. “Finite-Sample Analysis of Proximal Gradient TD Algorithms.” In: *UAI*. Citeseer. 2015, pp. 504–513.

[8] Hamid Reza Maei and Richard S Sutton. *Gradient temporal-difference learning algorithms*. University of Alberta Edmonton, Alberta, 2011.

[9] Ashique Rupam Mahmood and Richard S Sutton. “Off-policy learning based on weighted importance sampling with linear computational complexity.” In: *UAI*. Citeseer. 2015, pp. 552–561.

[10] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (2015), p. 529.

[11] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*. Vol. 37. University of Cambridge, Department of Engineering Cambridge, England, 1994.

[12] Richard S Sutton. “Learning to predict by the methods of temporal differences”. In: *Machine learning* 3.1 (1988), pp. 9–44.

[13] Richard S Sutton, A Rupam Mahmood, and Martha White. “An emphatic approach to the problem of off-policy temporal-difference learning”. In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 2603–2631.

[14] Gerald Tesauro. “Temporal difference learning and TD-Gammon”. In: *Communications of the ACM* 38.3 (1995), pp. 58–68.

[15] JN Tsitsiklis and B Van Roy. *An analysis of temporal-difference learning with function approximationTechnical*. Tech. rep. Report LIDS-P-2322). Laboratory for Information and Decision Systems . . ., 1996.

[16] Christopher JCH Watkins and Peter Dayan. “Q-learning”. In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[17] Christopher John Cornish Hellaby Watkins. “Learning from delayed rewards”. PhD thesis. King’s College, Cambridge, 1989.

[18] Huizhen Yu. “Convergence of least squares temporal difference methods under general conditions”. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. Citeseer. 2010, pp. 1207–1214.

[19] Huizhen Yu. “On convergence of emphatic temporal-difference learning”. In: *Conference on Learning Theory*. 2015, pp. 1724–1751.