Tutorial of Reinforcement: A Special Focus on Q-Learning

TINGWU WANG,

MACHINE LEARNING GROUP,

UNIVERSITY OF TORONTO

Contents

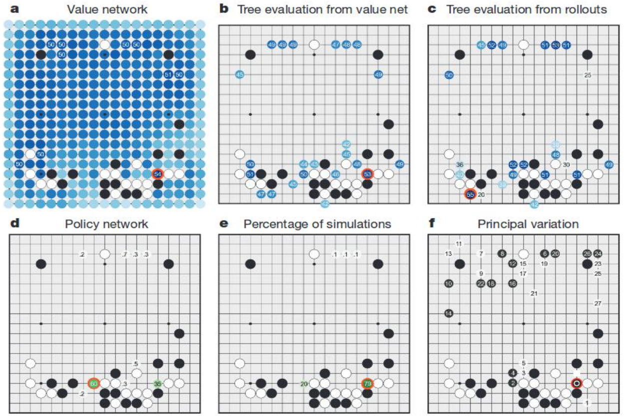
1. Introduction
   1. Discrete Domain vs. Continous Domain
   2. Model Based vs. Model Free
   3. Value-based vs. Policy-based
   4. On-policy vs. Off-policy
2. Prediction vs. Control: Marching Towards Q-learning
   1. Prediction: TD-learning and Bellman Equation
   2. Control: Bellman Optimality Equation and SARSA
   3. Control: Switching to Q-learning Algorithm
3. Misc: Continous Control
   1. Policy Based Algorithm
   2. NerveNet: Learning Stuctured Policy in RL
4. Reference

Introduction

1. Today's focus: Q-learning [1] method.
   1. Q-learning is a {
      discrete domain,
      value-based,
      off-policy,
      model-free,
      control,
      ~~**often shown up in ML finals**~~
      } algorithm.
2. Related to Q-learning [2]:
   1. Bellman-equation.
   2. TD-learning.
   3. SARSA algorithm.

Discrete Domain vs. Continous Domain

1. Discrete action space (**our focus**).
    1. Only several actions are available (e.g. up, down, left, right).
    2. Often solved by value based methods (DQN [3], or DQN + MCTS [4]).
    3. Policy based methods work too (TRPO[5] / PPO[6], not our focus).

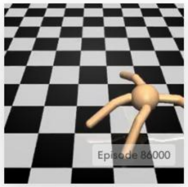


Discrete Domain vs. Continous Domain

1. Continuous action space (**not our focus**).
    1. Action is a value from a continous interval.
        1. **Infinite number of choices.**
        2. E.g.: Locomotion control of robots (MuJoCo [7]).

           Actions could be the forces applied to each joint (say: 0 - 100 N).
    2. If we apply discretization to the action space, we have discrete domain problems (autonomous car).

Walker2d-v1
Make a 2D robot walk.

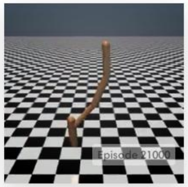


Ant-v1
Make a 3D four-legged robot walk.

Humanoid-v1
Make a 3D two-legged robot walk.

HalfCheetah-v1
Make a 2D cheetah robot run.

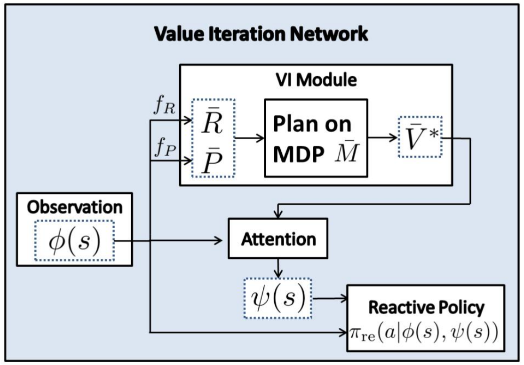
Swimmer-v1
Make a 2D robot swim.



Hopper-v1
Make a 2D robot hop.

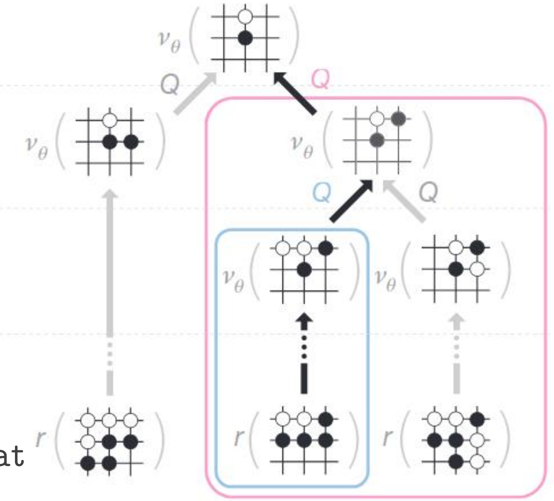
Model Based vs. Model Free

1. Model Based RL make use of dynamical model of the environment. (**not our focus**).
   1. Pros
      1. Better sample efficiency and transferabilty (VIN [8]).
      2. Security/performance gaurantee (if the model is good).
      3. Monte-Carlo Tree Search (used in AlphaGo[4]).
      4. …
   2. Cons
      1. The dynamical models are difficult to train itself.
      2. Time consuming.
      3. …



Model Based vs. Model Free

1. Model Free RL makes no assumption of the environments' dynamical model (**our focus**)
    1. In the ML community, more focus has been put on Model-free RL.
    2. E.g. :
        1. In Q-learning, we can choose our action by looking at Q(s, a), without worrying about what happens next.
        2. In AlphaGo, the authors combine the model-free method with model-based method (much stronger performance given a perfect dynamical model for Chess/GO).



Value-based vs. Policy-based

1. Value based methods are more interested in "Value" (**our focus**)
    1. Estimate the expected reward for different actions given the initial states (table from Silver's slides [9]).

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$;
    until S is terminal

Value-based vs. Policy-based

1. Policy-based methods directly model the policy (**not our focus**).

$$Q_\theta(s, a) = f(\phi(s, a), \theta) \longrightarrow \pi_\theta(s, a) = g(\phi(s, a), \theta)$$

   1. Objective function is the expected average reward.

$$J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s, a) \mathcal{R}_s^a$$

      1. Usually solved by policy gradient or evolutionary updates.

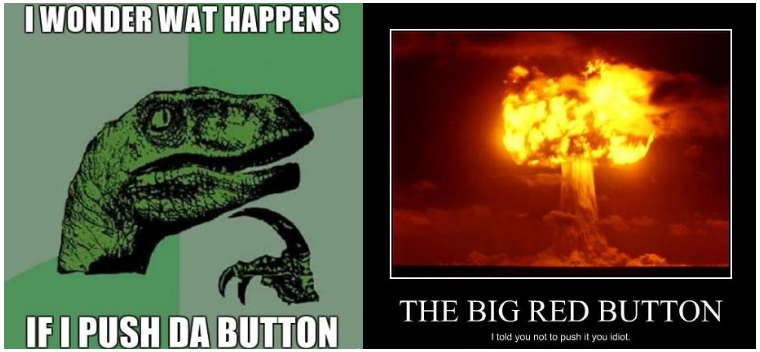
   2. If using value function to reduce variance --> actor-critic methods.

On-policy vs. Off-policy

1. Behavior policy & target policy.

   My own way of telling them (works most of the time):

   1. Behavior policy is the policy used to generate training data.
      1. Could be generated by other agents (learning by watching)
      2. Could be that the agent just want to do something new to explore the world.
      3. Re-use generated data.



   2. Target policy is the policy the agent want to use if the agent is put into testing.
   3. Behavior policy == target policy: On-policy, otherwise Off-policy

Contents

1. **Introduction**
   1. **Discrete Domain vs. Continous Domain**
   2. **Model Based vs. Model Free**
   3. **Value-based vs. Policy-based**
   4. **On-policy vs. Off-policy**
2. Prediction vs. Control: Marching Towards Q-learning
   1. Prediction: TD-learning and Bellman Equation
   2. Control: Bellman Optimality Equation and SARSA
   3. Control: Switching to Q-learning Algorithm
3. Misc: Continous Control
   1. Policy Based Algorithm
   2. NerveNet: Learning Stuctured Policy in RL
4. Reference

Prediction: TD-learning and Bellman Equation

1. Prediction:
   1. Evaluation certain policy (could be crappy).
   2. **Bellman Expectation Equation** (covered in lecture slides).

      $$q_\pi(s, a) = \mathbb{E}_\pi \left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a \right]$$

      Take out the Expectation if the process is deterministic.
   3. Algorithms:
      1. Monte-Carlo algorithm (**not our focus**).
         1. It learns directly from episodes of experience.
      2. Dynamic Programming (**not our focus**)
         1. Only applicable when the dynamical model is known and small.
      3. TD-learning algorithm (**related to Q-learning, covered in lecture slides**).
         1. Update value $V(S_t)$ toward estimated return $R_{t+1} + \gamma V(S_{t+1})$

            $$V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right)$$

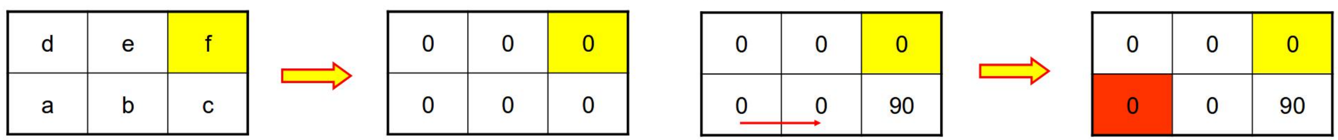
Prediction: TD-learning and Bellman Equation

1. Prediction Examples:

$$V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right)$$

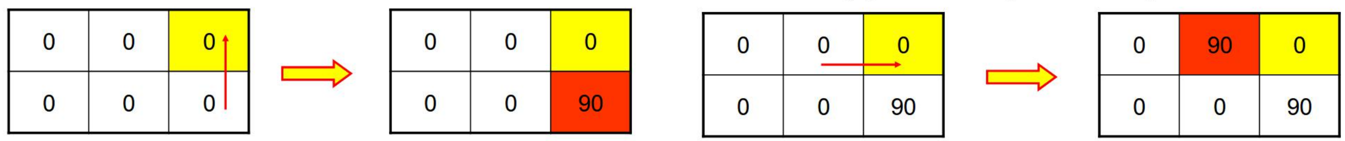
$$\alpha = 0.9 \quad \gamma = 1$$

$a \rightarrow b \quad V(a) \leftarrow 0 + 0.9[0 + 0 - 0] = 0$



$c \rightarrow f \quad V(c) \leftarrow 0 + 0.9[100 + 0 - 0] = 90$

$e \rightarrow f \quad V(e) \leftarrow 0 + 0.9[100 + 0 - 0] = 90$



2. Since the trajectory is generated by the policy we want to evaluate, eventually the value function converges to the true value under this policy.

Control: Bellman Optimality Equation and SARSA

1. Control:
   1. Obtaining the optimal policy.
      1. Looping over Bellman Expectation Equation and improve policy.
   2. **Bellman Optimality Equation** (covered in lecture slides).

      $$Q^*(s,a) = \mathbb{E}\left[r_{t+1} | s_t = s, a_t = a\right] + \gamma \mathop{\mathbb{E}}_{s_{t+1}}\left[\max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a\right]$$
   3. SARSA:
      1. Fix the policy to be epsilon-greedy policy from Bellman Optimality Equation.
      2. Updating the policy using Bellman Expectation Equation (TD).
      3. When the Bellman Expectation Equation converges, the Bellman Optimality Equation is met.

```
Initialize Q(s,a), ∀s ∈ S, a ∈ A(s), arbitrarily, and Q(terminal-state, ·) = 0
Repeat (for each episode):
   Initialize S
   Choose A from S using policy derived from Q (e.g., ε-greedy)
   Repeat (for each step of episode):
      Take action A, observe R, S'
      Choose A' from S' using policy derived from Q (e.g., ε-greedy)
      Q(S,A) ← Q(S,A) + α[R + γQ(S',A') − Q(S,A)]
      S ← S'; A ← A';
   until S is terminal
```

Control: Switching to Q-learning Algorithm

1. Switching to off-policy method.
    1. SARSA has the same target policy and behavior policy (epsilon-greedy).
    2. Q-learning might has different target policy and behavior policy.
        1. Target policy: greedy policy (Bellman Optimality Equation).
        2. Common behavior policy for Q-learning: Epsilon-greedy policy.
            1. Choose random policy with probability of epsilon, greedy policy with probability of (1 - epsilon)
            2. Decaying epsilon with time.

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\left[R + \gamma \max_a Q(S', a) - Q(S, A)\right]$
        $S \leftarrow S'$;
    until S is terminal

Contents

1. **Introduction**
   1. **Discrete Domain vs. Continous Domain**
   2. **Model Based vs. Model Free**
   3. **Value-based vs. Policy-based**
   4. **On-policy vs. Off-policy**
2. **Prediction vs. Control: Marching Towards Q-learning**
   1. **Prediction: TD-learning and Bellman Equation**
   2. **Control: Bellman Optimality Equation and SARSA**
   3. **Control: Switching to Q-learning Algorithm**
3. Misc: Continous Control
   1. Policy Based Algorithm
   2. NerveNet: Learning Stuctured Policy in RL
4. Reference

Policy Based Algorithm

1. Policy Gradient (**not our focus**)
   1. Objective function:

      $$J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s,a)\mathcal{R}_s^a$$

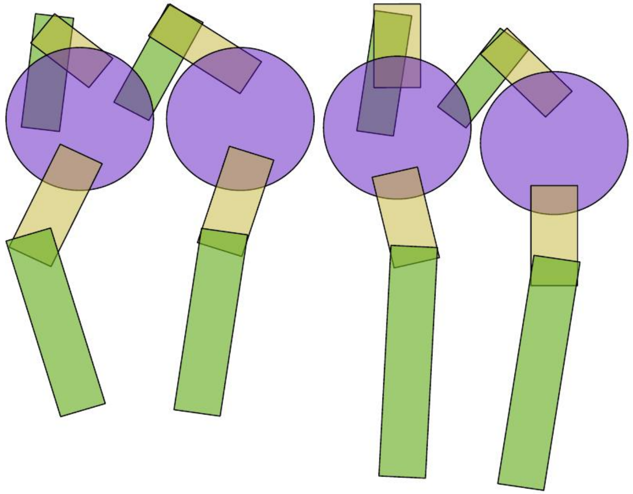
   2. Takeing the gradient (Policy Gradient Theorem)

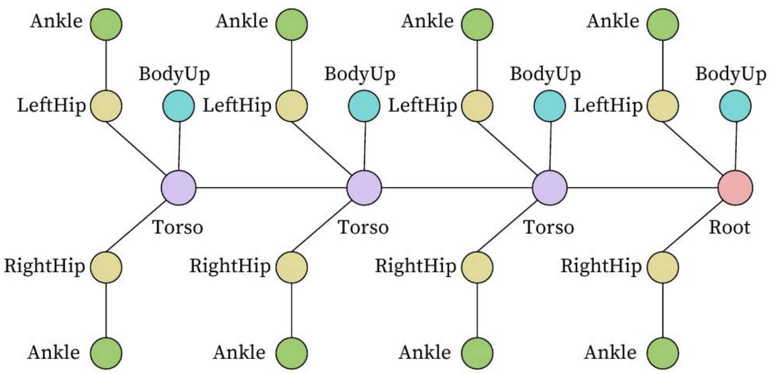
      $$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\ Q_w(s,a)]$$

      1. Variants:
         1. If Q_w is the empirical return: REINFORCE algorithm [10].
         2. If Q_w is the estimation of action-value function: Actor Critics [11].
         3. If adding KL constraints on policy updates: TRPO / PPO.
         4. If policy is deterministic: DPG [12] / DDPG [13] (Deterministic Policy Gradient).

NerveNet: Learning Stuctured Policy in RL

1. NerveNet:
   1. In traditional reinforcement learning, policies of agents are learned by MLPs which take the concatenation of all observations from the environment as input for predicting actions.
   2. We propose NerveNet to explicitly model the structure of an agent, which naturally takes the form of a graph.





Contents

1. Introduction
    1. Discrete Domain vs. Continous Domain
    2. Model Based vs. Model Free
    3. Value-based vs. Policy-based
    4. On-policy vs. Off-policy
2. Prediction vs. Control: Marching Towards Q-learning
    1. Prediction: TD-learning and Bellman Equation
    2. Control: Bellman Optimality Equation and SARSA
    3. Control: Switching to Q-learning Algorithm
3. Misc: Continous Control
    1. Policy Based Algorithm
    2. NerveNet: Learning Stuctured Policy in RL
4. Reference

Reference

[1] Watkins, Christopher JCH, and Peter Dayan. "Q-learning." Machine learning 8.3-4 (1992): 279-292.

[2] Sutton, Richard S., Doina Precup, and Satinder Singh. "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning." Artificial intelligence 112.1-2 (1999): 181-211.

[3] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

[4] Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." Nature 529.7587 (2016): 484-489.

[5] Schulman, John, et al. "Trust region policy optimization." Proceedings of the 32nd International Conference on Machine Learning (ICML-15). 2015.

[6] Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

[7] Todorov, Emanuel, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control." Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on. IEEE, 2012.

[8] Tamar, Aviv, et al. "Value iteration networks." Advances in Neural Information Processing Systems. 2016.

[9] Silver, David, UCL Course on RL, http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

[10] WILLIANMS, RJ. "Toward a theory of reinforcement-learning connectionist systems." Technical Report (1988).

[11] Konda, Vijay R., and John N. Tsitsiklis. "Actor-critic algorithms." Advances in neural information processing systems. 2000.

[12] Silver, David, et al. "Deterministic policy gradient algorithms." Proceedings of the 31st International Conference on Machine Learning (ICML-14). 2014.

[13] Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2015).