Review in Reinforcement Learning

Yuhan Zhao

Group Meeting June

June 10, 2022

Outline

1. Review in RL
   - Model Based Methods
   - Model Free Methods
   - Continuous Settings

Concepts and Settings

Reinforcement learning:

- Learn to make decisions, a new learning pattern (Sutton & Barto, 2018).
- More than MDP (multi-armed bandits, POMDP).

Reason to use MDP:

- Elegant framework and math descriptions for decision making.
- Able to quantify things and obtain analytical results.

Concepts and Settings

In general, RL uses discounted infinite horizon MDP:

- Described by the tuple $\langle \mathcal{S}, \mathcal{A}, P, u, \gamma \rangle$.
  - $\mathcal{S}, \mathcal{A}$ are finite state and action sets.
  - $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition kernel, $p(s'|s, a)$.
  - $u : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the state-action reward, $u(s, a)$.
  - $\gamma \in (0, 1)$ is the discounted factor.
- Policy $\pi : \Delta(\mathcal{A}) \times \mathcal{S} \to [0, 1]$ is a conditional probability, $\pi(a|s)$.
- Discounted reward $\sum_{t=0}^{\infty} \gamma^t \mathbb{E}_\pi[U_t|s_0]$.
- Sometimes we have initial state distribution $\rho(s)$.

Notations:

- Random variables $S_t, A_t, U_t := u_t(S_t, A_t)$ at time t.
- Feasible policy set $\Pi := \{\pi \in \Delta(\mathcal{A}) \times \mathcal{S} : \sum_a \pi(a|s) = 1, \forall s \in \mathcal{S}\}$.

Goal of RL

Find the policy π^* by solving a discounted MDP:

$$\max_{\pi \in \Pi} \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_\pi [u_t(S_t, A_t) | s_0]. \tag{1}$$

- The optimal policy π^* is stationary and deterministic.
- Averaged MDPs are also studied in some literature. The objective is $\lim_{T \to \infty} \sum_{t=0}^{T} \frac{1}{T} \mathbb{E}_\pi [u_t(S_t, A_t) | s_0]$ (Filar & Vrieze, 1997).

MDP Example

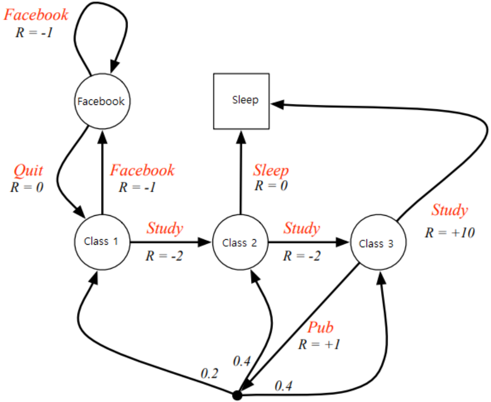

Figure: Student MDP Example from David Silver lecture slide.

Value Functions

Define state value function $v^\pi(s)$ and action value function $q^\pi(s,a)$ given a policy $\pi \in \Pi$:

$$v^\pi(s) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_\pi[u_t(S_t, A_t)|s], \tag{2}$$

$$q^\pi(s,a) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_\pi[u_t(S_t, A_t)|s, a]. \tag{3}$$

Relationship:

$$v^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q^\pi(s,a).$$

Outline

1. Review in RL
   - Model Based Methods
   - Model Free Methods
   - Continuous Settings

Models Based Methods

What does a model refer to in RL?

- Reward u and transition kernel P (Sutton & Barto, 2018).
- Discrete case: a table. Continuous case: a function.

Model based methods are also called planning.

Two categories, both uses the Bellman equation.

- Iterative methods.
- Linear programming (LP).

Bellman Equation

For any policy $\pi \in \Pi$,

$$v^\pi(s) = \sum_a u(s,a)\pi(a|s) + \gamma \sum_{s',a} p(s'|s,a)\pi(a|s)v^\pi(s'), \quad \forall s \in \mathcal{S}. \tag{4}$$

$$q^\pi(s,a) = u(s,a) + \gamma \sum_{s',a'} p(s'|s,a)\pi(a'|s')q^\pi(s',a'), \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \tag{5}$$

For optimal policy π^*,

$$v^*(s) = \max_a \left\{ u(s,a) + \gamma \sum_{s',a} p(s'|s,a)v^\pi(s') \right\}, \quad \forall s \in \mathcal{S}. \tag{6}$$

$$q^*(s,a) = u(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} q^\pi(s',a'), \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \tag{7}$$

Models Based Methods — LP

LP treats values v as decision variables.

$$\begin{aligned} \min_{v} \quad & \frac{1}{|\mathcal{S}|} \sum_{s} v(s) \\ \text{s.t.} \quad & v(s) \geq u(s,a) + \gamma \sum_{s'} p(s'|s,a) v(s'), \quad \forall a \in \mathcal{A}, \forall s \in \mathcal{S}. \end{aligned}$$

- v represents the upper bound of discounted value. So we minimize v.
- Constraints of LP assume that optimal policy is deterministic.
- Dual problem leads to occupancy measure.

Model Based Methods — Iterative Methods

Based on generalized policy iteration (GPI). Two processes:

- Policy evaluation (or prediction).
- Policy improvement (or update).

Almost all RL methods are well described as GPI.

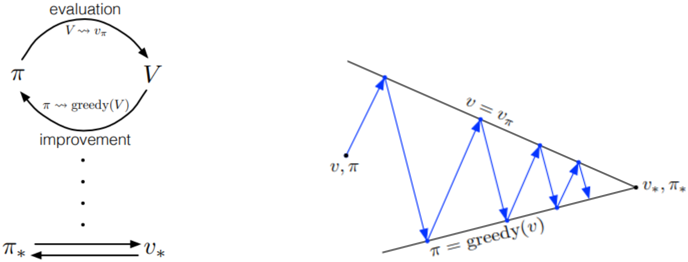

Figure: Illustration of generalized policy iteration.

Model Based Methods — Iterative Methods

Policy evaluation:

- Compute value function $v^\pi(s)$ or $q^\pi(s, a)$ given a policy π.
- Equivalent to solving a linear system $v = Tv$. Matrix inversion or iteration. Guaranteed to converge.

Policy improvement:

- Use previous value function v^π or q^π to generate a new policy π'.
- Greedy maximization (most common).

Based on how to perform policy evaluation, we have different variants. For example, policy iteration and value iteration.

Model Based Methods — Iterative Methods

Policy iteration: compute exact v^π or q^π.

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\left[r + \gamma V(s')\right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma V(s')\right]$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Figure: Policy iteration algorithm

Model Based Methods — Iterative Methods

Value iteration: update v^π or q^π only once in each iteration.

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$

Figure: Value iteration algorithm.

λ-policy iteration unifies two methods (Tsitsiklis & Bertsekas, 1996).

Outline

Model Free Settings

We can *only* observe state, action, and reward samples.

- $s_0, a_0, u_0, s_1, a_1, u_1, \ldots, s_T, a_T, u_T$.
- Do not know the transition kernel P.
- Do not know reward function:
  - Discrete case: do not know exact $u(s, a)$ table.
  - Continuous case: do not know structure of $u(s, a)$ function.
- Data trajectory terminology:
  - Episode: a sequence of trajectory with horizon T.
  - Stage: the t-th step in an episode.

Three approaches for solving MDP:

- Estimate model.
- Estimate value (Value-based methods).
- Estimate policy (Policy-based methods).

Model Free Methods — Estimate Model

Estimate the transition matrix P and reward table u.

- More important to estimate P, use empirical frequency.
- Related to system identification but not the same.
- Not common in many RL research.

Model Free Methods — Estimate Value

Key idea of value-based methods:

- Estimate q-function using data and perform GPI.
- Use greedy maximization to generate a new policy.

Why not v-function?

- v-function requires model to generate the new policy.

$$\pi^*(\cdot|s) = \arg\max_a \left\{ u(s,a) + \gamma \sum_{s',a} p(s'|s,a) v(s') \right\}, \quad \forall s \in \mathcal{S}.$$

- q-function only require greedy maximization.

$$\pi^*(\cdot|s) = \arg\max_a q(s,a), \quad \forall s \in \mathcal{S}.$$

Model Free Methods — Estimate Value

How to estimate q-function?

- Monte Carlo (MC) sampling (offline).
- Temporal Difference (TD) learning (online).
- Function approximation such as Deep Q-network (DQN).

Online vs Offline methods:

- Online: observes and processes the sample at each stage.
- Offline: operates on batches of samples.

Estimate Value — MC Sampling

Use episodic sample trajectory $\{s_0, a_0, u_0, \ldots, s_T, a_T, u_T\}$ generated by the policy π to approximate $q^\pi(s, a)$:

$$\sum_{t=0}^{T} \gamma^t u_t \approx \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_\pi[u_t(S_t, A_t)|s_0, a_0] = q^\pi(s_0, a_0).$$

- Each trajectory can estimate q-function for multiple (s, a) pairs.
- First-time visit MC and Every-time visit MC.

Estimate Value — MC Sampling

Algorithm 1: First-visit MC prediction.

$q^\pi(s,a) \leftarrow 0$, $\mathrm{Return}(s,a) \leftarrow 0 \ \forall (s,a)$;
for *each episode* **do**
  Generate $\{s_0, a_0, u_0, \ldots, s_T, a_T, u_T\}$ using π ;
  $G \leftarrow 0$;
  for $t = T, \ldots, 0$ **do**
    $G \leftarrow G + \gamma u_t$;
    if $(s_t, a_t) \notin \{(s_0, a_0), \ldots, (s_{t-1}, a_{t-1})\}$ **then**
      Append G to $\mathrm{Return}(s_t, a_t)$;
      $q^\pi(s,a) \leftarrow \mathrm{ave}(\mathrm{Return}(s_t, a_t))$;

Estimate Value — MC Sampling

Every (s, a) pair should be visited infinitely often to estimate q-function.

- Where to start?
- How to generate a "useful" trajectory?
- How to do policy improvement?

Core: use stochastic policy to encourage exploration.

- Exploring start.
- Use stochastic policy to ensure all (s, a) pair can be visited.
- On-policy vs Off-policy methods.

Estimate Value — MC Sampling

Use GPI to update the policy:

- A policy is updated at each iteration.
- There should be a policy to generate sample trajectories at each iteration.

On-policy methods:

- Use the updated policy for data simulation.
- ϵ-greedy to ensure the policy is stochastic.

Off-policy methods:

- Use stochastic behavior policy for data simulation.
- Use the target policy for policy update.
- Use important sampling to estimate the q-value.

Estimate Value — TD Learning

We can write:

$$\sum_{t=0}^{\infty} \gamma^t \mathbb{E}_\pi[u_t(S_t, A_t)|s_0, a_0] = u_0(s_0, a_0) + \gamma \mathbb{E}_\pi q^\pi(S_1, A_1)$$

Temporal Difference (TD):

$$\delta_t = u_t(S_t, A_t) + \gamma q(S_{t+1}, A_{t+1}) - q(S_t, A_t)$$

- δ_t is the error in $q(S_t, A_t)$, available at time $t + 1$.

Estimate Value — TD Learning

TD(0) or one-step TD:

- Update q-function at every stage by boot strapping.

Algorithm 2: TD(0) for prediction.

Initialize $q^\pi(s, a)\ \forall(s, a)$, $\alpha \in (0, 1]$;
for *each episode* **do**
    Initialize S, A ;
    for *each stage* $t = 0, \ldots, T-1$ **do**
        Observe U_t and S' ;
        Take action A' from π ;
        $q(S, A) \leftarrow q(S, A) + \alpha[U_t + \gamma q(S', A') - q(S, A)]$;
        $S \leftarrow S'$, $A \leftarrow A'$;

Estimate Value — TD Learning

GPI can be applied to each stage. Some RL methods with TD(0):

- SARSA: on policy, use ϵ-greedy policy to generate data.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[U_t + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)].$$

- Q-learning: off policy, use ϵ-greedy policy to generate data.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[U_t + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)].$$

- Expected-SARSA: depending on what policy to generate data.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[U_t + \gamma \mathbb{E}_\pi[Q(S_{t+1}, a)] - Q(S_t, At)].$$

Estimate Value — TD Learning

MC method, q-function is estimated by episodic data.

$$\sum_{t=0}^{\infty} \gamma^t \mathbb{E}_\pi [u_t(S_t, A_t) | s_0, a_0] \approx U_0 + \gamma U_1 + \cdots + \gamma^T U_T.$$

one-step TD, q-function bootstraps on 1-step reward.

$$\sum_{t=0}^{\infty} \gamma^t \mathbb{E}_\pi [u_t(S_t, A_t) | s_0, a_0] \approx U_0 + \gamma q(S_1, A_1)$$

n-step TD, q-function bootstraps on n-step reward.

$$\sum_{t=0}^{\infty} \gamma^t \mathbb{E}_\pi [u_t(S_t, A_t) | s_0, a_0] \approx U_0 + \gamma U_1 + \cdots + \gamma^{n-1} U_{n-1} + \gamma^n q(S_n, A_n).$$

Estimate Value — TD Learning

An example of n-step TD with $n = 2$:

$$q(S_t, A_t) \leftarrow q(S_t, A_t) + \alpha[U_t + \gamma U_{t+1} + \gamma^2 q(S_{t+2}, A_{t+2}) - q(S_t, A_t)].$$

- The learning happens after the first n samples are observed.
- The learning processes data stage-by-stage after the first n stages.
- When $t + n > T$, we set $U_{t+n} = 0$.
- The policy improvement is greedy argmax.

Estimate Value — Function Approximation

Use parameterized functions to approximate q-function.

- Suitable when state space $\mathcal{S}$ is large.
- Feature extraction is generally required to process $\mathcal{S}$.
- Action space is still discrete and small.

Approximation methods:

- Linear function approximation: $q(s, a) = \theta^{\mathsf{T}} x$.
- Nonliner function approximation $q_\theta(s, a)$:
  - Neural network (Mnih, 2015).
  - Polynomials.
  - Kernel based functions.

Estimate Value — Function Approximation

Minimize least square error

$$\min_{\theta} \mathbb{E}_{\pi,\mu}[q^{\pi}(s,a) - q_{\theta}(s,a)]^2,$$

μ is the state distribution of interest. Stochastic Gradient Descent (SGD).

$$\theta_{t+1} \leftarrow \theta_t - \alpha[q^{\pi}(S_t, A_t) - q_{\theta}(S_t, A_t)]\nabla_{\theta} q_{\theta}(S_t, A_t).$$

The problem becomes estimating $q^{\pi}(S_t, A_t)$:

- Using sample trajectory, $q^{\pi} \leftarrow G_t$.
- Using TD(0), $q^{\pi} \leftarrow U_t + \gamma q_{\theta}(S_{t+1}, A_{t+1})$.
- Other ...

Estimate Value — Function Approximation

Least square TD (Bradtke & Barto, 1996):

- Use linear parameterization to approximate v-function.
- Minimize Bellman residual $\|V_\theta - U - \gamma P \Pi V_\theta\|$.

Least square policy iteration (Lagoudakis, 2003):

- Use linear parameterization to approximate q-function.
- Minimize Bellman residual $\|Q_\theta - U - \gamma P \Pi Q_\theta\|$.
- Use greedy argmax to update policy.

Model Free Methods — Estimate Policy

Comparison:

- Value-based methods first find value functions and then update policy.
- Policy-based methods search policy directly.

Key idea:

- Parameterize policy π_θ so that the value becomes $v^{\pi_\theta}(s)$.
- Maximize the value $v^{\pi_\theta}(s)$ because $v^*(s) = \max_\pi v^\pi(s)$.
- $\theta_{t+1} \leftarrow \theta_t + \alpha \frac{\partial v^{\pi_\theta}(s_0)}{\partial \theta}$.

Suitable for continuous action spaces or large discrete action spaces.

Model Free Methods — Estimate Policy

Policy gradient:

- $$\frac{\partial v^{\pi_\theta}(s)}{\partial \theta} = \sum_s d^{\pi_\theta}(s) \sum_a \frac{\partial \pi_\theta(a|s)}{\partial \theta} q^{\pi_\theta}(s,a)$$

  with $d^\pi(s) = \sum_{t=0}^{\infty} \gamma^t \Pr(s_0 \to s_t, t, \pi)$.
- Use $\nabla \log(f(x)) = \frac{\nabla f(x)}{f(x)}$, we have

  $$\frac{\partial v^{\pi_\theta}(s)}{\partial \theta} = \mathbb{E}_\pi[q^{\pi_\theta}(s,a) \nabla_\theta \log(\pi_\theta(a|s))]$$

  with $\mathbb{E}_\pi$ refers to $\mathbb{E}_{s \sim d^\pi, a \sim \pi}$.
- Need to estimate q-function to compute the policy gradient.

Model Free Methods — Estimate Policy

Many ways to parameterize the policy π:

- Linear approximation: $\pi_\theta = \theta^\mathsf{T} x$.
- Exponential soft-max: $\pi_\theta(a|s) = \frac{e^{h_\theta(s,a)}}{\sum_b e^{h_\theta(s,b)}}$.
- Neural network.

Based on how to estimate q-function:

- REINFORCE (Williams, 1992).
  - Estimate q^π via MC sampling: $q^\pi(s,a) \approx \sum_{t=0}^{T} \gamma^t U_t$.
  - Use baseline to reduce learning variance: $q^\pi(s,a) \to q^\pi(s,a) - b(s)$.
- Actor-critic (Sutton, 1984).
  - Parameterize q-function with another w: $q_w(s,a)$.
  - Use one-step TD to update q-function stage-by-stage. Online method.
  - $w_{t+1} \leftarrow w_t + \alpha \delta_t \nabla_w q_w(S_t, A_t)$ with one-step TD error $\delta_t = U_t + \gamma q_w(S_{t+1}, A_{t+1}) - q_w(S_t, A_t)$.

Model Free Methods — Estimate Policy

Some well known policy gradient methods (Lil'Log).

- Deterministic Policy Gradient (DPG) (Silver, 2014).
- Deep Deterministic Policy Gradient (DDPG) (Lillicrap, 2016).
- Trust Region Policy Optimization (TRPO) (Schulman, 2015).
- Proximal Policy Optimization (PPO) (Schulman, 2017).
- Phasic Policy Gradient (PPG) (Cobbe, 2020).

Outline

Basic Settings and Approaches

The tuple $\langle \mathcal{S}, \mathcal{A}, P, u, \gamma \rangle$.

- Continuous state discrete action.
- Continuous state continuous action.

Do methods for discrete MDPs apply to continuous counterparts?

- Yes, but some of them have additional challenges.
- Infinite dimensional problem. Only option: function approximation.
- Previous value and policy approximation methods are ready to use.

Approaches:

- Learn the model. Too complex and rarely used (Hasselt, 2012).
- Learn the value.
- Learn the policy.

Approaches for Continuous MDP

Additional challenges happens in value-based methods.

- Curse of dimensionality to discretize action space.
- Greedy argmax is hard. Require global maximizer of the q-function.

Value function estimation:

- Minimize square error $\mathbb{E}_\pi[q^\pi - q_\theta]^2$ with SGD by estimating q^π.
- Minimize one-step TD error (Bellman residual): $\|V_\theta - U - \gamma P \Pi V_\theta\|_w^2$. (weighted norm, same for q)
- Minimize projected one-step TD error because of accuracy issue: $\|V_\theta - \mathrm{proj}[U + \gamma P \Pi V_\theta]\|_w$. (weighted norm, same for q)

Not trivial to extend the online value-based methods to continuous settings except for some problems with quadratic q-function.

Approaches for Continuous MDP

Policy based methods are much better suited.

- Q: Are mixed strategies ($\pi(\cdot|s)$ is pdf) in continuous MDP equivalent to a pure strategy ($\pi(\cdot|s) = \mu(s)$ is a number)?

TD(λ) learning is missing.

Recommended Materials

- RL course notes (David Silver, UCL).
- Reinforcement Learning An Introduction (Sutton & Barto, 2018).
- Markov Decision Processes (Puterman, 1990).
- Neuro-Dynamic Programming (Bertsekas & Tsitsiklis, 1996).