Chapter 5: Temporal difference methods (TD)

Planning

- MDP model known/given
- Solve to get v_*, q_*, π_*
  - Bellman opt. eq.
  - Policy iteration
  - Value iteration

Learning

- MDP model unknown
- Estimate v_*, q_*, π_*
  - Sampling
  - Exploration
  - RL

Prediction

- Predict effect of policy
- Policy evaluation v_π

Control

- Find optimal policy
- Find v_*, q_*

needed for learning

- Idea of TD:
  - Use observed/visited states to estimate v_* iteratively
  - Sampling / exploration informs estimation
  - New states / actions depend on estimated v_*

feedback

$$v_0 \to v_1 \to v_2 \to \cdots \to v_*$$

δR, δS

$$S_0 \longrightarrow S_1 \xrightarrow{R_1} S_2 \xrightarrow{R_2} \cdots$$

$$v_0 \quad \searrow v_1 \nearrow \quad \searrow v_2 \nearrow$$

δv

See MC course

- Stochastic approximation:
  - Iteration, map: $x_{n+1} = T(x_n)$
    $x_n \to x^*$ $\qquad x^* = T(x^*)$ attractive fixed point
  - Stochastic iteration: $X_{n+1} = T(X_n)$

$$X_{n+1} = (1-a_n) X_n + a_n T(X_n)$$
$$= X_n + a_n \underbrace{\left(T(X_n) - X_n \right)}_{\text{difference}}$$

TD error $\to 0$

$a_n \downarrow 0$

$\sum a_n = \infty$

$\sum a_n^2 < \infty$

5.1 TD(0)

- Policy iteration (prediction, not control)

- Bellman equation:

$$v_\pi(s) = E_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \,\middle|\, S_t = s\right]$$

$$v_\pi = \rho_\pi + \gamma P_\pi v_\pi \qquad = \underbrace{T}_{\text{Bellman operator}}(v_\pi)$$

- TD(0) iteration:

$$\underset{s}{V_{k+1}(S_t)} = V_k(S_t) + \alpha_k \big[\overbrace{R_{t+1} + \gamma V_k(S_{t+1}) - V_k(S_t)}^{\text{TD error}}\big]$$

$$= (1-\alpha_k)\, V_k(\underset{s}{S_t}) + \alpha_k \big[\underbrace{R_{t+1} + \gamma V_k(\overset{s'}{S_{t+1}})}_{\text{Bellman iteration update}}\big]$$

do nothing

policy evaluation

- SA version of value iteration
- SA:

expectation $\quad E_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \,|\, S_t = s\right]$

$$\Big\Downarrow$$

RV $\quad R_{t+1} + \gamma v_\pi(S_{t+1})$ for transition $\underset{s}{S_t} \to \underset{s'}{S_{t+1}}$

- α_k : annealing sequence

$\alpha_k = \alpha$ constant : Convergence in mean with episode loop

$\alpha_k \sim 1/k$: Convergence in probability

- TD(0) algorithm SB p. 120

Input: policy π
Parameters: $\alpha \in (0,1]$, # episodes, # time steps
Initialize $V(s)$

Loop for each episode:
    Initialize S
    Loop for each time steps:
        $A \leftarrow$ action from π for S
        Get S', R from action A
        $V(S) \leftarrow V(S) + \alpha\left(R + \gamma V(S') - V(S)\right)$
        $S \leftarrow S'$

5.2 Sarsa

- On-policy control iteration for estimating q_*

- Bellman optimality equation:

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

- Sarsa iteration:

$$Q_{k+1}(S_t, A_t) = Q_k(S_t, A_t) + \alpha_k\left[R_{t+1} + \gamma Q_k(S_{t+1}, A_{t+1}) - Q_k(S_t, A_t)\right]$$

$$= (1-\alpha_k) Q_k(S_t, A_t) + \alpha_k\left[R_{t+1} + \gamma Q_k(S_{t+1}, A_{t+1})\right]$$

A_t chosen ε-greedy from $Q_k(S_t, \cdot)$ before update
A_{t+1} " " " $Q_k(S_{t+1}, \cdot)$

↳ define policy π at each stage

- SA of Bellman opt. eq.
- Optimal actions A_t, A_{t+1} from current Q_k
- Use all states : s, a, r, s', a'
- On-policy : policy used to update Q

  $\quad\quad =$

  policy used to generate new actions

- Convergence: Same as TD(0)

- Sarsa algorithm SB p.130

Parameters : α, # episodes, # time steps, ε

Initialize $Q(s,a)$

Loop for each episode :

    Initialize S

    Choose A from S using ε-greedy with Q

    Loop for each time step: sarsa

        Get R, S' from action A

        Choose A' from S' using ε-greedy from Q

        $Q(S,A) \leftarrow Q(S,A) + \alpha\left(R + \gamma Q(S',A') - Q(S,A)\right)$

        $S \leftarrow S'$

        $A \leftarrow A'$

$$\begin{tikzcd}
S \arrow[rr, "R"] \arrow[dr, "\varepsilon\text{-}Q"'] & & S' \arrow[dr, "\varepsilon\text{-}Q"] & \\
& A \arrow[ur] & & A'
\end{tikzcd}$$

5.3. Q-learning

· Off-policy control iteration for estimating q_*

· Q-learning iteration:

$$Q_{k+1}(S_t, A_t) = Q_k(S_t, A_t) + \alpha_k\left[R_{t+1} + \gamma \max_a Q_k(S_{t+1}, a) - Q_k(S_t, A_t)\right]$$

$$= (1-\alpha_k)\, Q_k(S_t, A_t) + \alpha_k\left[R_{t+1} + \gamma \max_a Q_k(S_{t+1}, a)\right]$$

Find max over action

A_t chosen in ε-greedy way from $Q_k(S_t, \cdot)$

· Variation of Sarsa
 · A_t chosen ε-greedy from Q_k
 · A_{t+1} " greedy " Q_k $\varepsilon = 0$
 · A_{t+1} not used after

· Off-policy : Action selected $\neq$ action used to update Q

· Convergence : Same as TD(0)

· Q-learning algorithm SB. p.131

Parameters : α, # episodes, # time steps, ε
Initialize $Q(s,a)$

Loop for each episode:
  Initialize S
  Loop for each time steps:
    Choose A from S using ε-greedy from Q — Define policy
    Get S', R from A
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left(R + \gamma \max_a Q(S',a) - Q(S,A)\right)$
    $S \leftarrow S'$

$$\begin{tikzcd} & & R \\ S \arrow[rr] \arrow[dr, "\varepsilon\text{-}g"'] & & S' \\ & A \arrow[ur] & \end{tikzcd}$$

$$\max_a Q(S', a)$$

for update

- Maximization bias:

  max estimated q $\neq$ max in expectation

  $$\max \hat{f}(\cdot) \neq \max E[f(\cdot)]$$

  - Choosing largest value necessarily above expectation

$\Rightarrow$ Q i.e. $\hat{q}$ generally $>$ true q_*

V i.e. $\hat{v}$ " " " v_*

Conclusions – to go further

- Represent $v(s)$ $q(s,a)$ — SB Chaps 9–12
  $|\mathcal{S}|$ $|\mathcal{S}| \times |\mathcal{A}|$
  - Parameterizations / Projections : $v(s;\theta)$
    $q(s,a;\theta)$
  - Neural networks : $s \longrightarrow \overset{\theta}{\boxed{\;\;}} \longrightarrow v(s;\theta)$
    deep RL
    $s, a \rightrightarrows \boxed{\;\;} \longrightarrow q(s,a;\theta)$
  - Adapt TD(0), Sarsa, Q-learning to update parameters θ

- Estimation, sampling : $Q_n \to Q_{n+1}$ — SB Chaps 5–7
  $V_n \to V_{n+1}$
  - Use longer history
  - TD(λ)
  - Eligibility traces

- Policy approximation : $\pi(s) \to \pi(s;\theta)$ — SB Chap 13
  - Gradient ascent : $\theta_{n+1} = \theta_n + \beta \underbrace{\nabla J(\theta)}_{?}$
  - Policy gradient
  - Actor-critic methods
    $\downarrow$ $\downarrow$
    $\pi \rightleftarrows q(s,a)$

- Other topics
  - Multi-agent
  - Partially observed MDPs : POMDPs
  - etc.