ML818 Reinforcement Learning and Planning
Lecture Notes

Hugo Touchette
Department of Mathematical Sciences
Stellenbosch University, South Africa

Started: May 3, 2021; last compiled: June 27, 2022

References

- **[SB]** Sutton, Barto, Reinforcement Learning: An Introduction, 2nd Edition, MIT Press, 2018. Available online.
- **[Silver]** D. Silver, Introduction to Reinforcement Learning, 2015. See website

1. Introduction

- Basic diagram:
- Basic concepts:
  - Learning from interactions (exploration, trial and error)
  - Use experience to improve performance
  - Goal-directed learning (reward)
  - Actions can affect future (delayed reward)
  - Actions contingent on situations (associativity)
  - Exploration vs exploitation
  - Clear separation of action (agent), state (environment), reward
  - Output/goal: Find optimal policy mapping state to action
  - Uncertainty in environment and action effects (need probabilistic framework)
  - 3rd paradigm of ML
- Reward hypothesis (Silver:L1): All goals can be described by the maximisation of expected cumulative reward.
  - Reward communicates what we want to achieve, not how we want to achieve it (no instructions).
- Comparison with supervised and unsupervised learning:
  - Supervised learning:

    - Training set of examples
    - Instructive feedback: indicate correct action to take (input, label)
    - External supervisor
    - Extrapolate, generalize
    - Not interactive
  - Reinforcement learning:
    - No examples of desired behaviour
    - No representative set of examples
    - Learn from experience, not training set
    - Evaluate actions to be taken (need for exploration)
    - Online, on the fly learning
  - Unsupervised learning:
    - Goal is to find hidden structure (eg classification boundary)
    - Not necessarily based on reward

2. Multi-armed bandit problem

SB:C2. Section to read for the coursework.

2.1. Model

- Action: $A \in \{1, \dots, k\}$
- Reward: R
  - Set of possible values depend on problem considered.
  - $P(r|a)$ (stationary)
  - Reward process:

$$\begin{array}{cccc} A_1 & A_2 & \cdots & A_n \\ \downarrow & \downarrow & & \downarrow \\ R_1 & R_2 & \cdots & R_n \end{array} \tag{1}$$

- Action value:

$$q(a) = E[R|A=a] = \sum_r r P(r|a) \tag{2}$$

- Goal: Learn set of actions that maximize expected reward. (Maximize reward over time if played repeatedly.)
- Optimal strategy: Choose a^* with highest expected reward:

$$a^* = \arg\max_a q(a) \tag{3}$$

  - Deterministic strategy.
  - Any mixed strategy has necessarily an expected reward smaller than or equal to $q(a^*)$.
- Non-associative:

- No situation (environment) affecting action to be taken.
- Just act to maximize reward.
- In RL: learn action in a given situation to max reward (associative).
- Bandit problems can be contextual by adding a situation variable:

$$(\text{situation}, \text{action}) \to \text{reward}$$

- Full RL problem:

$$(\text{situation}, \text{action}) \to (\text{reward}, \text{new situation})$$

2.2. Off-line approximation

Note: Slightly different notation from SB.

- Idea: Action value averaged over time is an estimator of $q(a)$ since reward is stationary.
- Action sequence (trajectory): $(A_i)_{i=1}^n$
- Reward sequence (trajectory): $(R_i)_{i=1}^n$
- Action value estimator:

$$Q_n(a) = \frac{\sum_{i=1}^n R_i \mathbb{1}_{A_i=a}}{\sum_{i=1}^n \mathbb{1}_{A_i=a}} \tag{4}$$

  - Rationale:
$$Q_n(a) = \sum_r r\,\hat{P}_n(r|a)$$
    and
$$\hat{P}_n(r|a) = \frac{\hat{P}_n(r,a)}{\hat{P}_n(a)} = \frac{\frac{1}{n}\sum_{i=1}^n \mathbb{1}_{R_i=r}\mathbb{1}_{A_i=a}}{\frac{1}{n}\sum_{i=1}^n \mathbb{1}_{A_i=a}}.$$
  - Note: No hats in SB for estimators.
- Greedy action selection (policy): At time $n+1$

$$A_{n+1} = \arg\max_a Q_n(a)$$

- ε-greedy:
  - Probability $1-\varepsilon$: Greedy
  - Probability ε: $A_{n+1} \sim \mathcal{U}\{1,\dots,k\}$ (uniform action)

2.3. On-line approximation

Note: Slightly different notation from SB.

- Fix action at a
- Look at all reward received for that action: $R_1, \dots, R_n$
- Iteration:
$$Q_n(a) = Q_{n-1}(a) + \frac{1}{n}[R_n - Q_{n-1}(a)]$$
- Asynchronous update:
$$\begin{aligned} Q_n(a) &= Q_{n-1}(a) + \alpha_n[R_n - Q_{n-1}(a)] \\ &= (1-\alpha_n)Q_{n-1}(a) + \alpha_n R_n \end{aligned} \tag{5}$$
for the realised action $A_n = a$ at time n and
$$Q_n(a') = Q_{n-1}(a')$$
for all other actions $a' \neq a$. In compact form:
$$Q_n(a) = Q_{n-1}(a) + \alpha_n[R_n - Q_{n-1}(a)]\mathbb{1}_{A_n=a}.$$
    - Example of stochastic approximation or iteration (see Monte Carlo course).
    - Initial value: $Q_1(a) = 0$ for all a
    - Learning rate: $\alpha_n = 1/n$
    - Optimism: Put larger than average initial value to encourage exploration.

See Q1 of the CW.

3. Markov process (MP)

- Discrete time: $t = 0, 1, \dots$
- State: S_t
- State space: $\mathcal{S}$ (finite)
- Fundamental transition probability:
$$p(s'|s) = P(S_{t+1} = s'|S_t = s) \tag{6}$$
    - Basic transition: $S_t \to S_{t+1}$
    - Assumed stationary (homogeneous, time-independent).
    - Markov chain
$$S_0 \to S_1 \to \cdots S_t \tag{7}$$
    - Normalisation:
$$\sum_{s'} p(s'|s) = 1 \tag{8}$$
for all s.

    - Matrix form: $P_{ij} = P(i \to j) = p(j|i)$

- Evolution equation:
$$p(s') = \sum_s p(s'|s)p(s) \tag{9}$$
In matrix form: $p' = pP$.

- Expectation:
$$\mathbb{E}[f(S_{t+1})|S_t = s] = \sum_{s'} f(s')\, p(s'|s) \tag{10}$$
In matrix form: Pf.

- Marginalisation:
$$\sum_y p(x,y) = p(x) \tag{11}$$
and
$$\sum_y p(x,y|z) = p(x|z) \tag{12}$$

- Chain rule:
$$p(x|y)p(y) = p(x,y) \tag{13}$$
and
$$p(x|y,z)p(y|z) = p(x,y|z) \tag{14}$$

4. Markov reward process (MRP)

- Reward: $R_t \in \mathcal{R} \subset \mathbb{R}$
    - Cost or utility in control theory.
    - Random variable correlated with transitions in MP.
    - Sometimes given as function $r(s,s')$ of $S_t = s$ and $S_{t+1} = s'$.

- Fundamental conditional probability:
$$p(s',r|s) = P(S_{t+1} = s, R_{t+1} = r|S_t = s) \tag{15}$$
    - Basic transition:
$$\begin{array}{ccc} & & R_{t+1} \\ S_t & \to & S_{t+1} \end{array} \tag{16}$$
    - Evolution with reward for each transition (state reached).
    - Assumed stationary.
    - No actions involved yet: just evolution of state with reward/cost.
    - Reward probability:
$$p(r|s) = \sum_{s'} p(s',r|s) \tag{17}$$
Similar to bandit problem (reward process).

    - Transition probability:

$$p(s'|s) = \sum_r p(s', r|s) \tag{18}$$

- Expected state reward:

$$\rho(s) = \mathbb{E}[R_{t+1}|S_t = s] = \sum_r r\, p(r|s) \tag{19}$$

    - Expected one-time reward from state s (when leaving s).
    - Sometimes used with $p(s'|s)$ to define MRPs instead of $p(s', r|s)$.
    - Transition from state s, then get expected reward $\rho(s)$.

- Return:

$$G_t = R_{t+1} + R_{t+2} + \cdots + R_T \tag{20}$$

    - Total reward/cost to go from time t.
    - Also called cost or utility.
    - Time additive, best calculated/estimated backwards in time:

$$G_T = 0, \quad G_{T-1} = R_T, \quad G_{T-2} = R_{T-1} + \gamma R_T, \quad \ldots \tag{21}$$

    - T: final/termination time. In many cases, random time. Defines episodes.
    - Episodic process/task (finite-horizon): $T < \infty$
    - Continuing process/task (indefinite or infinite horizon): $T = \infty$
    - Can express episodic processes as continuing processes by defining termination/absorbing state with 0 reward.

- Discounted return:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 G_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{22}$$

    - Discount rate: $\gamma \in [0, 1]$
    - Balances present and future rewards.
    - Discount makes sure G_t is finite even for infinite horizon. (Upper bound: stay in the state with largest reward forever.)
    - Myopic: $\gamma \approx 0$, immediate rewards count
    - Far-sighted: $\gamma \approx 1$
    - Iteration:

$$G_t = R_{t+1} + \gamma G_{t+1} \tag{23}$$

- Value function:

$$v(s) = \mathbb{E}[G_t|S_t = s] \tag{24}$$

    - Expected return (cost to go) from state s.
    - Characterises (starting) states leading to more return in the future.

    - Does not depend on t, since transition probabilities assumed stationary and infinite return considered.
    - For $\gamma = 0$: $v(s) = \rho(s)$ immediate reward.s

- Bellman equation:

$$\begin{aligned} v(s) &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s] \end{aligned} \tag{25}$$

  Explicitly:

$$\begin{aligned} v(s) &= \sum_{s',r} r\, p(s', r|s) + \gamma \sum_{s',r} v(s')\, p(s', r|s) \\ &= \sum_{r} r\, p(r|s) + \gamma \sum_{s'} v(s')\, p(s'|s) \end{aligned} \tag{26}$$

    - Immediate reward + discounted value of successor state.
    - Only need $\rho(s)$ and $p(s'|s)$ to calculate the value function of MRPs.
    - Linear equation:
$$v = \rho + \gamma P v. \tag{27}$$
    - Unique solution:
$$v = (I - \gamma P)^{-1} \rho. \tag{28}$$
    - Can be solved as (backup) iteration since a contraction.

5. Markov decision process (MDP)

SB:C3; Silver:L2.

5.1. Model

- Environment state: $S_t \in \mathcal{S}$
    - Information state for making decision.
    - External to agent.
    - Complete state to make decision.
- Action: $A_t \in \mathcal{A}(s)$
    - Agent's state.
    - State space (available actions) usually depends on current state.
    - Can take $\mathcal{A}(s) = \mathcal{A}$.
- Reward: $R_t \in \mathcal{R}$
    - Signal from environment.
    - Defines good vs bad actions and thus goal of RL problem.

    - Convention: low or negative reward $\rightarrow$ bad action; high or positive reward $\rightarrow$ good action.
    - Zero/baseline reward not important.
    - Function of current state and action.
    - In control problems, given as deterministic function $r(s, a, s')$.
    - More generally: random variable correlated with state and action.

- History/trajectory:
$$S_0, A_0; R_1, S_1, A_1; R_2, S_2, A_2; \dots \tag{29}$$

- Fundamental transition probability:
$$p(s', r|s, a) = P(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a) \tag{30}$$
    - Basic transition:
$$\begin{tikzcd} & R_{t+1} \\ S_t \arrow[r] & S_{t+1} \\ A_t \arrow[ur] & \end{tikzcd} \tag{31}$$
    - Assumed stationary.
    - Simplified Markov model of decision process (coupled dynamics of environment, agent and reward).
    - Normalisation:
$$\sum_{s', r} p(s', r|s, a) = 1 \tag{32}$$
      for all s, a.
    - State transition probability:
$$p(s'|s, a) = \sum_r p(s', r|s, a) \tag{33}$$
    - Reward probability:
$$p(r|s, a) = \sum_{s'} p(s', r|s, a) \tag{34}$$

- Expected reward:
$$\rho(s, a) = \mathbb{E}[R_{t+1}|S_t = s, A_t = a] \tag{35}$$
  Explicity:
$$\begin{aligned} \rho(s, a) &= \sum_{s', r} r\, p(s', r|s, a) \\ &= \sum_r r\, p(r|s, a) \end{aligned} \tag{36}$$
    - Expected reward in one timestep if in state s and take action a.
    - Does not depend on t.
    - Sometimes used with $p(s'|s, a)$ to defined MDPs (see Silver) instead of $p(s', r|s, a)$ (see SB). SB: Considering state jointly with reward results in a clearer in presentation.

5.2. Policy

- Policy transition probability:

$$\pi(a|s) = P(A_t = a|S_t = s) \tag{37}$$

  - Mapping of states to actions/decision.
  - Probabilistic policy (random/noisy/exploratory actions).
  - Assumed stationary.
  - Deterministic policy: $\pi(s) = a$
  - Joint probability:

$$p(s', r, a|s) = p(s', r|s, a)\pi(a|s). \tag{38}$$

    Follows from chain rule.
  - Conditional expectation:

$$\mathbb{E}_\pi[\cdot|S_t = s] = \mathbb{E}_\pi \mathbb{E}[\cdot|S_t = s, A_t] \tag{39}$$

  - Transition probability:

$$p_\pi(s'|s) = \sum_a p(s'|s, a)\pi(a|s) = \sum_{a,r} p(s', r|s, a)\pi(a|s). \tag{40}$$

    * Effective transition probability under π.
    * MDP with fixed policy π = MRP.

- Expected state reward under policy π:

$$\rho_\pi(s) = \mathbb{E}_\pi[R_{t+1}|S_t = s] = \mathbb{E}_\pi \mathbb{E}[R_{t+1}|S_t = s, A_t] = \mathbb{E}_\pi[\rho(s, A_t)] \tag{41}$$

  Explicitly:

$$\begin{aligned} \rho_\pi(s) &= \sum_{s',r,a} r\, p(s', r|s, a)\pi(a|s) \\ &= \sum_r r\, p(r|s) \\ &= \sum_a \rho(s, a)\pi(a|s) \end{aligned} \tag{42}$$

  - Expected one-time reward from state s.
  - $\rho(s, a)$ does not depend on policy since s and a fixed.
  - Does not depend on t, since transition probabilities assumed stationary.

5.3. Value functions

- State value function:
$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] \tag{43}$$
    - Expected future reward (cost to go) from state s with policy π.
    - Long term value of state s under policy π.
    - Function of $\pi(\cdot|s)$ since s fixed.
    - Short: Value function.
    - Does not depend on t, since we consider stationary policies and continuing tasks (infinite horizon).
- State-action value function:
$$q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \tag{44}$$
    - Expected future reward (cost to go) if action a taken from state s.
    - Value function of action a when in state s under policy π.
    - Function of number $\pi(a|s)$ since s and a fixed.
    - Short: Action value function.
    - State-action pair taken at the same time.
    - Does not depend on t, since we consider stationary policies and continuing tasks (infinite horizon).
- Connection:
$$v_\pi(s) = \mathbb{E}_\pi[q_\pi(s,A_t)|S_t = s] = \sum_a q_\pi(s,a)\pi(a|s) \tag{45}$$
    - Value function = expected action value function over policy.
    - Deterministic policy: $a = \pi(s)$. Then
$$v_\pi(s) = q_\pi(s,\pi(s)) \tag{46}$$

5.4. Bellman equations

- Bellman equation for value function:
$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s] \tag{47}$$

  Explicitly:
$$\begin{aligned} v_\pi(s) &= \sum_{s',r,a} r\, p(s',r|s,a)\pi(a|s) + \gamma \sum_{s',r,a} v_\pi(s')\, p(s',r|s,a)\pi(a|s) \\ &= \sum_r r\, p(r|s) + \gamma \sum_{s'} v_\pi(s') p_\pi(s'|s) \\ &= \sum_{r,a} \rho(s,a)\pi(a|s) + \gamma \sum_{s',a} v_\pi(s') p(s'|s,a)\pi(a|s) \end{aligned} \tag{48}$$

- ◦ Immediate reward + discounted value of successor state.
  - ◦ Only need $\rho_\pi(s)$ and $p_\pi(s'|s)$ to calculate the value function of MDPs.
  - ◦ Linear equation:
$$v_\pi = \rho_\pi + \gamma P_\pi v_\pi \tag{49}$$
  - ◦ Unique solution:
$$v_\pi = (1 - \gamma P_\pi)^{-1} \rho_\pi \tag{50}$$

- Bellman equation for action value function:

$$q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1})|S_t = a, A_t = a] \tag{51}$$

  Explicitly:

$$\begin{aligned} q_\pi(s,a) &= \sum_{s'r} r\, p(s',r|s,a) + \gamma \sum_{a',s',r} q_\pi(s',a')\pi(a'|s')p(s',r|s,a) \\ &= \sum_{r} r\, p(r|s,a) + \gamma \sum_{a',s'} q_\pi(s',a')\pi(a'|s')p(s'|s,a) \\ &= \rho(s,a) + \gamma \sum_{a',s'} q_\pi(s',a')\pi(a'|s')p(s'|s,a) \end{aligned} \tag{52}$$

  - ◦ Immediate reward + discounted reward of successor state, action.
  - ◦ Can be seen as linear equation for matrix $q(s,a)$.
  - ◦ Only need $\rho(s,a)$, $p(s'|s,a)$ and policy $\pi(a|s)$ to calculate the action value function of MDPs.

5.5. Optimal policies

SB:3.6

- Optimal value function:
$$v_*(s) = \max_\pi v_\pi(s) \tag{53}$$
  - ◦ Max on $\pi(\cdot|s)$ with s fixed - possible policies from state s.
  - ◦ Defines best possible actions/decisions from s.
  - ◦ Defines partial ordering of policies (from state s).
  - ◦ At least one solution - optimal policy.

- Optimal action value function:

$$q_*(s,a) = \max_\pi q_\pi(s,a) \tag{54}$$

  - ◦ Optimal policy used for successor states.
  - ◦ Max also on $\pi(\cdot|s)$ with s fixed - possible policies from state s.
  - ◦ Same solution as $v_*(s)$ (see below).

- Optimal policy:
$$\pi_* = \arg\max_\pi v_\pi(s) \tag{55}$$

  - Defines optimal policy $\pi(\cdot|s)$ from state s.
  - Solution depends on s.

- General results (Silver:L2): For any MDPs
  - There exists at least one optimal policy π_*
  - All optimal policies achieves the optimal v_*
  - All optimal policies achieves the optimal q_*
  - Optimal policy is deterministic (deterministic action for given state).
  - Optimal policy is stationary.

- Optimal policy:
$$\pi_*(s) = \arg\max_a q_*(s,a) \tag{56}$$

- Optimal value function:
$$v_{\pi_*}(s) = q_{\pi_*}(s, \pi_*(s)) \tag{57}$$
  so that
$$v_*(s) = \max_a q_*(s,a) \tag{58}$$
  The maximum defines the deterministic optimal policy.

- Knowing π_* $\Leftrightarrow$ knowing q_* $\Leftrightarrow$ knowing v_* (see next section).

5.6. Bellman's optimality equation

- Value function:
$$v_*(s) = \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \tag{59}$$
  - Optimal from s = take optimal action from s and then optimal from S_{t+1}.
  - For deterministic actions: policy determined by moving to larger value function.
  - Expectation independent of π_*: only an expectation on the one-step transition on reward and state reached from s and a.
  - Max is out of expectation because it influences expected reward.
  - Nonlinear equation because of max (compare with linear Bellman equation for v_π).
  - Max defines the optimal policy (optimal deterministic action from state s).

- Action value function:
$$\begin{aligned} q_*(s,a) &= \mathbb{E}[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') | S_t = s, A_t = a] \\ &= \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \end{aligned} \tag{60}$$
  Latter line, more explicitly:
$$\begin{aligned} q_*(s,a) &= \rho(s,a) + \gamma \sum_{s'} v_*(s') p(s'|s,a) \\ &= \rho(s,a) + \gamma (P_a v_*)(s) \end{aligned} \tag{61}$$

  - Optimal from (s, a) = optimal after reaching S_{t+1} by taking optimal action at next time.
  - One-step greedy if optimal policy known from S_{t+1} (not the same as saying one-step greedy is optimal).
  - Expectation independent of π_*: only an expectation on the one-step transition on reward and state reached from s and a.
  - Max is inside expectation because it does not influences expected reward.
  - Additional max on a gives v_*.
  - Nonlinear equation because of max (compare with linear Bellman equation for q_π).
  - Max defines the optimal policy (optimal deterministic action from state s).

- Bellman's optimality principle:
  - An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.
  - From state s, first optimal action, then optimal policy from successor state.
- Planning: Solve optimality equations to find π_*.
- Backup diagrams and variants: See notes.

5.7. Examples

- Student MRP and MDP: Silver:L2
- Gridworld: SB:C2 and Silver:L2

6. Dynamic programming

SB:C4; Silver:L3

- MDP model known.
- DP: Solve optimality equations iteratively to find v_*, q_* and π_* (planning).
- No learning of dynamics and policy (model).
- DP in general: Solve problems by solving subproblems.

6.1. Policy evaluation

- Goal: Compute v_π for given π.
- Direct method: Solve linear equation for v_π.

- Iterative method:

$$v_{k+1}(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s] \tag{62}$$

  Vector form:

$$v_{k+1} = \rho_\pi + \gamma P_\pi v_k \tag{63}$$

  - Initial value v_0 arbitrary except $v_0(\text{terminal state}) = 0$.
  - v_π fixed point.
  - Called Bellman backup.
  - Iteration is a contraction.
  - Convergence: $v_k \to v_\pi$ as $k \to \infty$.
  - Synchronous update.
  - Can also use in-place udpate (see Silver:L3).

6.2. Policy iteration

- Consider only deterministic policies from now on.
- v_π known for policy π. See if choosing action $a \neq \pi(s)$ (off policy) improves v_π.
  - Value function on policy: $v_\pi(s) = q_\pi(s, \pi(s))$
  - Action value function off policy: $q_\pi(s, a)$, $a \neq \pi(s)$
- Policy improvement theorem: Let π and π' be such that

$$q_\pi(s, \pi'(s)) \geq v_\pi(s) \tag{64}$$

  for all s. Then π' is as good as or better than π in the sense that

$$v_{\pi'}(s) \geq v_\pi(s). \tag{65}$$

- Greedy policy selection:

$$\pi'(s) = \arg\max_a q_\pi(s, a) \tag{66}$$

  - Equation for updating π.
  - Fixed point: π_* for q_*, yielding v_*.
  - Policy improvement gives strict improvement unless optimal.
  - Results also apply to stochastic policies (not needed).
- Iteration algorithm:

$$\pi_0 \to v_{\pi_0} \to \pi_1 \to v_{\pi_1} \to \cdots \tag{67}$$

  - Step 1 (evaluation): Evaluate v_π exactly or iteratively.
  - Step 2 (policy improvement): Greedy selection.

6.3. Value iteration

- Iteration:
$$v_{k+1}(s) = \max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s, A_t = a] \tag{68}$$
  More explicitly:
$$\begin{aligned} v_{k+1}(s) &= \max_a [\rho(s,a) + \gamma \sum_{s'} v_k(s') p(s'|s,a)] \\ &= \max_a [\rho(s,a) + \gamma (P_a v_k)(s)] \end{aligned} \tag{69}$$
  - Iterative use of Bellman's optimality equation (dynamic programming).
  - Uses only one step of iterative policy evaluation.
  - Cannot be written in vector form since the max depends on s.
  - Max defines optimal policy at stage $k+1$ (policy improvement).
  - Policy improvement is greedy
  - Can also used ε-greedy with annealing sequence to force exploration.
  - Contraction: $v_k \to v_*$ as $k \to \infty$.
  - Can be calculated in synchronous or in-place way.
- Deterministic value iteration (Silver:L3):
  - Optimal solution $v_*(s')$ known for some states, e.g., terminal state.
  - Propagate solution to other state with
$$v_*(s) = \max_a [\rho(s,a) + \gamma (P_a v_*)(s)] \tag{70}$$
  - Example: Maze (Silver:L1), Shortest path (Silver:L3).

6.4. Action value iteration

- Iteration 1:
$$q_{k+1}(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_{a'} q_k(S_{t+1}, a') | S_t = s, A_t = a] \tag{71}$$
  More explicitly:
$$q_{k+1}(s,a) = \rho(s,a) + \gamma \max_{a'} \sum_{s'} q_k(s', a') p(s'|s,a) \tag{72}$$
  - Fixed point: q_*
  - Max defines policy improvement at stage $k+1$ (see below).
  - Contraction: $q_k \to q_*$ as $k \to \infty$.
  - Can be calculated in synchronous or in place way.

- Iteration 2:
$$q_k(s,a) = \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s, A_t = a] \tag{73}$$
More explicitly:
$$\begin{aligned} q_k(s,a) &= \rho(s,a) + \sum_{s'} v_k(s') p(s'|s,a) \\ &= \rho(s,a) + \gamma (P_a v_k)(s) \end{aligned} \tag{74}$$

- Policy improvement:
$$\pi_{k+1}(s) = \arg\max_a q_k(s,a) \tag{75}$$
    - Equation strictly valid only for optimal policy.
    - Only an estimate of π_* in the iteration.

6.5. Bellman operator*

Silver:L3

- Bellman backup (iteration) operator:
$$T_\pi(v) = \rho_\pi + \gamma P_\pi v \tag{76}$$

- γ-contraction:
$$\|T_\pi(u) - T_\pi(v)\| \leq \gamma \|u - v\| \tag{77}$$

- Contraction mapping theorem: For any metric space V that is complete (i.e., closed) under an operator T that is a γ-contraction:
    - T converges to a unique fixed point.
    - Convergence at a linear rate of γ.

- Bellman optimality backup operator:
$$T_*(v) = \max_a [\rho(\cdot, a) + \gamma P_a v] \tag{78}$$
Also a γ-contraction.

7. Temporal difference algorithms

- Model free: Sample states directly (simulations or experiments).
- Exploration: Policy selection and estimation of value function by sampling.
- Iterative, on-line learning. Better for continuing tasks.
- Learning need to behave non-optimally to explore all actions and then find the optimal one.
- Prediction (policy evaluation) vs control (finding optimal policy).

- Prediction: Based on value function. Control: Based on action value function.
- On-policy: Improved policy used to generate histories (generate new states and actions).
- Off-policy: Improve a side (off-line) policy (target policy) other than one used in simulations (behavior policy).

7.1. TD(0)

- Policy iteration (prediction, not control).
- Iteration:

$$\begin{aligned} V_{k+1}(S_t) &= V_k(S_t) + \alpha_k \underbrace{[R_{t+1} + \gamma V_k(S_{t+1}) - V_k(S_t)]}_{\text{TD error}} \\ &= (1-\alpha_k)V_k(S_t) + \alpha_k \underbrace{[R_{t+1} + \gamma V_k(S_{t+1})]}_{\text{target}} \end{aligned} \tag{79}$$

  - Iterative form of the Bellman equation.
  - Bellman operator: $T(v) = \rho + \gamma P v$
  - Stochastic approximation: $R_{t+1} + \gamma V(S_{t+1})$
  - Stop at terminal state or after some time.
- Convergence:
  - $\alpha_k = \alpha$: Convergence in mean with episode loop.
  - α_k satisfying SA conditions: Convergence in probability.
- Algorithm: SB:p.98.

7.2. Sarsa

- On-policy control iteration for estimating q_*.
- Iteration:

$$\begin{aligned} Q_{k+1}(S_t, A_t) &= Q_k(S_t, A_t) + \alpha_k[R_{t+1} + \gamma Q_k(S_{t+1}, A_{t+1}) - Q_k(S_t, A_t)] \\ &= (1-\alpha_k)Q_k(S_t, A_t) + \alpha_k[R_{t+1} + \gamma Q_k(S_{t+1}, A_{t+1})] \end{aligned} \tag{80}$$

  with A_t and A_{t+1} chosen in an ε-greedy way from Q_k (i.e., before the update).

  - Iterative form of the optimality Bellman equation in which the optimal actions are estimated for the current estimate of q.
  - Use all of s, a, r, s', a' = Sarsa.
  - Bellman operator: $T(q) = \rho + \gamma P q$
  - Stochastic approximation: $R_{t+1} + \gamma q(S_{t+1}, A_{t+1})$
  - Stop at terminal state or after some time.

  - On-policy method: Policy used to update q is the policy used to generate states and actions.

- Convergence: Same as TD(0).
- Algorithm: SB:p.106.

7.3. Q-learning

- Off-policy control iteration for estimating q_*.
- Iteration:

$$\begin{aligned} Q_{k+1}(S_t, A_t) &= Q_k(S_t, A_t) + \alpha_k[R_{t+1} + \gamma \max_a Q_k(S_{t+1}, a) - Q_k(S_t, A_t)] \\ &= (1 - \alpha_k)Q_k(S_t, A_t) + \alpha_k[R_{t+1} + \gamma \max_a Q_k(S_{t+1}, a)] \end{aligned} \tag{81}$$

  with A_t chosen in an ε-greedy way from Q_k (i.e., before the update).

  - A_t estimated for the current estimate of q in ε-greedy way, but the update is done with the greedy action, which is not used after.
  - Off-policy method: Action selected at each time is not necessarily the action coming from the max in the update.
  - Sarsa = Q-learning for $\varepsilon = 0$ (greedy).

- Convergence: Same as TD(0) and Sarsa.
- Algorithm: SB:p.107.
- Maximisation bias: max over estimated $q \neq$ true maximum of expectation. Choosing largest value is necessarily above expectation.

8. Function approximations

To do.

9. Policy gradient methods

9.1. Framework

- Idea: Approximate/represent/update policy $\pi(a|s)$ rather than action value function.
- Advantages of policy approximation methods:
  - Not based a priori on action value function.
  - Can retain action randomness for exploration (action value methods lead to deterministic policies).
  - Easier to approximate/represent than action value function.
  - Good approach for including a priori information in control/actions.

- ◦ More adapted for continuous control/actions.

- Parameterised policy:

$$\pi(a|s,\theta) = P(A_t = a|S_t = s, \theta_t = \theta) \tag{82}$$

  - ◦ Parameters: θ
  - ◦ Assumed differentiable w/r θ.
  - ◦ For exploration, policy should not be deterministic.

- Gradient ascent:

$$\theta_{t+1} = \theta_t + \alpha \underbrace{\widehat{\nabla J(\theta_t)}}_{\text{gradient estimate}} \tag{83}$$

- Parameterised learned (estimated) value function: $\hat{v}(s, w)$

- Exponential softmax ansatz:

$$\pi(a|s,\theta) = \frac{e^{h(s,a,\theta)}}{\sum_b e^{h(s,b,\theta)}} \tag{84}$$

  - ◦ Exponential ensures positivity (for representing probabilities).
  - ◦ Linear features: $h(s,a,\theta) = \theta^T x(s,a)$

- Gaussian actions:

$$\pi(a|s,\theta) = \frac{1}{\sqrt{2\pi\sigma(s,\theta)^2}} e^{-(a-\mu(s,\theta))^2/2\sigma(s,\theta)^2} \tag{85}$$

9.2. Episodic tasks

- Episodic cost:

$$J(\theta) = v_{\pi_\theta}(s_0) \tag{86}$$

  for starting state s_0.

- Policy gradient theorem:

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla_\theta \pi(a|s,\theta) \tag{87}$$

  - ◦ Proof p. 269.
  - ◦ $\mu(s)$ = distribution of states under π
  - ◦ $q_\pi(s,a)$ = action value function for policy $\pi_\theta = \pi(\cdot|\cdot,\theta)$.

- REINFORCE:

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla_\theta \pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)} \tag{88}$$

  - ◦ Stochastic approximation in S_t and A_t of

$$\nabla J(\theta) = \mathbb{E}_\pi \left[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\theta) \right] \tag{89}$$

- Uses complete return from time t (not online, more in line with Monte Carlo methods).
- May have high variance and thus slow learning.
- Can only be used for episodic tasks with known return for finite hitting/termination time.
- Eligibility vector:

$$\frac{\nabla_\theta \pi(a|s,\theta)}{\pi(a|s,\theta)} = \nabla \ln \pi(A_t|S_t,\theta) \tag{90}$$

- REINFORCE with baseline:

$$\theta_{t+1} = \theta_t + \alpha(G_t - b(S_t))\frac{\nabla_\theta \pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)} \tag{91}$$

  - Does not change expected value, but changes variance properties.
  - Value function baseline: $b(S_t) = \hat{v}(S_t, w_t)$
  - Can include learning for parameters w of value function and for parameters θ for policy.

- One-step actor-critic:

$$\begin{aligned}\theta_{t+1} &= \theta_t + \alpha\big(R_{t+1} + \gamma\hat{v}(S_{t+1}, w) - \hat{v}(S_t, w)\big)\frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}\\ &= \theta_t + \alpha\delta_t\frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}\end{aligned} \tag{92}$$

  - Coupled with gradient ascent for value function.
  - Actor part: take action and update value function.
  - Critic part: use value function to update policy.
  - Algorithm: see p. 274.

9.3. Continuing tasks

To complete.

- One step stationary cost:

$$J(\theta) = \lim_{t\to\infty} E[R_{t+1}|S_t \sim \mu_\pi, A_t \sim \pi] \tag{93}$$

- See algorithm p. 277.