Chapter 1: Markov processes (MP)

- State: $S_t \in \mathcal{S}$ ($\mathcal{S}$: state space) $\quad t = 0, 1, 2, \ldots$
- Markov evolution:

$$P(S_{t+1} = s' \mid S_t = s, S_{t-1}, S_{t-2}, \ldots, S_0) = P(S_{t+1} = s' \mid S_t = s)$$

(S_{t+1}: future; S_t: present; $S_{t-1}, S_{t-2}, \ldots, S_0$: past)

  - Future independent of past given present
  - Markov chain: $S_0 \to S_1 \to S_2 \to \cdots$
- Transition probability:

$$p(s' \mid s) = P(S_{t+1} = s' \mid S_t = s)$$

  - Assumed homogeneous (stationary, time-independent)
  - Normalization: $\sum_{s'} p(s' \mid s) = 1 \quad \forall\, s$
- Transition matrix: P_{ij} (row) $= p(j \mid i) = P(i \to j)$ (column)
  - $|\mathcal{S}| \times |\mathcal{S}|$ matrix
  - $\sum_j P_{ij} = 1 \quad \forall\, i$ (row sums = 1)
  - $0 \le P_{ij} \le 1$ (stochastic matrix)
- Probability distribution: $p_t(s) = P(S_t = s)$
- Probability vector: p_t, $p_{t,i} = p_t(i)$
  - $|\mathcal{S}|$-dim vector
  - Row vector

· Evolution equation :

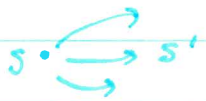
$$p_{t+1}(s') = \sum_s p(s'|s)\, p_t(s)$$

· Matrix notation: $p_{t+1} = p_t P$ right product

$$(\,-\,p_{t+1}\,-\,) = (\,-\,p_t\,-\,)\begin{pmatrix} & P & \end{pmatrix}$$

$$(p_{t+1})_j = \sum_i (p_t)_i\, P_{ij}$$

· Expectations :



$$E[f(S_{t+1})\,|\,S_t = s] = \sum_{s'} f(s')\, p(s'|s)$$

· Matrix notation :

$$F_i = E[f(S_{t+1})\,|\,S_t = i] = \sum_j f(j)\, p(j|i)$$

$$\begin{pmatrix} \\ F \\ \\ i \end{pmatrix} = \begin{pmatrix} & P & \end{pmatrix}\begin{pmatrix} \\ f \\ \\ j \end{pmatrix} \qquad = \sum_j P_{ij}\, f(j)$$

$$\Rightarrow \quad F = Pf \qquad \text{left product}$$

· Marginalization: $p(x) = \sum_y p(x,y)$

$$p(x|z) = \sum_y p(x,y|z)$$

· Chain rule : $p(x,y) = p(x|y)\,p(y) = p(y|x)\,p(x)$

$$p(x,y|z) = p(x|y,z)\,p(y|z)$$

· Deterministic transition: $p(s'|s) = \begin{cases} 1 & \text{for same } s' \\ 0 & \text{otherwise} \end{cases} = \delta_{s,s'(s)}$

$s \to s'$

Chapter 2: Markov reward processes (MRP)

- State: S_t $\quad t = 0, 1, \ldots$
- Reward: $R_t \in \mathbb{R}$ $\quad t = 0, 1, \ldots$
  - Cost / utility at time t
  - RV correlated with transitions $S_t \to S_{t+1}$
  - Sometimes: fct of s, s' $\quad r(s, s')$
- Transition probability (dynamics):

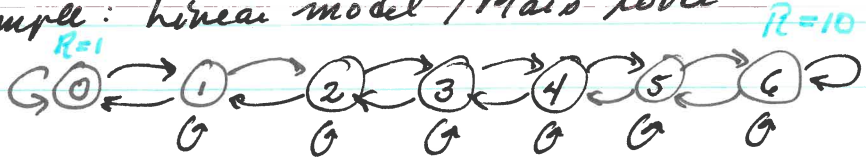
$$p(s', r \mid s) = P(S_{t+1} = s', R_{t+1} = r \mid S_t = s)$$

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

  - Reward depends on s, s'
  - Assumed homogeneous (time-independent, stationary)
  - No actions yet: dynamics + rewards
- Reward probability: $p(r \mid s) = \sum_{s'} p(s', r \mid s)$
- Transition probability: $p(s' \mid s) = \sum_{r} p(s', r \mid s)$ pure dynamics
- Expected state reward:

$$\rho(s) = E[R_{t+1} \mid S_t = s] = \sum_{r} r \, p(r \mid s)$$

  - One-time reward from state s
- Example: linear model / Mars rover



R deterministic w/r S_t

$\rho(0) = 1 \quad \rho(6) = 10$

· Return: $G_t = R_{t+1} + R_{t+2} + \cdots + R_T$

present: t — future: $t+1, t+2, \ldots, T$

$S_t \to S_{t+1} \to S_{t+2} \to \cdots S_T$ (rewards $R_{t+1}, R_{t+2}, \ldots, R_T$)

S_t: G_t — S_T: final time

· Total reward / cost to go from time t

· Episodic process: T finite

Process stops / terminates

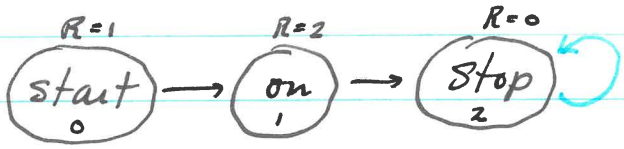
Calculate backward

$\Rightarrow G_T = 0 \quad G_{T-1} = R_T \quad G_{T-2} = R_{T-1} + R_T \quad \cdots$

· Continuing process: $T = \infty$

Process goes on and on

· Example:



$0 \to 1 \to 2$ $\quad 1 \to 2$ $\quad 2$ $\quad$ Episodic

$0 \to 1 \to 2 \to 2 \to 2 \to \cdots$ $\quad$ Continuing

· Discounted return: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 G_{t+3} + \cdots$

$$= \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}$$

· Discount rate $\gamma \in [0,1]$ — Balances present / future rewards

· $G_t < \infty$ for $\gamma \in [0,1)$

· Myopic: $\gamma \approx 0$ — $G_t = R_{t+1}$ for $\gamma = 0$ immediate reward

· Far-sight: $\gamma \approx 1$

· Iteration:

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

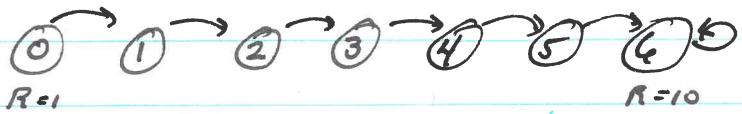
- Value function: $v(s) = E[G_t \mid S_t = s]$

$$S_t \rightarrow \underbrace{R_{t+1} \rightarrow S_{t+1}}_{G_t} \rightarrow \gamma R_{t+2} \rightarrow S_{t+2} \rightarrow \gamma^2 R_{t+3} \rightarrow S_{t+3} \rightarrow \cdots$$

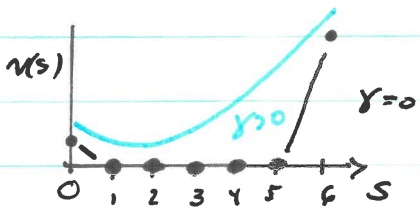
$S_t = s$

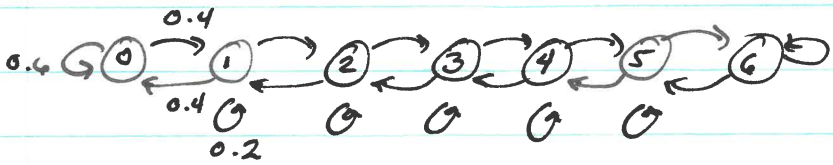
- State value fct: Expected return / cost to go from state s
- Doesn't depend on time t
  - Stationary MP
  - Continuing process / infinite return
- $\gamma = 0$: $v(s) = \rho(s)$ cost when leaving s

- Example: linear model



deterministic moves / transitions





$v(s)$? See CW

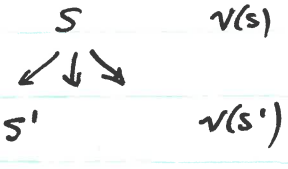
- Bellman equation:

$$v(s) = E[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

$$= E[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]$$

$$S_t \rightarrow \underbrace{R_{t+1} \rightarrow S_{t+1}}_{G_t} \rightarrow \underbrace{R_{t+2} \rightarrow S_{t+2}}_{G_{t+1}} \rightarrow \cdots$$

$S_t = s$



$v(s)$ $v(S_{t+1})$

· Explicit expectation:

$$V(s) = \sum_r r\, p(r|s) + \gamma \sum_{s'} V(s')\, p(s'|s)$$

$$= \sum_{r,s'} r\, p(s', r|s) + \gamma \sum_{s',r} V(s')\, p(s', r|s)$$

$$= \rho(s) + \gamma (PV)(s) \qquad P_{ij} = p(j|i)$$

· Matrix notation: $V = \rho + \gamma P V$ $\qquad V_i = V(i)$

- $V_i = V(i)$ column vector $|\mathcal{S}| \times 1$
- $\rho_i = \rho(i)$ " "
- γ scalar
- $P_{ij} = p(j|i)$ $|\mathcal{S}| \times |\mathcal{S}|$ matrix

· Solution: $V = (I - \gamma P)^{-1} \rho$

- Always exists for $\gamma \in [0, 1)$
- Unique solution

· Bellman operator: $T(V) = \rho + \gamma P V$

$V = T(V)$ Value function = fixed pt of T

· Example: linear model.

- Write ρ, P (7×1, 7×7)
- Solve numerically.

· Two ways to define MRPs:

① $p(s', r \mid s)$

↳ $p(r \mid s)$

↳ $\rho(s) = E[R_{t+1} \mid S_t = s]$ $\rho(s)$

↳ $p(s' \mid s)$

↳ P P

$$v(s) = E[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]$$

$$v = \rho + \gamma P v$$

Chapter 3: Markov decision processes (MDP)

SB: Chap 3

3.1. Model

- Environment state: $S_t \in \mathcal{S}$ $\quad t = 0, 1, \ldots$
  - System to control
  - Info for making decision

- Agent state: $A_t \in \mathcal{A}(s)$ $\quad t = 0, 1, \ldots$
  - Controller
  - Action state / decision — available actions from given state
  - State space can depend on current state
  - Simplification: $\mathcal{A}(s) = \mathcal{A}$ $\forall s$

- Reward: $R_t \in \mathbb{R}$
  - Signal from system / environment
  - Defines good (+) / bad (−) actions
  - Zero / baseline not important
  - RV correlated with states / actions
  - Sometimes expressed as $r(s, a, s')$

- Transition probability (dynamics):

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

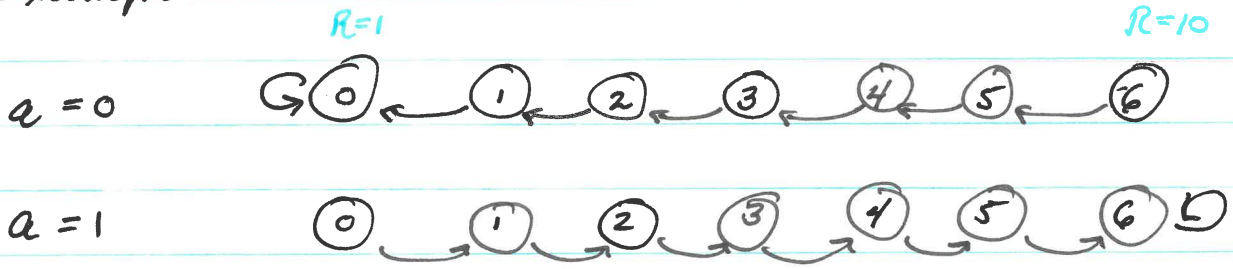
Reward / System / Agent

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

$$A_0 \nearrow \quad A_1 \nearrow$$

- Assumed homogeneous (time-independent, stationary)

· Example:



- 2 deterministic actions: left, right
- Deterministic reward given state and doesn't depend on action (same for a=0,1)

· State transition probability:

$$p(s'|s,a) = \sum_r p(s',r|s,a)$$

Transition matrix for each a

$p_a(j|i)$ P_a

· Reward probability:

$$p(r|s,a) = \sum_{s'} p(s',r|s,a)$$

(a: action)

compare with MRP

· Expected reward: $\rho(s,a) = E[R_{t+1} | S_t = s, A_t = a]$

$s = S_t \longrightarrow S_{t+1}$ (R_{t+1})

$a = A_t$

$$\rho(s,a) = \sum_r r\, p(r|s,a)$$

$$= \sum_{r,s'} r\, p(s',r|s,a)$$

- One-step reward from state s and action a
- Doesn't depend on time (stationary)

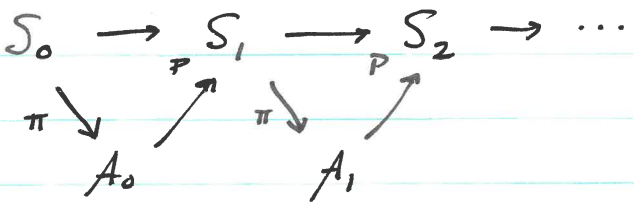
3.2. Policy

- How to choose action / control / decision at given state?

- Policy transition probability:

$$\pi(a|s) = P(A_t = a \mid S_t = s)$$

(action, state)



- Mapping state → action
- Probabilistic policy: actions can be random (noise in environment, exploration)
- Assumed homogeneous (time-independent, stationary)

- Deterministic policy: $\pi(s) = a$ (one state, one action)

- Joint probability: $p(s', r, a|s) = p(s', r|s, a)\,\pi(a|s)$

- System transition probability:

$$P_\pi(s'|s) = \sum_a p(s'|s,a)\pi(a|s) \quad \text{(Average over actions)}$$

$$= \sum_{a,r} p(s', r|s,a)\,\pi(a|s)$$

- Effective dynamics under policy π
- Transition matrix: P_π $\quad (P_\pi)_{ij} = p_\pi(j|i)$

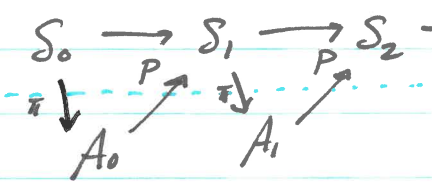
$$S_0 \xrightarrow{P_\pi} S_1 \xrightarrow{P_\pi} S_2 \xrightarrow{P_\pi} \cdots$$

· Comparison:

· No control: $S_0 \xrightarrow{P} S_1 \xrightarrow{P} S_2 \to \cdots$ $\quad p(s'|s)$

· Open-loop control: $S_0 \xrightarrow[A_0]{P} S_1 \xrightarrow[A_1]{P} S_2 \to \cdots$ $\quad p(s'|s,a)$

· Closed-loop control: feedback



$p(s'|s,a)$ $\quad \pi(a|s)$

· Reduced dynamics: $S_0 \xrightarrow{P_\pi} S_1 \xrightarrow{P_\pi} S_2 \xrightarrow{P_\pi} \cdots$ $\quad p_\pi(s'|s)$

· Expected reward under policy π:

$$\rho_\pi(s) = E_\pi[R_{t+1} | S_t = s]$$

$$= E_\pi \, E[R_{t+1} | S_t = s, A_t]$$

$$= E_\pi[\rho(s, A_t)]$$

$$\rho_\pi(s) = \sum_a \rho(s,a)\, \pi(a|s) \quad \text{average over actions}$$

$$= \sum_{a,r} r \; p(r|s,a)\, \pi(a|s)$$

$$= \sum_{a,r,s'} r \; p(s', r|s,a)\, \pi(a|s)$$

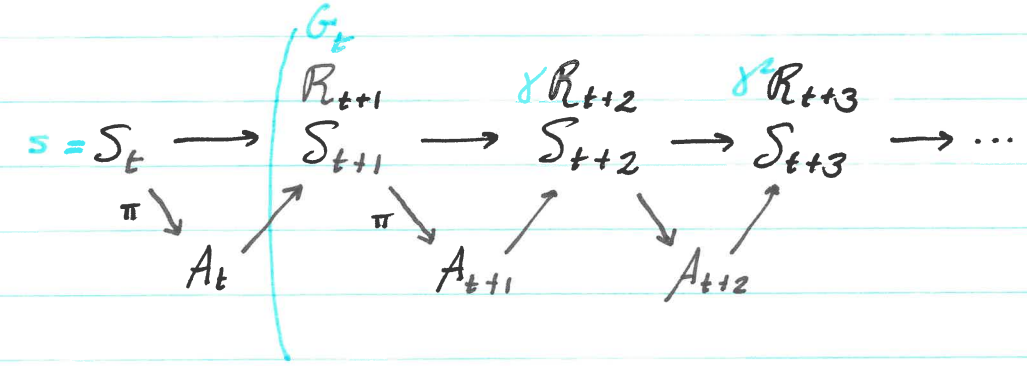
· One-time (step) reward from state s under policy π
· Doesn't depend on time (stationary)
· $\rho(s,a)$ doesn't depend on π
· Deterministic: $\rho_\pi(s) = \rho(s, \pi(s))$

3.3. Value Functions

· State value function:

$$v_\pi(s) = E_\pi[G_t \mid S_t = s]$$

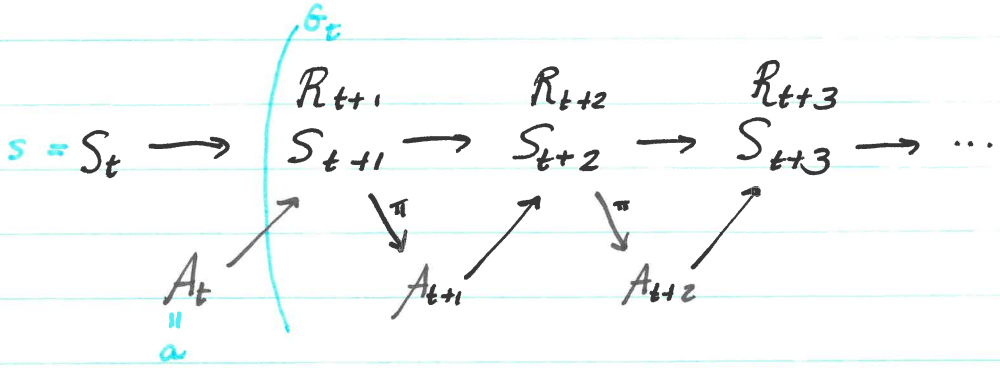
(return)



· Future return / cost to go from state s + policy π
· Long term value of s under π
· Function of $\pi(\cdot|s)$
· Doesn't depend on time t (stationary dynamics, infinite return)

· State-action value function:

$$q_\pi(s,a) = E_\pi[G_t \mid S_t = s,\ A_t = a]$$



· Return / cost to go if action a taken from state s
· Value of action a when in s if policy π followed after
· Doesn't depend on time t

· Connection :

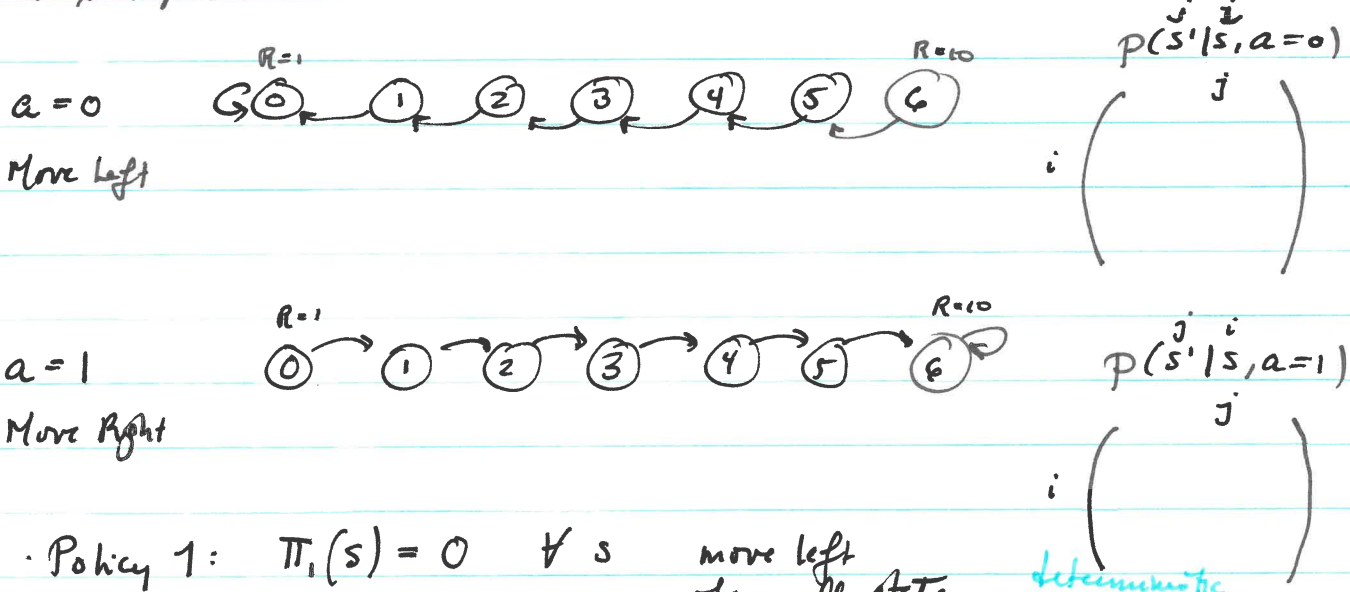
$$v_\pi(s) = E_\pi[\, q_\pi(s, A_t) \mid S_t = s \,] \quad \text{average over actions}$$

$$= \sum_a q_\pi(s,a)\, \pi(a|s)$$

$$v_\pi(s) = q_\pi(s, \pi(s)) \quad \text{for deterministic policy}$$

$a = \pi(s)$

· Example :



· Policy 1: $\pi_1(s) = 0 \quad \forall s$ move left from all state deterministic

· Policy 2: $\pi_2(s) = 1 \quad \forall s$ move right "

· Policy 3: $\pi_3 = \frac{1}{2}\pi_1 + \frac{1}{2}\pi_2$

random policy/control

CW: Calculate v_{π_1}, v_{π_2}, v_{π_3}

· Note : $\rho_\pi(s) = 1$ or $10 \quad s=0, s=6 \quad \forall \pi$ reward only depends on S

$\rho(s,a) = 1$ or $10 \quad s=0, s=6 \quad \forall a$

· Note : 2 actions per state $\Rightarrow$ 2^7 possible policies

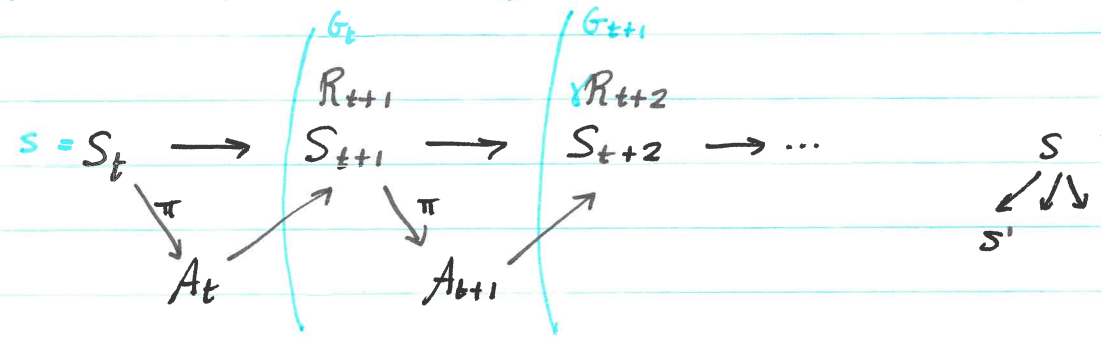
3.4. Bellman equations (BE)

- Return: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = R_{t+1} + \gamma G_{t+1}$

- Bellman equation for value function:

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

Compare with MRP



$$v_\pi(s) = \sum_{a} p(s,a)\,\pi(a|s) + \gamma \sum_{s',a} v_\pi(s')\, p(s'|s,a)\,\pi(a|s)$$

$$= \sum_{s',r,a} r\, p(s',r|s,a)\,\pi(a|s) + \gamma \sum_{s',r,a} v_\pi(s')\, p(s',r|s,a)\,\pi(a|s)$$

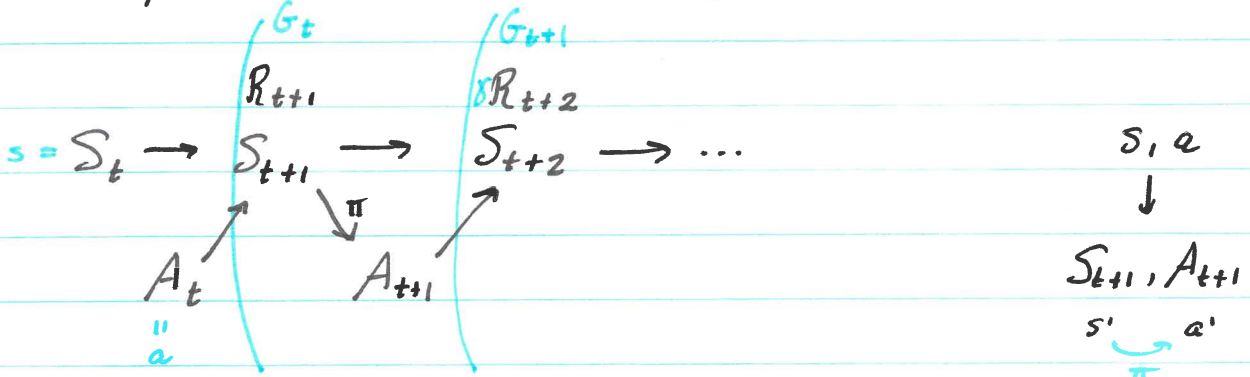
- Value (s) = immediate reward (s) + discounted value of successor state (s')

- Linear equation: $v_\pi = \rho_\pi + \gamma P_\pi v_\pi$

- Solution: $v_\pi = (I - \gamma P_\pi)^{-1} \rho_\pi$

· BE for action-value function:

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



$q_\pi(s,a)$ $\quad$ $q_\pi(S_{t+1}, A_{t+1})$

$$q_\pi(s,a) = p(s,a) + \gamma \sum_{a',s'} q_\pi(s',a')\, p(s'|s,a)\, \pi(a'|s')$$

$$= \sum_{s',r} r\; p(s',r|s,a) + \gamma \sum_{a',s',r} q_\pi(s',a')\, p(s',r|s,a)\, \pi(a'|s')$$

· Value (s,a) = immediate reward (s,a) + discounted value of successor state, action (s',a')

· Linear equation for "matrix" q_π

· Two ways of defining MDPs

① ②

$p(s', r \mid s, a)$

↳ $p(r \mid s, a)$

↳ $\rho(s,a) = E[R_{t+1} \mid S_t = s, A_t = a]$ ρ

↳ $p(s' \mid s, a)$

↳ P_a P_a

$\pi(a \mid s)$

↳ $\rho_\pi(s) = E_\pi[R_{t+1} \mid S_t = s]$ ρ_π

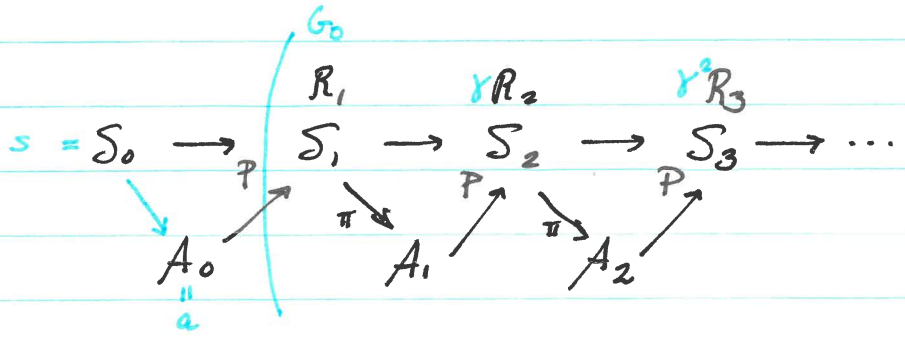
↳ $p_\pi(s' \mid s)$

↳ P_π P_π

$$v_\pi = \rho_\pi + \gamma P_\pi v_\pi$$

$$q_\pi = \rho + \gamma (P_a \pi q_\pi)$$

3.5 Optimal policies



Dynamics	Policy / control	Reward
$p(s' \mid s, a)$	$\pi(a \mid s)$	$p(s', r \mid s, a)$

$$v_\pi(s) = E_\pi[G_t \mid S_t = s] = E_\pi[G_0 \mid S_0 = s]$$

$$q_\pi(s,a) = E_\pi[G_t \mid S_t = s, A_t = a] = E_\pi[G_0 \mid S_0 = s, A_0 = a]$$

infinite horizon return

- Goal: Find π with max expected return
- Optimal value function:

$$v_*(s) = \max_\pi v_\pi(s)$$

  - Best actions / policies from s
  - At least one solution

- Optimal action-value function:

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

  - Optimal policy for successor states from s, a
  - Same solution π_* as v_*

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

- **General results** (Silver: L2): For any finite MDPs
  - There exists at least one optimal policy π_*
  - All optimal π_* achieve v_*
  - " " " " q_*
  - Optimal policy is deterministic (one action per state)
  - " " " stationary

$$\Rightarrow \quad \pi_*(s) = a$$

- Optimal policy:

$$\pi_*(s) = \arg\max_a q_*(s,a) \quad = \text{best action from } s$$

- Optimal value function:

$$v_*(s) = v_{\pi_*}(s) = q_*(s, \pi_*(s)) \quad \text{(} \pi_*(s) \text{: best action)}$$

or

$$v_*(s) = \max_a q_*(s,a) \quad \text{(best action)}$$

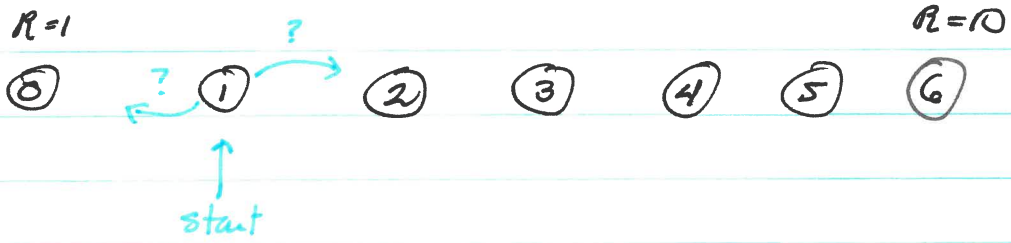
$$\pi_* \iff q_* \iff v_*$$

- Note: $v_\pi(s) = \sum_a q_\pi(s,a)\,\pi(a|s)$

See p. 3-6

$$= q_\pi(s, \pi(s)) \quad \text{deterministic action}$$

· Example:



$\gamma \approx 0$: Better to move left to get R=1 reward

$\gamma \approx 1$: " " " right " " R=10 "

Optimal policy: for each s, either left or right
$p(s'|s,a=0)$ $p(s'|s,a=1)$

π_* among 2^7 possible policies

3.6 Bellman optimality equations

See 3-12b

· Value function:

$$v_*(s) = \max_a \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right]$$

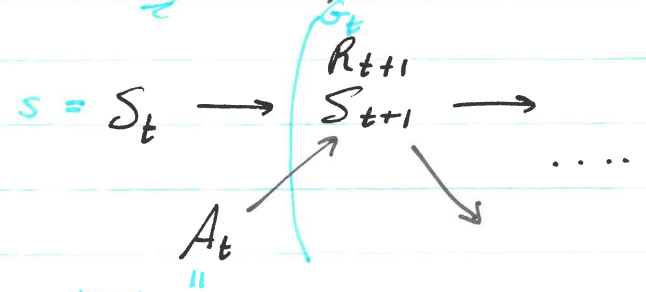
· Optimal from s = optimal action from s and then optimal policy after

· Expectation on one step — no E_π

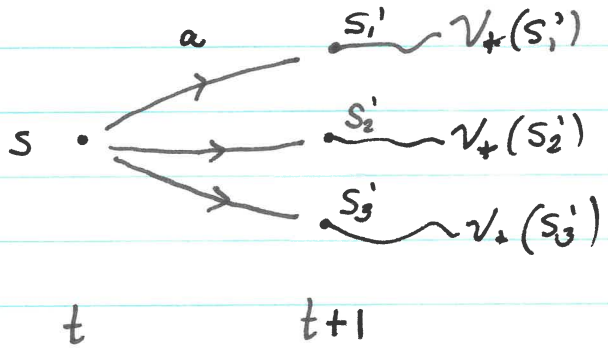
· max outside expectation — influences reward

· arg max defines $\pi_*(s)$

· Nonlinear equation because of max



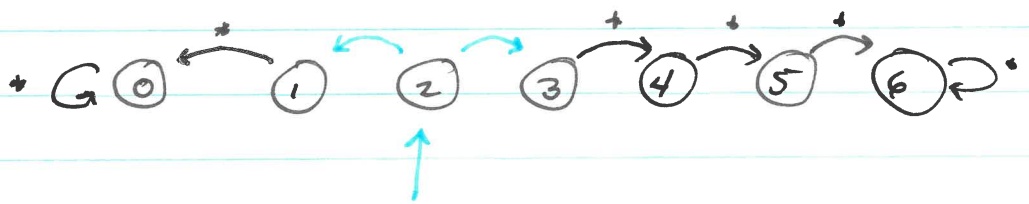
$$v_*(s) = \max_a \; p(s,a) + \gamma \sum_{s'} v_*(s') p(s'|s,$$



- Whatever action chosen at time t, rest of dynamics is optimal
- To be optimal at $S_t = s$, choose optimal action and then follow optimal policy from S_{t+1}

$$\Rightarrow \quad v_*(s) = \max_a \; E[\, R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a \,]$$

- Example:



· Action value function:

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

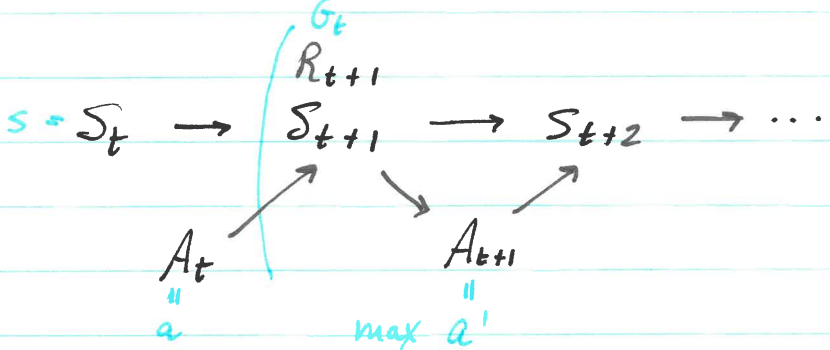
(a': action after)

$$= E\left[R_{t+1} + \gamma V_*(S_{t+1}) \mid S_t = s, A_t = a\right]$$

i.e.

$$q_*(s,a) = \rho(s,a) + \gamma \max_{a'} \sum_{s'} q(s';a')\, p(s'|s,a)$$

$$= \rho(s,a) + \gamma \sum_{s'} V_*(s')\, p(s'|s,a)$$



· Optimal from s,a = optimal after reaching S_{t+1} by taking optimal action after

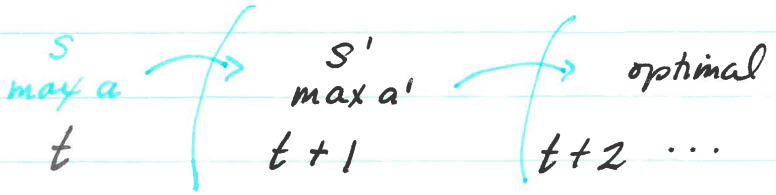
· Max inside – doesn't influence reward

· arg max defines $\pi_*(s')$

· Nonlinear equation because of max

· Bellman's optimality principle:

An optimal policy has the property that whatever the initial state / decision are, the remaining decisions constitute an optimal policy with regard to the state resulting from the first decision.



Chapter 4: Dynamic programming

SB: Chap 4

4.1 Policy evaluation

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

$$v_\pi = \rho_\pi + \gamma P_\pi v_\pi$$

$$\left(v_\pi\right) = \left(\rho_\pi\right) + \gamma \left(P_\pi\right)\left(v_\pi\right)$$

$$\Rightarrow \quad v_\pi = (I - \gamma P_\pi)^{-1} \rho_\pi$$

- Complexity: $O(|S|^3)$

- Iterative method: $v_0 \to v_1 \to \cdots \to v_\pi$

$$v_{k+1}(s) = E_\pi\left[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s\right]$$

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

Bellman backup

matrix product

$$= T_\pi(v_k)$$

$$T_\pi(v) = \rho_\pi + \gamma P_\pi v$$

Bellman operator

- Initial value v_0 arbitrary, except $v_0(\text{terminal state}) = 0$
- v_π fixed point of T_π
- Convergence: $v_k \to v_\pi$ as $k \to \infty$
- T_π is a contraction
- Complexity: $|S|^2 \times M$

matrix product # iterations

- Not exact v_π for finite # iterations

4.2. Policy iteration

- Consider deterministic policies $\pi(s)$

V_π, q_π

- Choose $a = \pi(s)$ — on policy — $V_\pi(s) = q_\pi(s, \pi(s))$
- Choose $a \neq \pi(s)$ — off policy — $V_\pi(s) \neq q_\pi(s, a)$

- Policy improvement theorem: Let π and π' such that

$$q_\pi(s, \pi'(s)) \geq q_\pi(s, \pi(s)) = V_\pi(s)$$

(π': off policy; π: on policy)

for all $s \in \mathcal{S}$. Then π' is as good or better than π in the sense that

$$V_{\pi'}(s) \geq V_\pi(s)$$

($V_{\pi'}$: better value under π')

for all $s \in \mathcal{S}$.

- Greedy policy selection:

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

  - Equation for updating π
  - Fixed point: π_* for q_* (optimal policy)
  - Strict improvement unless optimal

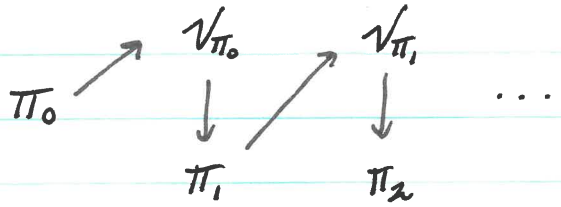
- ε-greedy policy selection:

$$\pi'(s) = \begin{cases} \arg\max_a q_\pi(s, a) & \text{Prob } 1-\varepsilon \\ \mathcal{U}\{\mathcal{S}\} & \text{Prob } \varepsilon \end{cases}$$

($\mathcal{U}\{\mathcal{S}\}$: exploration)

  - No strict improvement

· Policy iteration algorithm



Policy evaluation (Direct or iterative)

Policy improvement

· Convergence: $\pi_k \to \pi_*$, $v_k \to v_*$, $q_k \to q_*$ as $k \to \infty$

· Note: Policy evaluation for q_π

$$q_\pi(s,a) = \rho(s,a) + \gamma \sum_{a',s'} q_\pi(s',a')\,\pi(a'|s')\,p(s'|s,a)$$

$$= \rho(s,a) + \gamma \sum_{s'} q_\pi(s',\pi(s'))\,p(s'|s,a) \quad \text{(on policy)}$$

$$= \rho(s,a) + \gamma \sum_{s'} v_\pi(s')\,p(s'|s,a)$$

$$\to \quad q_\pi = \rho + \gamma P_a v_\pi$$

· Note:

Planning	Learning
· MDP model known	· MDP model unknown
· Solve to get v_π	· Estimate v_π by experience/sampling
· Solve to get v_*, π_*	· Converge to v_*, π_*
v_π, v_*	q_π, q_*

4.3. Value iteration

· Bellman optimality equation:

$$v_*(s) = \max_a \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right]$$

$q_*(s,a)$

$$v_*(s) = \max_a \left[\rho(s,a) + \gamma (P_a v_*)(s)\right]$$

· Bellman optimality operator:

$$T_*(\cdot) = \max_a \left[\rho + \gamma (P_a \cdot)\right]$$

↑ $T_*(\cdot)$ ↑ $(P_a \cdot)$

· Value iteration algorithm:

$$v_0 \longrightarrow v_1 = T_*(v_0) \longrightarrow v_2 = T_*(v_1) \longrightarrow \cdots$$

$$v_{k+1}(s) = \max_a \mathbb{E}\left[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a\right]$$

↑ $k+1$ ↑ k

$$v_{k+1}(s) = \max_a \left[\rho(s,a) + \gamma \sum_{s'} v_k(s') p(s'|s,a)\right]$$

$$= \max_a \left[\rho(s,a) + \gamma (P_a v_k)(s)\right]$$

- One step of iterative policy evaluation
- Max defines "optimal" policy at ~~stage~~ k+1 (policy improvement)
- Greedy policy improvement (can also use ε-greedy)
- Convergence: $v_k \to v_*$ as $k \to \infty$

4.4. Action value iteration

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

time $t+1$

update

previous estimate, $v_k(S_{t+1})$

$$q_{k+1}(s,a) = \rho(s,a) + \gamma \max_{a'} \sum_{s'} q_k(s',a') \, p(s'|s,a)$$

- Max defines policy improvement at stage $k+1$
- Fixed point: q_* — Bellman optimality equation
- Convergence: $q_k \to q_*$ as $k \to \infty$

- Policy improvement: $\pi_{k+1}(s) = \arg\max_a q_k(s,a)$

- Variant:

$$q_k(s,a) = E\left[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a \right]$$

$q_k \leftarrow v_k$

$$q_k(s,a) = \rho(s,a) + \sum_{s'} v_k(s') \, p(s'|s,a)$$

$$= \rho(s,a) + \gamma \left(P_a v_k \right)(s)$$

action a

- Max included in v_k

Chapter 5: Temporal difference methods (TD)

Planning	learning
· MDP model known/given	· MDP model unknown
· Solve to get V_*, q_*, π_*	· Estimate V_*, q_*, π_*
· Bellman opt. eq.	· Sampling
· Policy iteration	· Exploration
· Value iteration	· RL

Prediction	Control
· Predict effect of policy	· Find optimal policy
· Policy evaluation V_π	· Find V_*, q_*

needed for learning

· Idea of TD:
  · Use observed / visited states to estimate V_* iteratively
  · Sampling / exploration informs estimation
  · Now states / actions depend on estimated V_*

feedback

$$V_0 \to V_1 \to V_2 \to \cdots \to V_*$$

$\delta R, \delta S$

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

$$V_0 \hookrightarrow V_1 \nearrow \hookrightarrow V_2 \nearrow$$

δV

See MC course

· Stochastic approximation:
  · Iteration, map: $x_{n+1} = T(x_n)$

$x_n \to x^*$ $\quad x^* = T(x^*)$ attractive fixed point

  · Stochastic iteration: $X_{n+1} = T(X_n)$

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

$a_n > 0$
$\sum a_n = \infty$
$\sum a_n^2 < \infty$

TD error $\to 0$

5.1 TD(0)

- Policy iteration (prediction, not control)
- Bellman equation:

$$V_\pi(s) = E_\pi[R_{t+1} + \gamma V_\pi(S_{t+1}) \mid S_t = s]$$

$$V_\pi = \rho_\pi + \gamma P_\pi V_\pi \qquad = T(V_\pi)$$

Bellman operator

- TD(0) iteration:

TD error

$$V_{k+1}(S_t) = V_k(S_t) + \alpha_k [R_{t+1} + \gamma V_k(S_{t+1}) - V_k(S_t)]$$

S

$$= (1-\alpha_k) V_k(S_t) + \alpha_k [R_{t+1} + \gamma V_k(S_{t+1})]$$

S'

do nothing — policy evaluation — Bellman iteration update

- SA version of value iteration
- SA:

expectation

$$E_\pi[R_{t+1} + \gamma V_\pi(S_{t+1}) \mid S_t = s]$$

$$\Big\downarrow$$

RV

$$R_{t+1} + \gamma V_\pi(S_{t+1}) \quad \text{for transition } S_t \to S_{t+1}$$

S S'

- 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:

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

· 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_n
- Use all states: s, a, r, s', a'
- On-policy: policy used to update Q = 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)

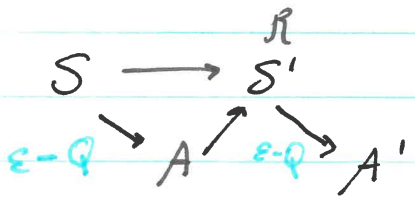
Get R, S' from action A

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

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

$S \leftarrow S'$

$A \leftarrow A'$



5.3. Q-learning

· Off-policy control situation for estimating q_*

· Q-learning situation:

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

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

Find max over action

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

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

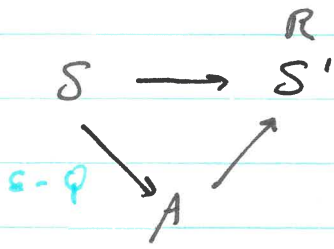
· Off-policy: Action selected
≠
action used to update Q

· 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 step:
    Choose A from S using ε-greedy from Q ← Define policy
    Get S', R from A
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left(R + \gamma \max_a Q(S',a) - Q(S,A)\right)$
    $S \leftarrow S'$



$\max_a Q(S', a)$

for update

- Maximization bias:

max estimated q ≠ max in expectation

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

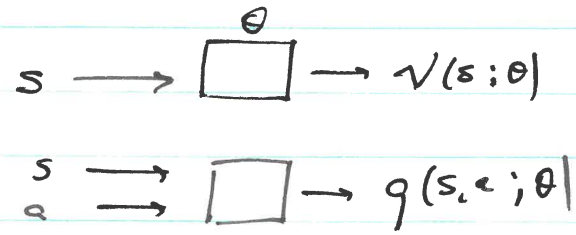
- Choosing largest value necessarily above expectation

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

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

Conclusions – to go further

- Represent $V(s)$ $|\mathcal{S}|$ — $q(s,a)$ $|\mathcal{S}| \times |\mathcal{A}|$ — SB Chaps 9–12
  - Parameterizations / Projections: $V(s;\theta)$, $q(s,a;\theta)$
  - Neural networks: deep RL



  - Adapt TD(0), Sarsa, Q-learning to update parameters θ

- Estimation, sampling: $Q_n \to Q_{n+1}$, $V_n \to V_{n+1}$ — SB Chaps 5–7
  - Use larger 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
  - Actn-critic methods
    - π ↔ $q(s,a)$

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