Markov Decision Process

Markov State → **State Transition**

$P_{\{s_{t+1}|s_t\}} = P_{\{s_{t+1}|s1,\cdots,s_t\}}$ $P_{ss'} = P_{\{s_{t+1}=s'|s_t=s\}}$

Markov Process

+

Reward Function + **Return**

$R_s = E[R_{t+1}|S_t = s]$ $G_t = R_{t+1} + \gamma R_{t+2} + R_{t+3} + \cdots$

⇩

Value Function

$V(s) = E[G_t|S_t = s]$ Bellman → $\begin{cases} R_{t+1} \\ \gamma V(S_{t+1}) \end{cases}$ → $V(s) = \mathcal{R}_s + \gamma \sum_{s' \epsilon s} P_{ss'} v(s')$

Markov Reward Process

+

Action and Policy

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

⇩

Value Function

$V_\pi(s) = E_\pi[G_t|S_t = s]$ $q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a]$ Bellman → $V_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s,a)$

$q_\pi(s,a) = \mathcal{R}_s^a + \gamma P_{ss'}^a v_\pi(s')$

The optimal value functions are recursively related by the Bellman optimality equations.

$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \epsilon s} P_{ss'}^a v_*(s')$

$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \epsilon s} P_{ss'}^a \max_{a'} q_*(s',a')$

⇒ How to solve q_* and v_* ⇒

Value Iteration
Policy Iteration
Q-learning
Sarsa

MDP Problem: Learning / Planning ⇒ Model-based / Model-free ⟹ Problem ⟹ Prediction and Control: P is the based of C

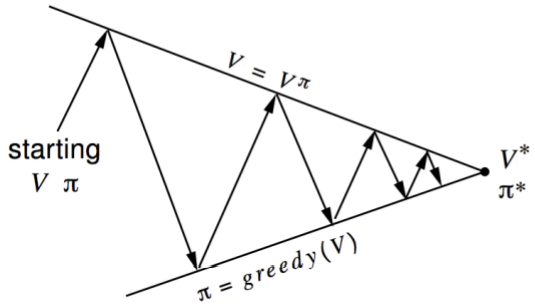
v_π v_* π_*

⇩

Model-based:

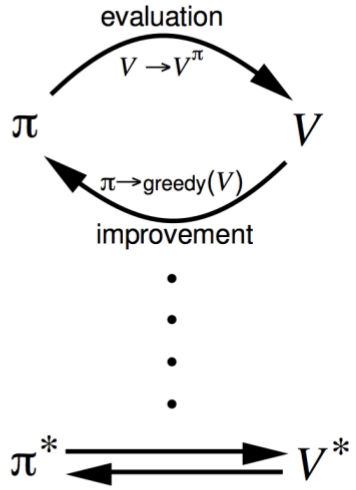
- DP
  - Prediction:
    - Input: MDP and policy π
    - Output: value function v_π
  - Control:
    - Input: MDP
    - Output: optimal value function v_*
      optimal policy π_*
- Algorithms

Problem	Bellman Equation	Algorithm
Prediction	Bellman Expectation Equation	Iterative Policy Evaluation
Control	Bellman Expectation Equation + Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality Equation	Value Iteration



Policy evaluation Estimate v_π
Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$
Greedy policy improvement



A control process

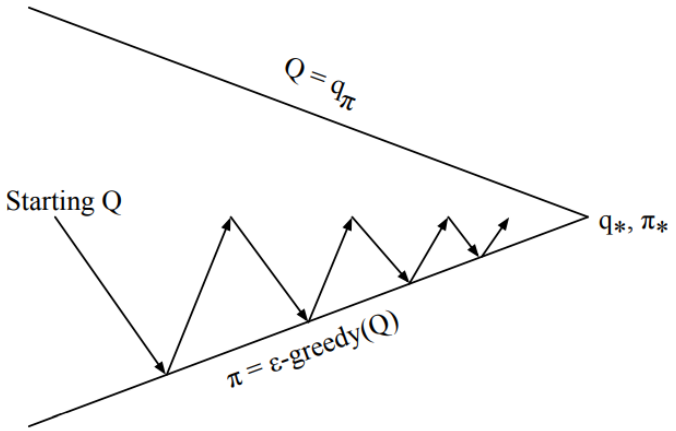
Model-free:

Prediction:
Estimate the value function of an unknown MDP

- Monte-Carlo
  - First-Visit Monte-Carlo Policy Evaluation
  - Every-Visit Monte-Carlo Policy Evaluation
- Temporal-Difference
  - TD(0)
  - TD(λ)
    - Forward-view
    - Backward View

Control:
Optimise the value function of an unknown MDP

- On-policy: MC control , TD control ,
- Off-policy: Q-learning

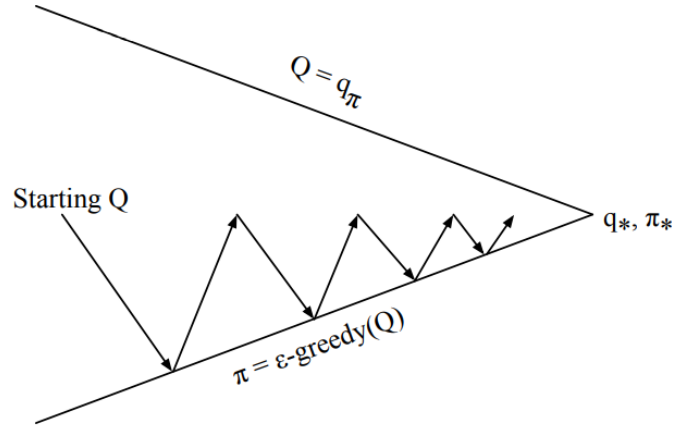


Every episode:

Policy evaluation Monte-Carlo policy evaluation, $Q \approx q_\pi$

Policy improvement ϵ-greedy policy improvement

MC control



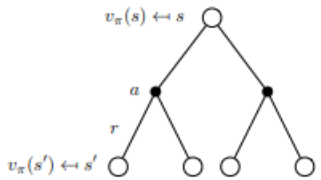
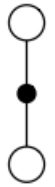
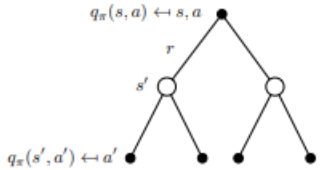
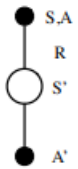
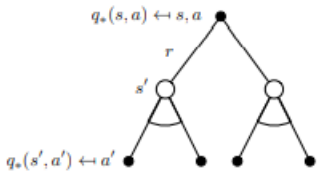
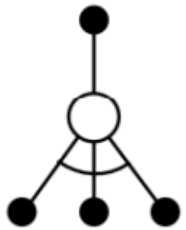
Every time-step:

Policy evaluation Sarsa, $Q \approx q_\pi$

Policy improvement ϵ-greedy policy improvement

Sarsa

Relationship Between DP and TD

	Full Backup (DP)	*Sample Backup (TD)*
Bellman Expectation Equation for $v_\pi(s)$	 Iterative Policy Evaluation	 TD Learning
Bellman Expectation Equation for $q_\pi(s, a)$	 Q-Policy Iteration	 Sarsa
Bellman Optimality Equation for $q_*(s, a)$	 Q-Value Iteration	 Q-Learning

So what we've been talking about is the same thing: "discrete case" or called lookup table

⇩

Value Function Approximation (Calculate w)

- a. Linear combinations of features: GD/LS
- b. Neural network
- c. Decision tree
- d. Nearest neighbour
- e. Fourier / wavelet bases
- ...

+ training method

- a. Incremental method
- b. Batch method

	"Compound"	Algorithm
Prediction →	GD + a	MC(on/off), TD(0)(on),
	LS + b	MC(on/off), LSMC(on/off), TD(on), LSTD(on/off)
	b + b	DQN
Control →	GD + a	MC, Sarsa, Gradient Q-learning
	LS + b	MC, Sarsa, LSPI

What we do is using Approximate in place of "previous V_π , q_π " and using training method to get w

Other form of table

Prediction

GD

On/Off-Policy	Algorithm	Table Lookup	Linear	Non-Linear
On-Policy	MC	✓	✓	✓
	TD(0)	✓	✓	✗
	TD(λ)	✓	✓	✗
Off-Policy	MC	✓	✓	✓
	TD(0)	✓	✗	✗
	TD(λ)	✓	✗	✗

LS

On/Off-Policy	Algorithm	Table Lookup	Linear	Non-Linear
On-Policy	MC	✓	✓	✓
	LSMC	✓	✓	-
	TD	✓	✓	✗
	LSTD	✓	✓	-
Off-Policy	MC	✓	✓	✓
	LSMC	✓	✓	-
	TD	✓	✗	✗
	LSTD	✓	✓	-

Control

Algorithm	Table Lookup	Linear	Non-Linear
Monte-Carlo Control	✓	(✓)	✗
Sarsa	✓	(✓)	✗
Q-learning	✓	✗	✗
Gradient Q-learning	✓	✓	✗

(✓) = chatters around near-optimal value function

Algorithm	Table Lookup	Linear	Non-Linear
Monte-Carlo Control	✓	(✓)	✗
Sarsa	✓	(✓)	✗
Q-learning	✓	✗	✗
LSPI	✓	(✓)	-

(✓) = chatters around near-optimal value function