Reinforcement Learning

Outline

- Model-based RL
- Exploration vs exploitation in RL
- Model-free RL
  - Monte-Carlo methods
  - Temporal difference learning
- Issues with terminology

Outline

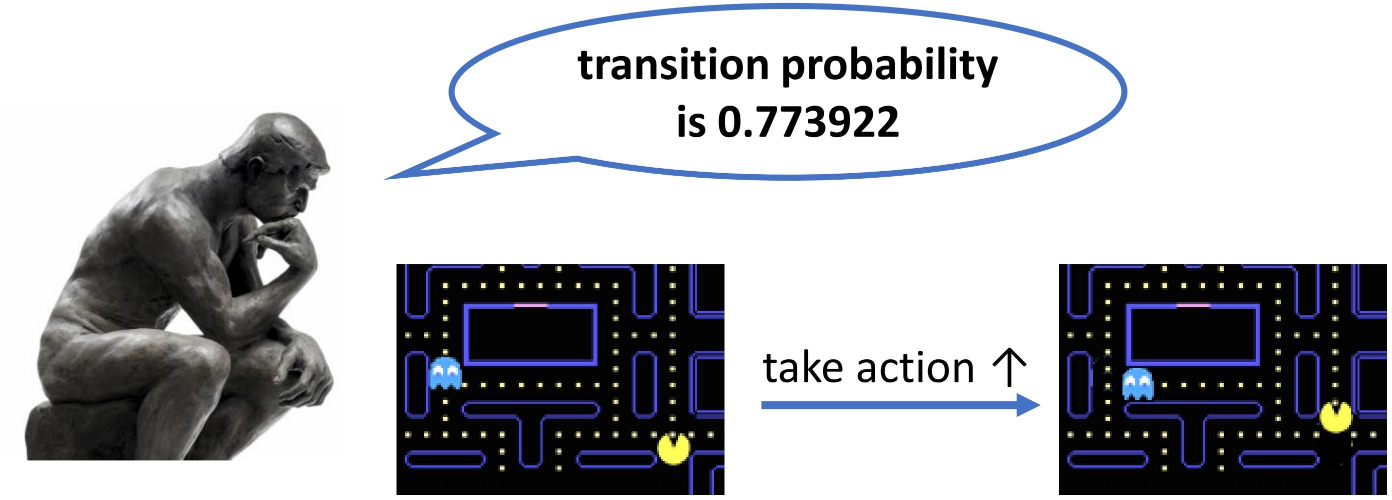
- **Model-based RL**
- Exploration vs exploitation in RL
- Model-free RL
    - Monte-Carlo methods
    - Temporal difference learning
- Issues with terminology

Model-based RL

- Previous lecture: given full knowledge of an MDP, how to find its optimal policies
- What if the agent does not have such full knowledge?

Model-based RL

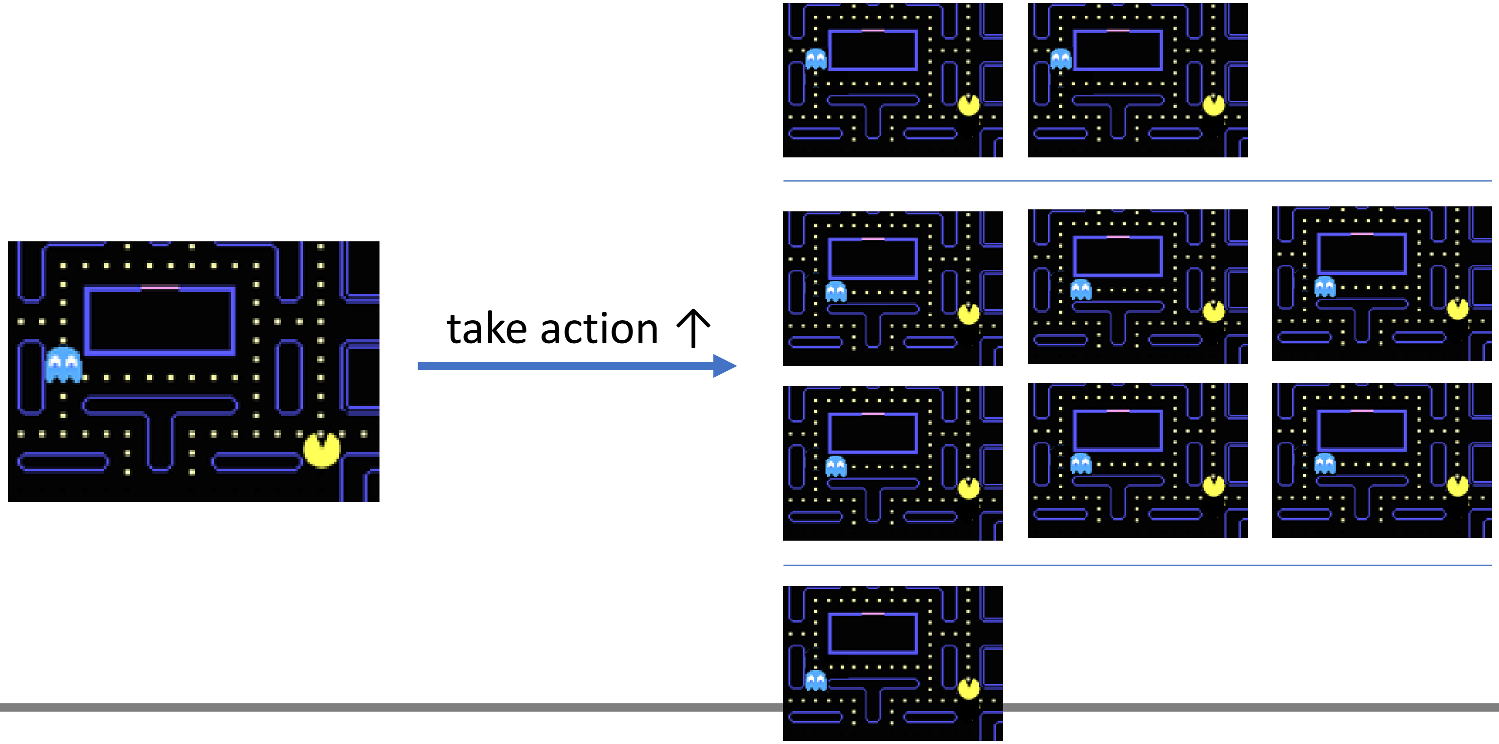
- Previous lecture: given full knowledge of an MDP, how to find its optimal policies
- What if the agent does not have such full knowledge?



Usually not available in RL!

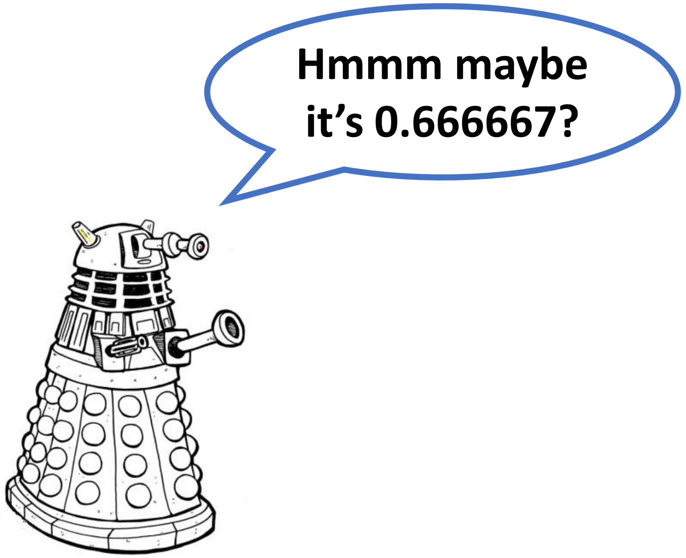
Model-based RL

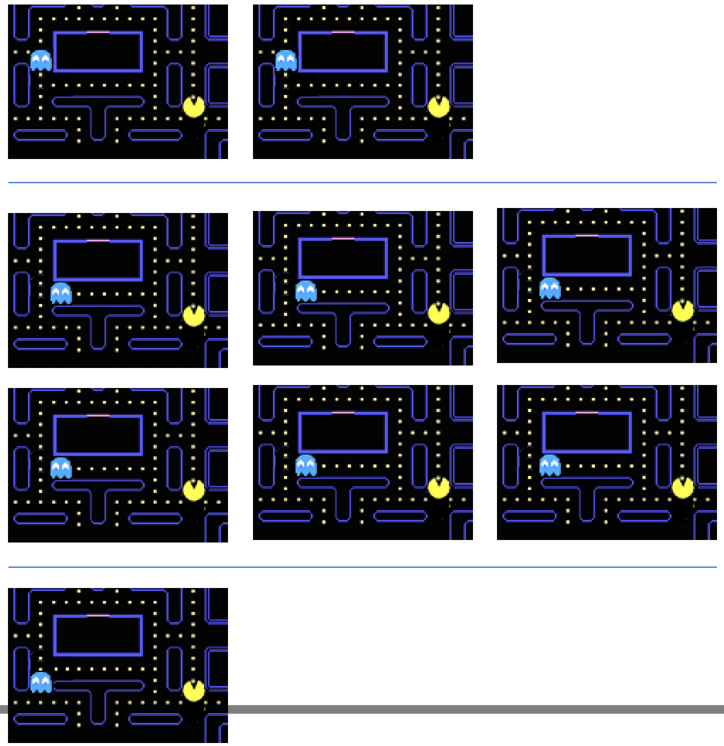
- Idea: if we do not know MDP M, why not estimate one from the information collected during interaction?



Model-based RL

- Idea: if we do not know MDP M, why not estimate one from the information collected during interaction?





Model-based RL

- Model-based reinforcement learning: maintain an estimated MDP $\widehat{M}$ ("model") and use it as the input of the planning algorithms
- $\widehat{M} \coloneqq \langle \hat{\mathcal{S}}, \hat{\mathcal{A}}, \hat{\mathcal{P}}, \hat{\mathcal{R}} \rangle$
  - $\hat{\mathcal{S}}$ and $\hat{\mathcal{A}}$: set of all visited states and actions
  - $\hat{\mathcal{P}}$: estimated transition probabilities
  - $\hat{\mathcal{R}}$: estimated immediate rewards

  - Can simply assume $\hat{\mathcal{S}} = \mathcal{S}$, $\hat{\mathcal{A}} = \mathcal{A}$ and let all unvisited states/actions have 0 estimated probability of transition

Model-based RL

- Estimating $\hat{\mathcal{P}}$
  - Let $N_{s,a}$ be the number of 'taking action a at state s' in the whole interaction history
  - Let $N_{s,a,s'}$ be the number of transition (s, a, s') occurred
  - $\hat{\mathcal{P}}(s, a, s') = \frac{N_{s,a,s'}}{N_{s,a}}$
  - By the law of large numbers, $\hat{\mathcal{P}}(s, a, s') \to \mathcal{P}(s, a, s')$ as $N_{s,a} \to \infty$

Model-based RL

- Estimating $\hat{\mathcal{R}}$
  - Let $R_{s,a,s'}$ be the sum of rewards received at transition (s, a, s') in the whole interaction history
  - Let $N_{s,a,s'}$ be the number of transition (s, a, s') occurred
  - $\hat{\mathcal{R}}(s, a, s') = \frac{R_{s,a,s'}}{N_{s,a,s'}}$
  - By the law of large numbers, $\hat{\mathcal{R}}(s, a, s') \to \mathcal{R}(s, a, s')$ as $N_{s,a,s'} \to \infty$

  - Trivial case: if reward is deterministic, then no need to estimate at all

Model-based RL

- (Vanilla) model-based RL algorithm
  - (0) Start with an arbitrary policy π and an estimated MDP $\widehat{M}$
  - (1) Interact with the environment using π, record transitions and rewards
  - (2) Update estimated MDP $\widehat{M}$
  - (3) Compute the optimal policy $\hat{\pi}^*$ of $\widehat{M}$ using PI/VI, update $\pi \leftarrow \hat{\pi}^*$
  - (4) Goto (1) until π converges

Model-based RL

- Real-world analogy to model-based RL: learning to play a game
  - (0) Start with an arbitrary strategy and an imaginary model of the game (including rules, stage designs, control, etc.)
  - (1) Play the game, see how the game works
  - (2) Update your imaginary model of the game
  - (3) Brainstorm some better strategy based on your imaginary model
  - (4) Goto (1) until you get bored

Model-based RL

- No strict restriction to #interaction between each update of π
  - Since PI/VI is computation-intensive, one may choose to call PI/VI every 10,000 steps of interaction
  - However, if interaction is more expensive, then updating π more frequently can be a good idea
    - Example of expensive interaction: controlling a robot by RL. During interaction it may crash and you have to spend much money & time to repair it

Model-based RL

- Convergence issue: will the vanilla model-based RL algorithm converge to optimal policies?

Model-based RL

- Convergence issue: will the vanilla model-based algorithm converge to optimal policies?

- **Not necessarily!**
- If the agent has never tried action a at state s, then it has no information about the consequence of taking a at s (i.e. $\hat{\mathcal{P}}(s, a, *) = 0$), thus $\hat{q}^*(s, a) = 0$
- If there exists a' such that $\hat{q}^*(s, a') > 0$, then the agent will **never** try a in the future!

Outline

- Model-based RL
- **Exploration vs Exploitation in RL**
- Model-free RL
    - Monte-Carlo methods
    - Temporal difference learning
- Issues with terminology

Exploration vs Exploitation

- Exploration: deliberately take actions that are not (seemingly) “optimal” according to the current knowledge
- Purpose of exploration: gain more information in the hope of discovering better policies

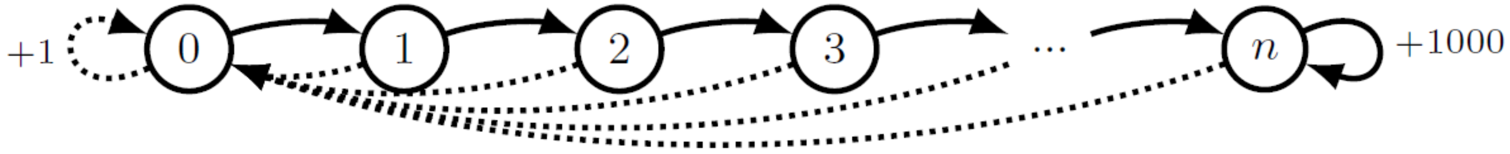
- Exploration vs exploitation dilemma

ε-greedy

- Simple exploration strategy: **$\boldsymbol{\varepsilon}$-greedy**
  - Choose a random action with probability ε
  - Choose the “optimal” action $\pi(s)$ with probability $1 - \varepsilon$.
  - ε is often set to small values like 0.03, 0.01, 0.003 or even smaller
- Pros of ε-greedy
  - Easy to implement
  - Stick to “optimal” actions most of the time → less likely to take disastrous actions (given that the initial policy is sufficiently good)

ε-greedy

- Cons of ε-greedy
  - Behaviour of the agent never actually converges (i.e. the agent never stops exploration)
  - Can be exponentially inefficient in some cases



Suppose the agent starts RL from s_0. It will soon discover that taking dotted actions will send it to s_0 and at s_0 it yields reward +1. Thus π will very likely be taking dotted actions at any states, unless it happens to discover s_n and reward +1000. This seldom happens because with ε-greedy exploration, the probability it reaches s_n by taking solid actions without going back to s_0 is only $(0.5\varepsilon)^n$.

R-MAX algorithm

- Systematic exploration: R-MAX algorithm
  - Assume $q(s, a) = R_{\max}$ (maximum immediate reward of that MDP), unless a has been taken at least m times at s
  - This forces the agent to try every possible actions many times before making any conclusion
- Pros of R-MAX
  - If m is set sufficiently large, then R-MAX algorithm has a high probability to be optimal in all but some polynomial steps in an infinitely long learning process (sample complexity theory)

R-MAX algorithm

- Cons of R-MAX
  - Will explore very aggressively (it tries *every* possible actions *many times*), which might not be appropriate in some real-world applications (robotics, automated driving, power station control, etc.)
  - If the state/action space is very large, the agent may spend too much time exploring without making any sensible plans (since most values are R_{max})

Other Exploration Strategies

- Other exploration strategies lie between ε-greedy (almost always greedy) and R-MAX (almost always explore)
  - e.g. UCB/UCRL, MoR-MAX, V-MAX, ICR/ICV, …
  - Basic idea: try to explore sufficiently in the long run, but also make use of collected information earlier than R-MAX (thus behave more “greedily”)

How to choose a strategy

- Choose exploration strategy based on the need of your application!
  - Real-world loss is negligible (e.g. game playing) → systematic exploration can be helpful
  - Experiments are expensive (robotics etc.) → more conservative (greedy) exploration

Outline

- Model-based RL
- Exploration vs exploitation in RL
- **Model-free RL**
  - Monte-Carlo methods
  - Temporal difference learning
- Issues with terminology

Model-free RL

- Maintaining a model of the environment can be expensive
  - Need to keep record of all interactions
  - Need to save $|\mathcal{S}|^2|\mathcal{A}|$ transition probabilities and rewards
  - If MDP is continuous, estimating transition probabilities can be difficult

- Can we update policies without keeping a model?
  - → Model-free RL!

Model-free RL: Monte-Carlo methods

- Recall that in the policy iteration (PI) algorithm:
  - (0) start from arbitrary π
  - (1) solve q^{π}
  - (2) improve π by $\pi(s) \leftarrow \mathrm{argmax}_{a \in \mathcal{A}}\, q^{\pi}(s, a)$
  - (3) goto (1) until π converges

- If v^{π} / q^{π} can be estimated by other means, then we do not need a model of MDP
- Since v^{π} / q^{π} are expected cumulative rewards, why not simply average the actual returns?

Model-free RL: Monte-Carlo methods

- Monte-Carlo (MC) value estimation
  - Initialization: $\forall s \in \mathcal{S}, N(s) = \hat{G}(s) = 0$
  - Whenever visit state $S_t = s$:
    - $N(s) \leftarrow N(s) + 1$
    - $\hat{G}(s) \leftarrow \hat{G}(s) + G_t$
    - $\hat{v}^\pi(s) \leftarrow \hat{G}(s)/N(s)$.
  - G_t is the actual cumulative reward $G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^k R_{t+k+1} + \cdots$ received during interaction starting from t until it ends or γ^k is sufficiently small

  - Can be done similarly for $\hat{q}^\pi$

Model-free RL: Monte-Carlo methods

- Convergence of Monte-Carlo value estimation algorithm
  - If $k \to \infty$, then $\hat{v}^{\pi}$ is an unbiased estimate of v^{π}
  - Thus, by the law of large numbers, $\hat{v}^{\pi}(s) \to v^{\pi}(s)$ as $N(s) \to \infty$

Model-free RL: Monte-Carlo methods

- Monte-Carlo reinforcement learning
  - (0) Start with arbitrary policy π
  - (1) Interact with the environment, record all cumulative rewards
  - (2) Update $\hat{v}^{\pi}$ or $\hat{q}^{\pi}$ with the MC value estimation algorithm
  - (3) Improve π by argmaxing $\hat{v}^{\pi}$ or $\hat{q}^{\pi}$
  - (4) Goto (1) until π converges

- As in model-based algorithms, exploration is necessary

Model-free RL: Monte-Carlo methods

- Pros of MC method
  - Does not need a model
  - Easy to implement
  - It even works if the decision process is not Markov
    - Does not care about transition probabilities at all
- Cons of MC method
  - Delayed update: need to wait until time $t + k$ before updating information on s_t
  - Does not rely on Markov property $\leftrightarrow$ Cannot utilize information more efficiently if the environment is Markov

Model-free RL: Monte-Carlo methods

- Incremental version of MC estimation:

$$\hat{v}^{\pi}(S_t) \leftarrow \hat{v}^{\pi}(S_t) + \frac{1}{N(S_t)}(G_t - \hat{v}^{\pi}(S_t))$$

  - Mathematically equivalent to $\hat{v}^{\pi}(S_t) \leftarrow \hat{G}(S_t)/N(S_t)$

- (Exponential) moving average version:

$$\hat{v}^{\pi}(S_t) \leftarrow \hat{v}^{\pi}(S_t) + \alpha(G_t - \hat{v}^{\pi}(S_t))$$

  - $0 < \alpha < 1$: update rate

Model-free RL: Temporal Difference

- $\hat{v}^{\pi}(S_t) \leftarrow \hat{v}^{\pi}(S_t) + \alpha(G_t - \hat{v}^{\pi}(S_t))$ has cumulative reward $G_t = R_{t+1} + \gamma R_{t+2} + \cdots$, thus need to wait many steps before updating S_t

- Any way to update instantly?

Model-free RL: Temporal Difference

- $G_t = R_{t+1} + \gamma R_{t+2} + \cdots = R_{t+1} + \gamma G_{t+1}$
- $v^\pi(S_t) \coloneqq \mathbb{E}[G_t|S_t] = \mathbb{E}[R_{t+1} + \gamma G_{t+1}|S_t]$
  $= \mathbb{E}[R_{t+1} + \gamma v^\pi(S_{t+1})|S_t].$
- Since we are interested in estimating $v^\pi(S_t)$, we can use $R_{t+1} + \gamma \hat{v}^\pi(S_{t+1})$ instead of G_t!

Model-free RL: Temporal Difference

- Temporal difference estimation (TD)

$$\hat{v}^{\pi}(S_t) \leftarrow \hat{v}^{\pi}(S_t) + \alpha\left(R_{t+1} + \gamma \hat{v}^{\pi}(S_{t+1}) - \hat{v}^{\pi}(S_t)\right)$$

- Compare to MC:

$$\hat{v}^{\pi}(S_t) \leftarrow \hat{v}^{\pi}(S_t) + \alpha(G_t - \hat{v}^{\pi}(S_t))$$

- $R_{t+1} + \gamma \hat{v}^{\pi}(S_{t+1})$ is sometimes called "TD target"
  $R_{t+1} + \gamma \hat{v}^{\pi}(S_{t+1}) - \hat{v}^{\pi}(S_t)$ is called "TD error"

Model-free RL: Temporal Difference

- (Vanilla) temporal difference RL
  - (0) Start with an arbitrary policy π
  - (1) Execute $A_t \leftarrow \pi(S_t)$, get R_{t+1} and S_{t+1}
  - (2) Update $\hat{v}^{\pi}(S_t)$ or $\hat{q}^{\pi}(S_t, A_t)$ with TD estimation
  - (3) Improve $\pi(S_{t+1})$ by argmaxing $\hat{v}^{\pi}(S_{t+1})$ or $\hat{q}^{\pi}(S_{t+1}, a)$
  - (4) Goto (1) until π converges

- Also needs exploration (e.g. using ε-greedy at (1))

Model-free RL: Temporal Difference

- Pros of TD (vs MC)
  - Instant update: whenever visit a state s, TD can update $\hat{v}(s)$ immediately
- Cons of TD
  - $R_{t+1} + \gamma v^{\pi}(S_{t+1})$ is unbiased estimate of G_t. However this is unavailable and $R_{t+1} + \gamma \hat{v}^{\pi}(S_{t+1})$ is used instead, which is a *biased* estimate of G_t.

MC vs TD

- MC: unbiased, but usually has a higher variance
- TD: biased, but usually has a lower variance

- TD also utilizes Markov property but MC does not
  - Using $\hat{v}^{\pi}(S_{t+1})$ to estimate $\hat{v}^{\pi}(S_t)$ in TD requires that transition between S_t and S_{t+1} is Markov
  - This is called "bootstrapping"
    - PI/VI also bootstraps
  - $\hat{v}^{\pi}(S_{t+1})$ is NOT used to estimate $\hat{v}^{\pi}(S_t)$ in MC

Other versions of TD algorithms

- Sarsa algorithm

$$\hat{q}^{\pi}(S_t, A_t) \leftarrow \hat{q}^{\pi}(S_t, A_t) + \alpha\big(R_{t+1} + \gamma \hat{q}^{\pi}(S_{t+1}, A_{t+1}) - \hat{q}^{\pi}(S_t, A_t)\big)$$

  - q version of TD
  - Use $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}$, hence the name

Other versions of TD algorithms

- Q-learning algorithm

$$\hat{q}^{\pi}(S_t, A_t) \leftarrow \hat{q}^{\pi}(S_t, A_t) + \alpha\left(R_{t+1} + \gamma \max_a \hat{q}^{\pi}(S_{t+1}, a) - \hat{q}^{\pi}(S_t, A_t)\right)$$

  - Use $\max_a \hat{q}^{\pi}(S_{t+1}, a)$ instead of $\hat{q}^{\pi}(S_{t+1}, A_{t+1})$
  - Integrate value evaluation and policy improvement into one step
- Analogy:
  - Policy Iteration (on v) → TD
  - Policy Iteration (on q) → Sarsa
  - Value Iteration (on q) → Q-learning

On-policy vs off-policy

- On-policy: estimated values $\hat{q}^{\pi}$ is about π
  - TD, Sarsa, MC
- Off-policy: estimated values $\hat{q}^{\pi}$ is NOT about π, but about some other (possibly better) policy
  - Q-learning
- In finite MDPs, their performance is similar
- In infinite MDPs, when function approximation is applied, off-policy algorithms can suffer from the stability issue (never converges)

Outline

- Model-based RL
- Exploration vs exploitation in RL
- Model-free RL
  - Monte-Carlo methods
  - Temporal difference learning
- **Issues with terminology**

Issues with terminology

- The use of “model-based” & “model-free” is sometimes confusing
  - Some researchers use “model” to refer to estimated MDPs, as in previous pages
    - “model-based” → plan with estimated MDPs
    - “model-free” → plan without estimated MDPs
    - both need interaction for collecting information
  - Some use “model” to refer to full prior (ground truth) knowledge of MDPs
    - “model-based” → plan with full knowledge of MDPs, does not need interaction
    - “model-free” → plan without full knowledge, need interaction

Issues with terminology

- In the second sense, “model-based RL” in previous pages is actually “model-free”
  - Thus making three categories:
    - “model-based RL”
    - “model-free model-based RL”
    - “model-free model-free RL” or simply “model-free RL”

- Since “model-based RL” in this sense is just solving MDPs without the need of interaction at all, some argue that it does not count as RL

Issues with terminology

- Suggestion: stick to the 1st interpretation, but call the case with full knowledge of MDP as "model-given RL"

	1st sense	2nd sense	Suggestion
Full knowledge available	Not really RL	Model-based RL	Model-given RL
Planning with estimated MDPs	Model-based RL	Model-free model-based RL	Model-based RL
Planning without estimated MDPs	Model-free RL	Model-free RL	Model-free RL

Issue with terminology (2)

- “Online RL” vs “offline RL”
  - Common sense:
    - “Online RL” = loop {interactions; update policy;}
    - “Offline RL” = interactions; learn a policy; utilize the policy in the application
      - After obtaining a policy, the learning process stops
  - Some researchers use “online RL” to refer to “model-free RL” and “offline RL” to “model-based RL” (1st sense)
    - This is now often treated as a mistake since model-based algorithms can be used online and model-free algorithms can be used offline