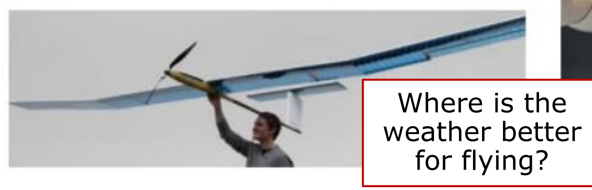
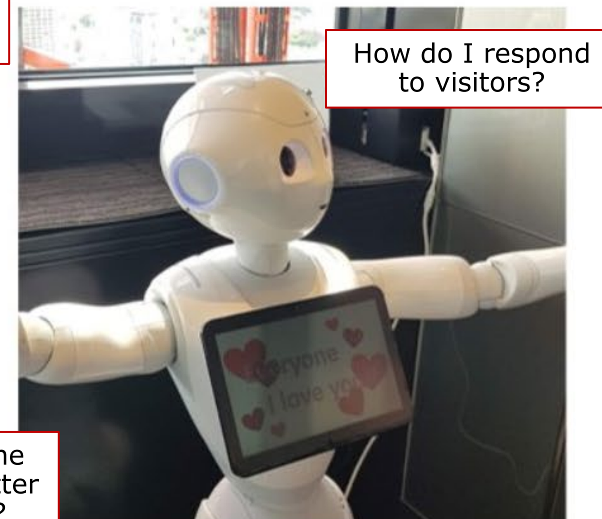
6 – Reinforcement Learning

Dr. Marija Popović

UNIVERSITÄT BONN

Motivation

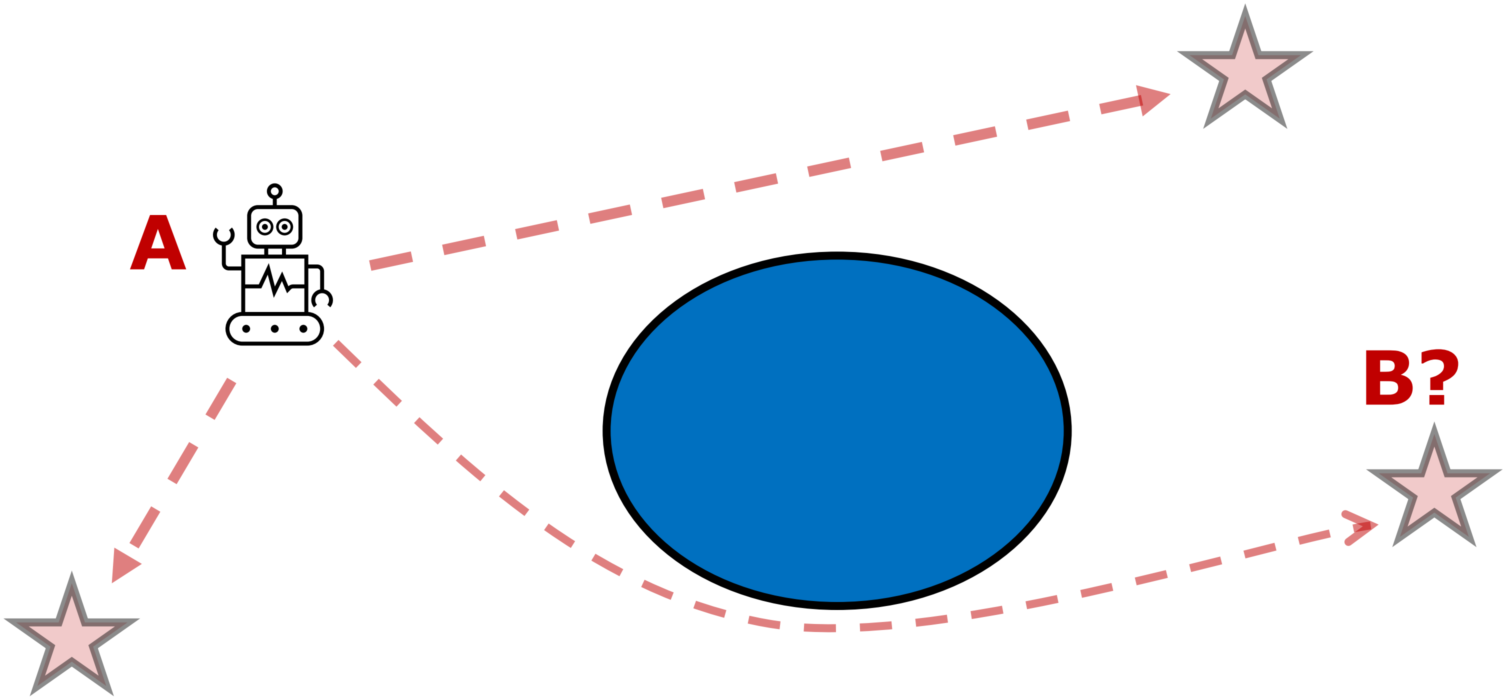




http://www.cs.columbia.edu/~allen/F17/NOTES/Lecture_2.pdf
http://asl.stanford.edu/aa274a/pdfs/lecture/lecture_1.pdf
Hitz et al. (2014), "Fully autonomous focused exploration for robotic environmental monitoring,“ in: IEEE ICRA.
Lehnert et al. (2020), “Performance improvements of a sweet pepper harvesting robot in protected cropping environments,“ in: JFR. 37(7): pp.1197-1223.

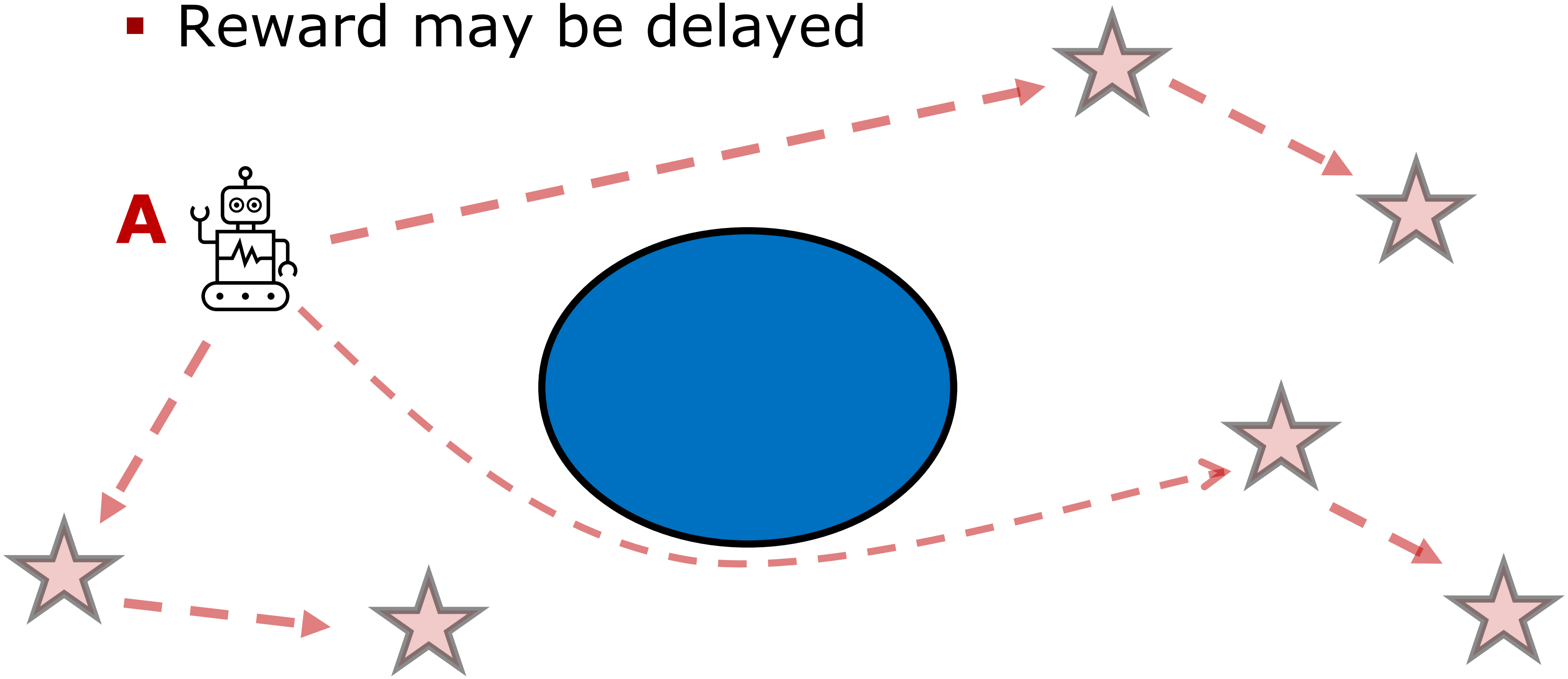
What is Decision-Making?

- Find goal location(s) (point B) to fulfill a particular task – *where*?



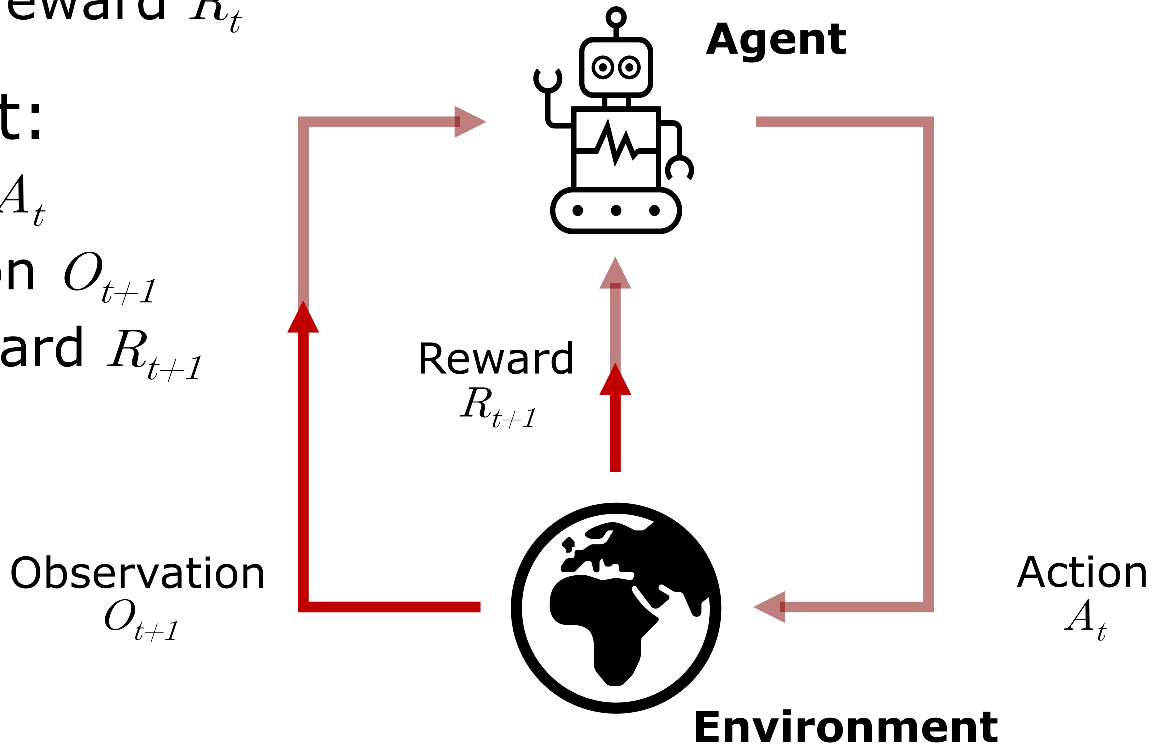
Sequential Decision-Making

- **Goal**: Choose actions to maximise the total expected future reward
- **Challenges**:
  - Subsequent actions depend on what is observed
  - Reward may be delayed



Agent and Environment

- Agent and environment interact continually
- At each time step t, the agent:
  - Performs action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}



MDP – Dynamic Programming

- **Markov Decision Process (MDP)**: General sequential decision-making framework
  - Markov dynamics
  - Fully observable environment
  - Stochastic state transitions
  - Reward functions

- **Dynamic programming**: Solution method for known MDPs

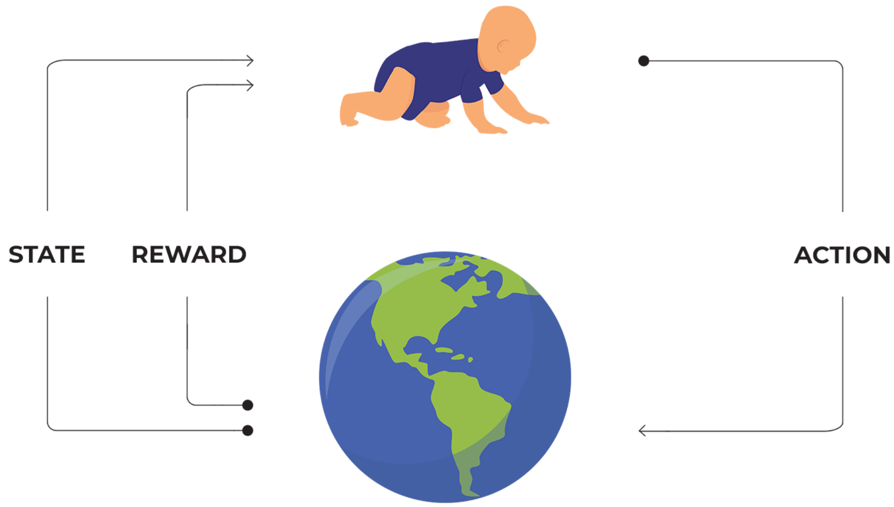
MDP – Reinforcement Learning

- **Markov Decision Process (MDP)**: General sequential decision-making framework
  - Markov dynamics
  - Fully observable environment
  - Stochastic state transitions
  - Reward functions

- **Reinforcement learning (RL)**: Solution method for unknown MDPs

Reinforcement Learning

- Learning from interaction
- Feedback restricted to a reward signal
- Can handle arbitrarily large MDPs
- Resembles human-like learning

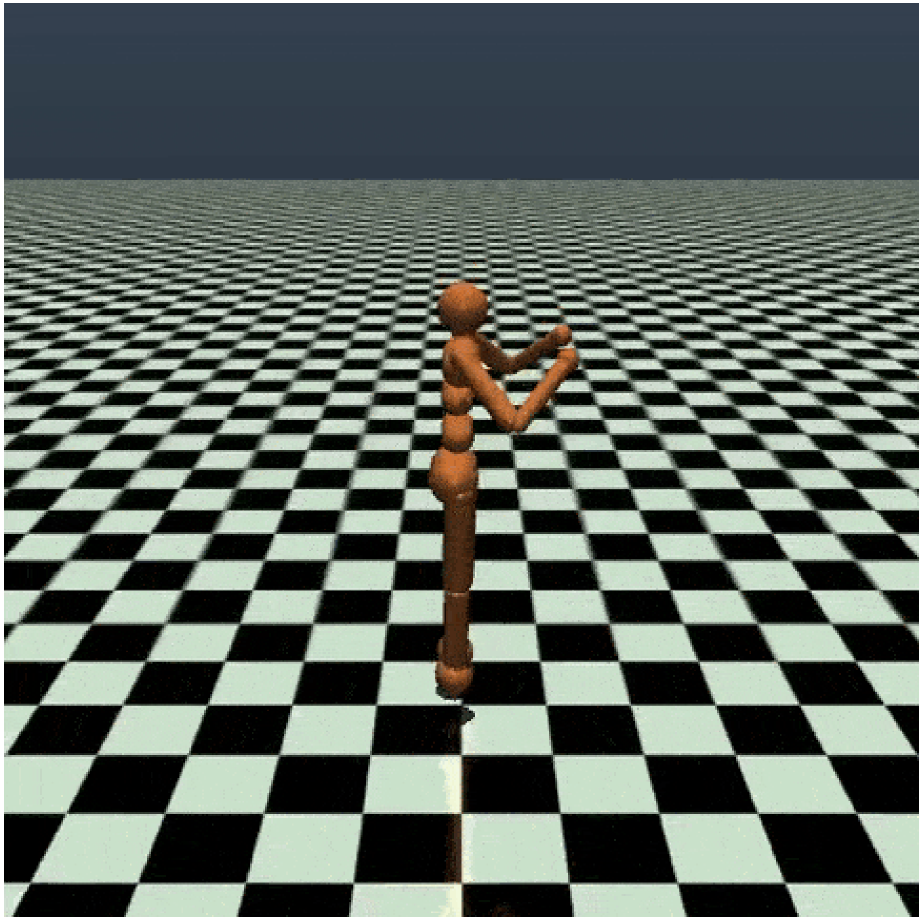


- Aim: Find optimal actions given limited feedback

https://towardsdatascience.com/reinforcement-learning-brain-and-psychology-part-1-introduction-b5f79a0475ab

Reinforcement Learning

- Example: Learning how to run



https://becomesentient.com/reinforcement-learning-intro/

Reinforcement Learning

- Example: Self-driving cars

https://news.voyage.auto/introducing-voyage-deepdrive-69b3cf0f0be6

Model-free RL

- **Model-free prediction**
  - **Estimate** values in an unknown MDP

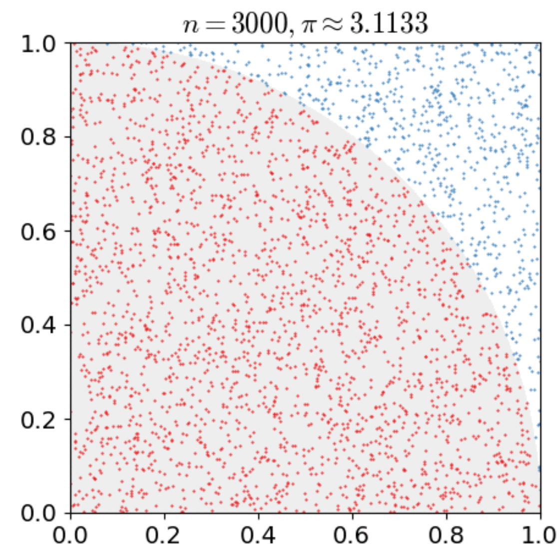
- **Model-free control**
  - **Optimise** values in an unknown MDP

Model-free Prediction I

Monte Carlo (MC) Methods

- Approximate quantity of interest by randomness and the law of large numbers

- <u>Example</u>: How to estimate π?
  - Draw (x, y) vector at random in $[0, 1]^2$
  - Count vectors with distance $<= 1$ from origin
  - Estimate is $\frac{\pi}{4}$



Monte Carlo RL

- Learn from **episodes**
  - Episode: Full state-action-reward-state sequence until terminal state reached
  - Return: Cumulative discounted reward of episode $G_t = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-1} R_T$

- **Model-free** prediction
  - Unknown MDP
  - True value function: $v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$
  - Sample expectation by mean return over episodes

First-Visit MC

- <u>Aim</u>: Evaluate value of state s

- If s is visited in an episode for the **first** time t:
  - Increment counter $\mathrm{N}(s) = \mathrm{N}(s) + 1$
  - Total return $\mathrm{S}(s) = \mathrm{S}(s) + G_t$
  - $\mathrm{V}(s) = \mathrm{S}(s) \ / \ \mathrm{N}(s)$
  - Run until convergence

Every-Visit MC

- Aim: Evaluate value of state s

- **Every** time s is visited in an episode:
  - Increment counter $\mathrm{N}(s) = \mathrm{N}(s) + 1$
  - Total return $\mathrm{S}(s) = \mathrm{S}(s) + G_t$
  - $\mathrm{V}(s) = \mathrm{S}(s) \ / \ \mathrm{N}(s)$
  - Run until convergence

Incremental Value Update

- Update each $\mathrm{V}(s)$ after episode of length T
- For t in $[T]$:
  - $\mathrm{N}(S_t) = \mathrm{N}(S_t) + 1$
  - $\mathrm{V}(S_t) = \mathrm{V}(S_t) + (1/\mathrm{N}(S_t)) * (G_t - \mathrm{V}(S_t))$

- **Running average**
  - $\mathrm{V}(S_t) = \mathrm{V}(S_t) + \alpha * (G_t - \mathrm{V}(S_t))$

MC Value Iteration?

- Is there a way to use MC methods for classical value iteration dynamic programming algorithms?

- Recall **optimal value function** definition:

$$v_*(s) = \boxed{\max_{a \in \mathcal{A}}} \mathcal{R}_s^a + \mathbb{E}_{p(s'|s,a)} \gamma v_*(s')$$

- One cannot sample the **maximisation** operator!

MC Policy Evaluation – Discussion

- Advantages:
  - Simple
  - Relatively insensitive to initial value guess

- Disadvantages:
  - Requires episodic environments
  - Inefficient if:
    - Long planning horizon needed
    - Only sparse rewards provided as feedback

Temporal-Difference (TD) Learning

- Model-free learning from episodes
- Learn from **incomplete** episodes after each time step by **bootstrapping** the true value function

- **TD(0) Algorithm**:
  - For each time step t:

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

  - TD target: $R_{t+1} + \gamma V(S_{t+1})$
  - TD error: $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$

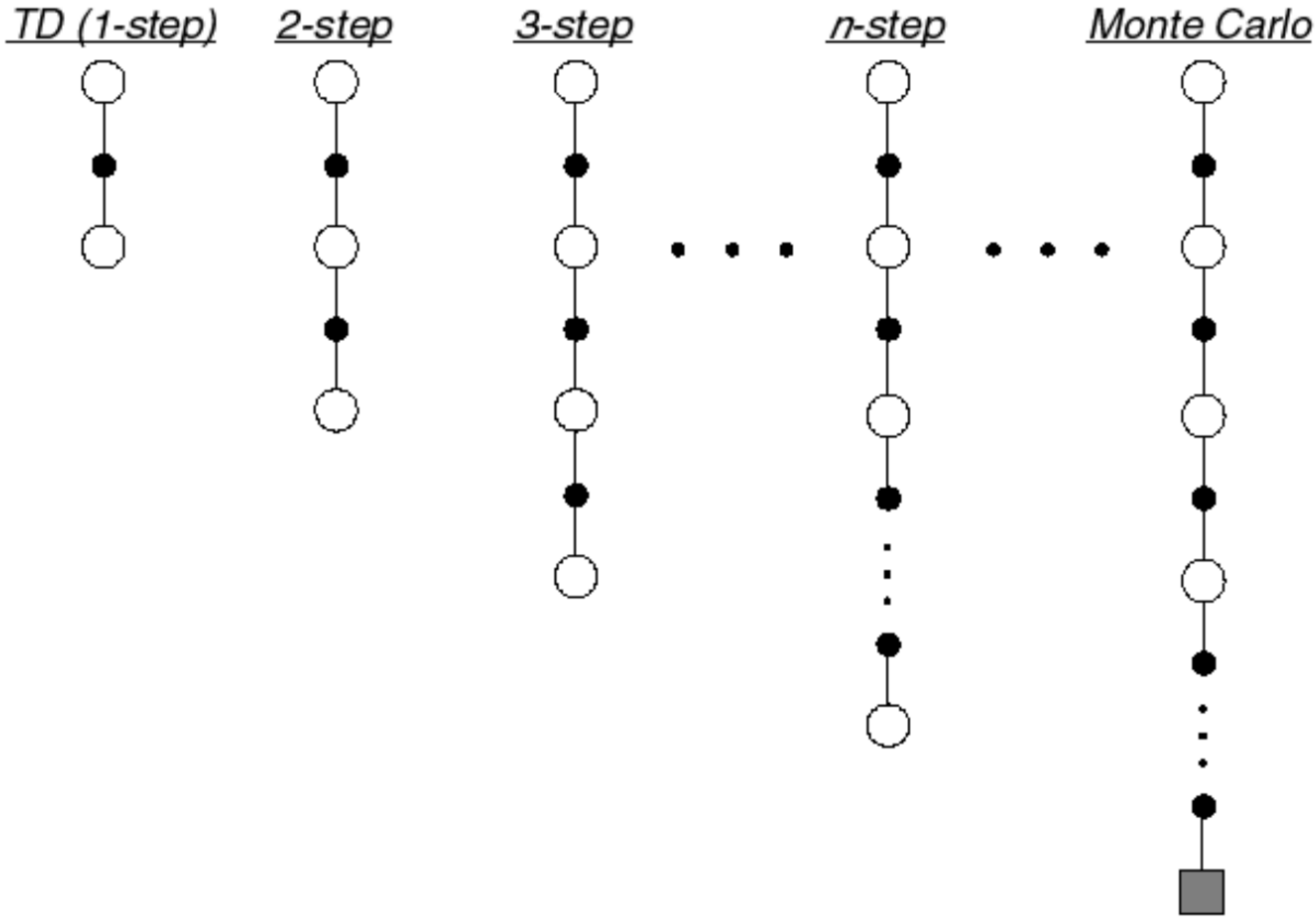
Model-free Prediction II

TD vs. MC

- **TD** learns in each step
  - Lower variance, faster convergence
  - Does not depend on episodic environments
  - Sensitive to poor initial value guess
  - Exploits Markov property

- **MC** learns after episode termination
  - High variance and low bias
  - Insensitive to poor initial value guess
  - Stable in non-Markovian environments

n-Step Prediction



https://www.davidsilver.uk/wp-content/uploads/2020/03/MC-TD.pdf

n-Step Prediction

$$
\begin{array}{lll}
n = 1 & (TD) & G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1}) \\
n = 2 & & G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2}) \\
\vdots & & \vdots \\
n = \infty & (MC) & G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-1} R_T
\end{array}
$$

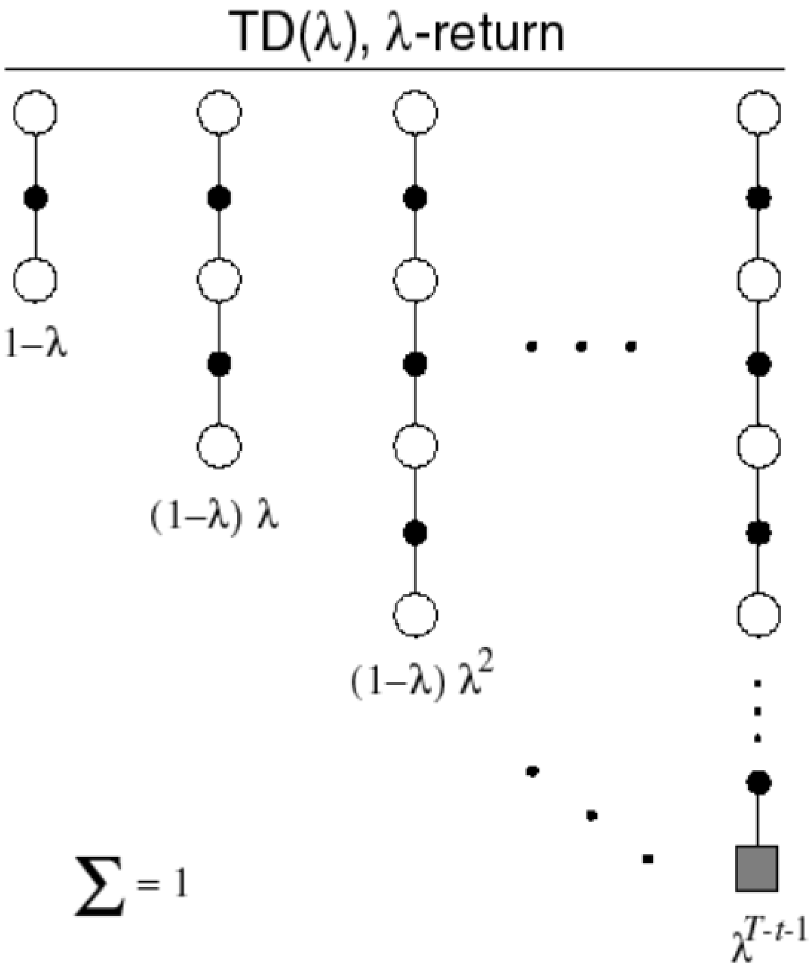
- n-step return

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

- n-step TD learning

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{(n)} - V(S_t) \right)$$

λ-return



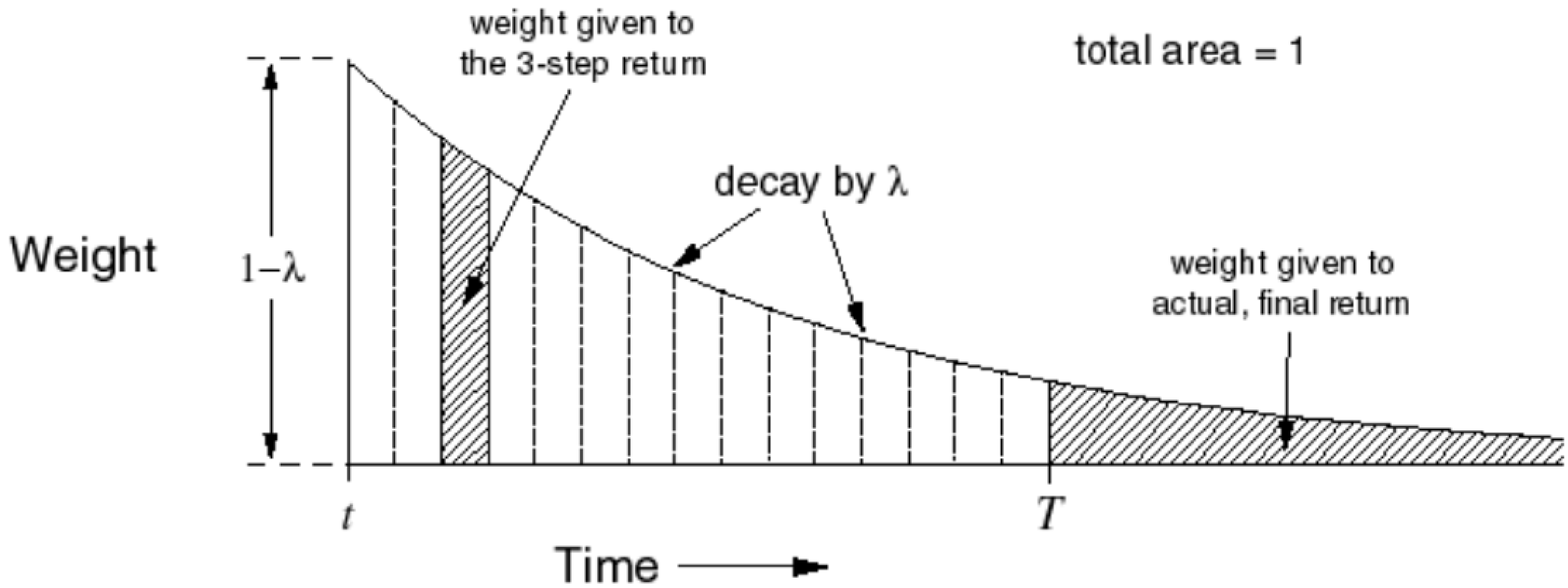
- Combination of all n-step returns
- Generalises n-step return TD-learning

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_t^{(n)}$$

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t^{\lambda} - V(S_t))$$

https://www.davidsilver.uk/wp-content/uploads/2020/03/MC-TD.pdf

TD(λ) Learning



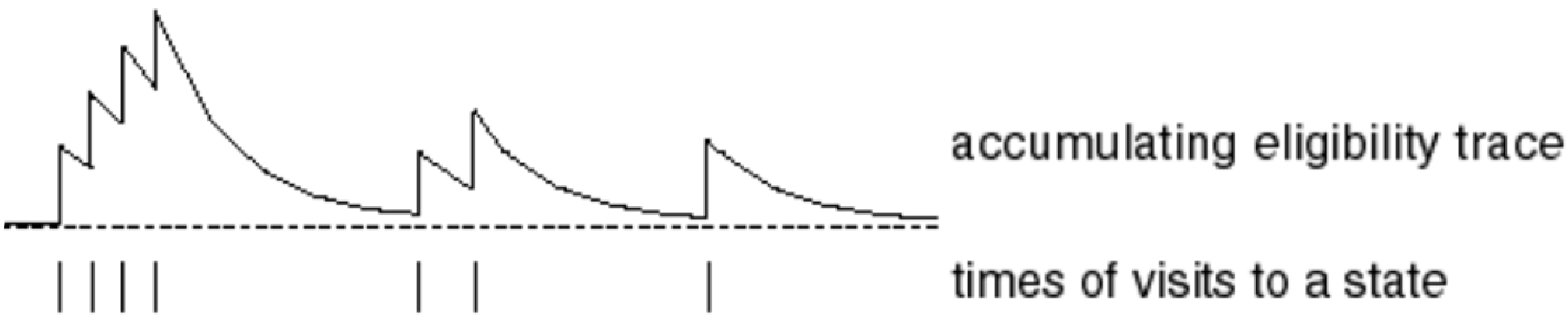
- λ is a decay parameter to trade-off short-term and long-term returns

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t^{\lambda} - V(S_t))$$

Eligibility Traces

- TD(λ) assigns larger weights to more recently seen states
- What about states that we see more frequently?

$$E_0(s) = 0$$
$$E_t(s) = \gamma \lambda E_{t-1}(s) + \mathbf{1}(S_t = s)$$



TD(λ) with Eligibility Traces

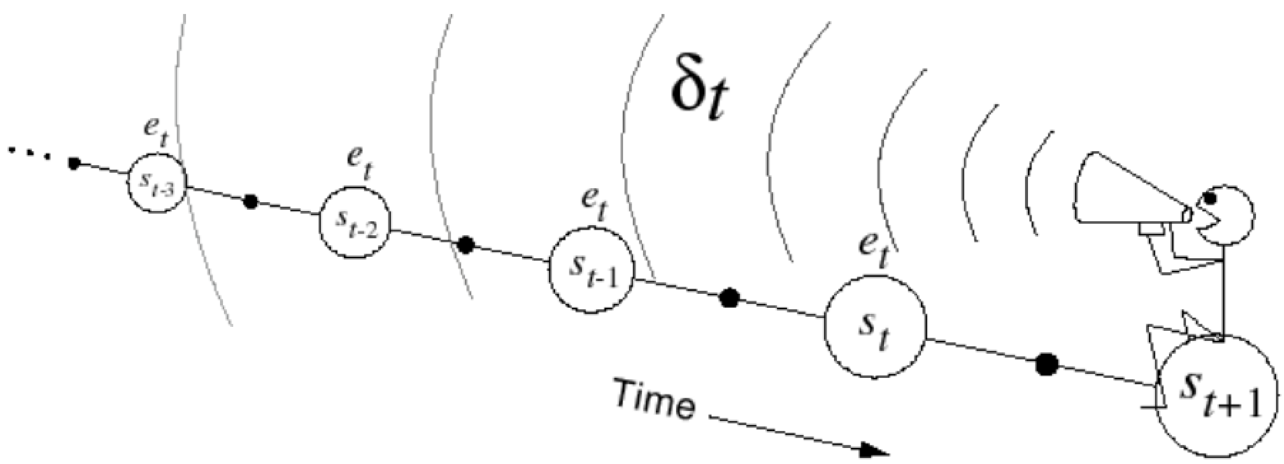
- TD error:

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

- TD(λ) update rule:

$$V(s) \leftarrow V(s) + \alpha \delta_t \underline{E_t(s)}$$

How recent?
How frequent?



TD(λ) with Eligibility Traces

Initialize $V(s)$ arbitrarily and $e(s) = 0$, for all $s \in S$

Repeat (for each episode) :

Initialize s

Repeat (for each step of episode) :

$a \leftarrow$ action given by π for s

Take action a, observe reward, r, and next state s'

$\delta \leftarrow r + \gamma V(s') - V(s)$

$e(s) \leftarrow e(s) + 1$

For all s :

$V(s) \leftarrow V(s) + \alpha \delta e(s)$

$e(s) \leftarrow \gamma \lambda e(s)$

$s \leftarrow s'$

Until s is terminal

https://towardsdatascience.com/eligibility-traces-in-reinforcement-learning-a6b458c019d6

Model-free Control I

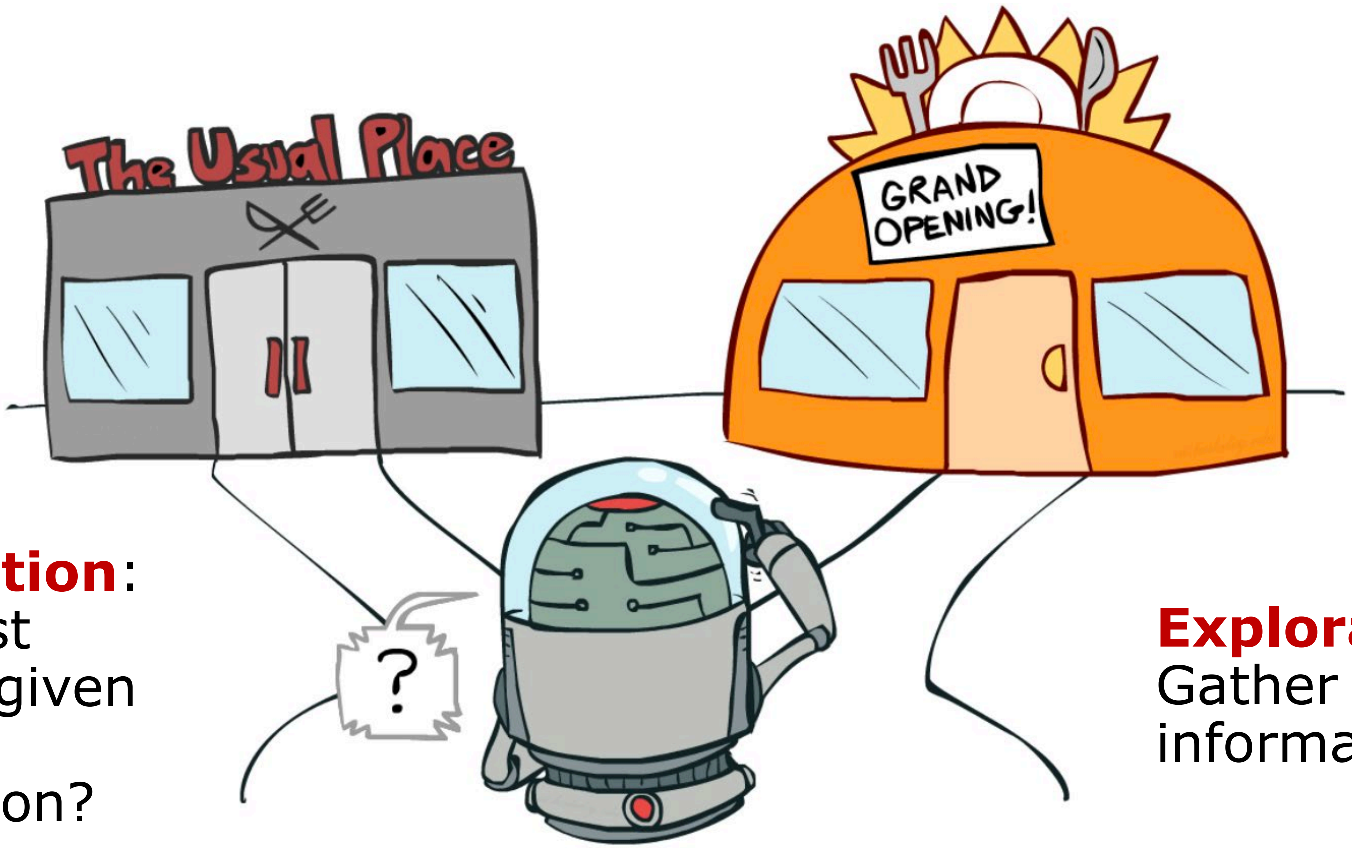
Recap: Policy Iteration

- The agent only cares about finding the optimal policy (not all the state-values)
- **Policy iteration** alternates the following steps, starting with an initial policy π_0 :
  - **Policy evaluation**: given policy π_k , calculate $v_k(s) = v_{\pi_k}$, $s \in \mathcal{S}$
  - **Policy improvement**: calculate maximum expected utility policy π_{k+1} :

$$\pi_{k+1} = \arg\max_{a \in \mathcal{A}} q_\pi(s, a)$$

Exploration vs. Exploitation

- Learning from experience



Exploitation:
Make best decision given current information?

Exploration:
Gather new information

https://medium.com/analytics-vidhya/the-epsilon-greedy-algorithm-for-reinforcement-learning-5fe6f96dc870

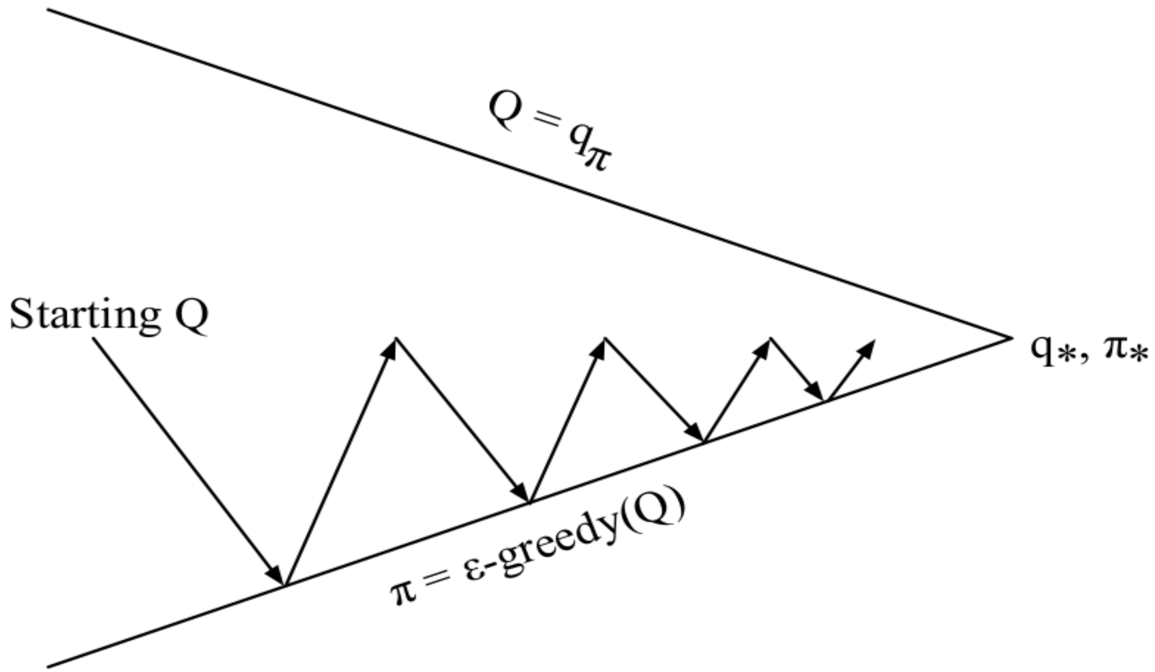
ε-greedy Policies

- Assume we have estimates $Q(s,a)$

$$\pi(a|s) = \begin{cases} \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q(s,a) & \text{with probability } 1\text{-}\varepsilon \\ \text{random action} & \text{with probability } \varepsilon \end{cases}$$

- ε is the **exploration rate**
- ε too high → slow convergence
- ε too low → suboptimal convergence

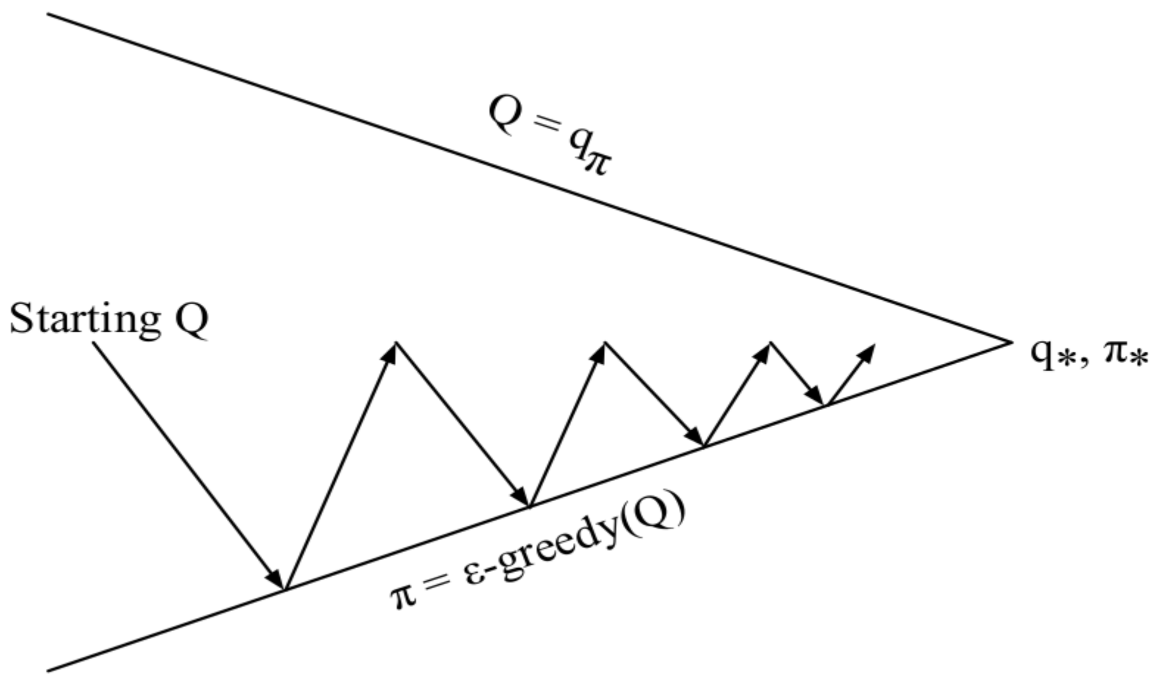
MC Policy Evaluation



After *every* episode:

- **Policy evaluation**: MC
- **Policy improvement**: ε-greedy

TD Policy Evaluation

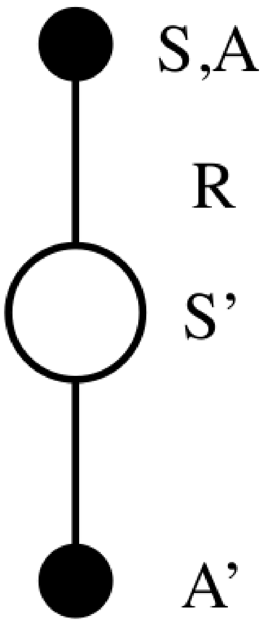


After *every* **time step**:

- **Policy evaluation**: **TD**
- **Policy improvement**: ε-greedy

- Lower variance
- Online
- Incomplete sequences

SARSA



- Bootstrap state-action values
- One-step policy evaluation by updating action-values (Q-values)
- ε-greedy policy improvement

Update rule for Q-values:

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

SARSA

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Repeat (for each step of episode):
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ε-greedy)
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until S is terminal

SARSA(λ)

- Weight more frequent and more recent **state-action pairs**
- Eligibility traces:

$$E_0(s,a) = 0$$

$$E_t(s,a) = \gamma\lambda E_{t-1}(s,a) + \mathbf{1}(S_t = s, A_t = a)$$

- Update rule:

$$\delta_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha \delta_t E_t(s,a)$$

SARSA(λ)

Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
Repeat (for each episode):
  $E(s,a) = 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
  Initialize S, A
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $\delta \leftarrow R + \gamma Q(S', A') - Q(S, A)$
    $E(S,A) \leftarrow E(S,A) + 1$
    For all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:
      $Q(s,a) \leftarrow Q(s,a) + \alpha \delta E(s,a)$
      $E(s,a) \leftarrow \gamma \lambda E(s,a)$
    $S \leftarrow S'$; $A \leftarrow A'$
  until S is terminal

Model-free Control II

On- vs. Off-Policy Learning

- **On-policy learning**
  - Evaluate π by sampling experience following π
  - e.g. SARSA

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

- **Off-policy learning**
  - Evaluate π by sampling experience following π'
  - π is the **target policy**, π' is the **behaviour policy**

Q-Learning

- Q-learning learns Q-values off-policy

- Choose next action following behaviour policy: $A_{t+1} \sim \pi'(\cdot|S_t)$

- TD target's next action A' from target policy: $A' \sim \pi(\cdot|S_t)$

- **Update rule**:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t))$$

Q-Learning

- Target policy π is greedy wrt. $Q(s,a)$

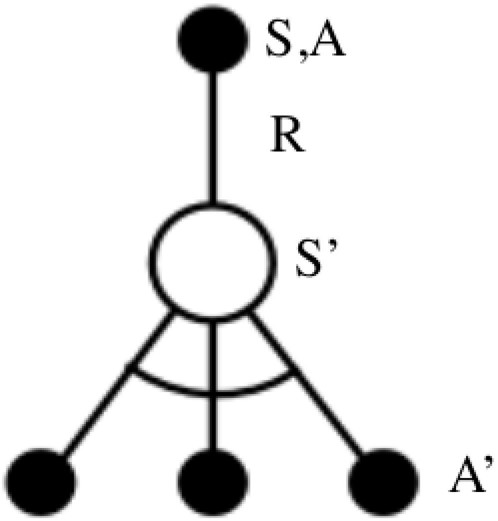
$$\pi(S_{t+1}) = \underset{a'}{\operatorname{argmax}}\, Q(S_{t+1}, a')$$

- Behaviour policy π' is ε-greedy wrt. $Q(s,a)$

- **Q-learning target**:

$$
\begin{aligned}
& R_{t+1} + \gamma Q(S_{t+1}, A') \\
&= R_{t+1} + \gamma Q(S_{t+1}, \underset{a'}{\operatorname{argmax}}\, Q(S_{t+1}, a')) \\
&= R_{t+1} + \max_{a'} \gamma\, Q(S_{t+1}, a')
\end{aligned}
$$

Q-Learning

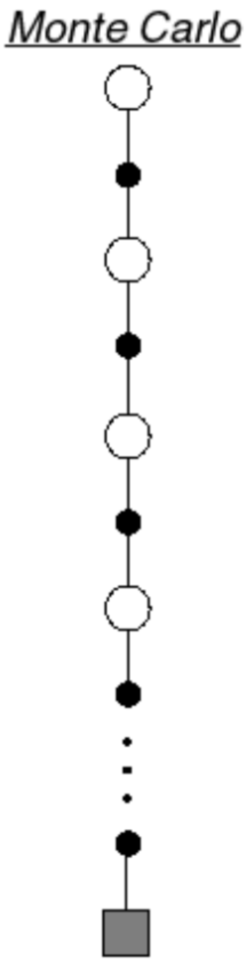


$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \boxed{\max_{a'} \gamma\, Q(S', a')} - Q(S, A))$$

Recap

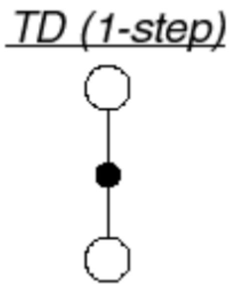
Recap

- **Model-free prediction**



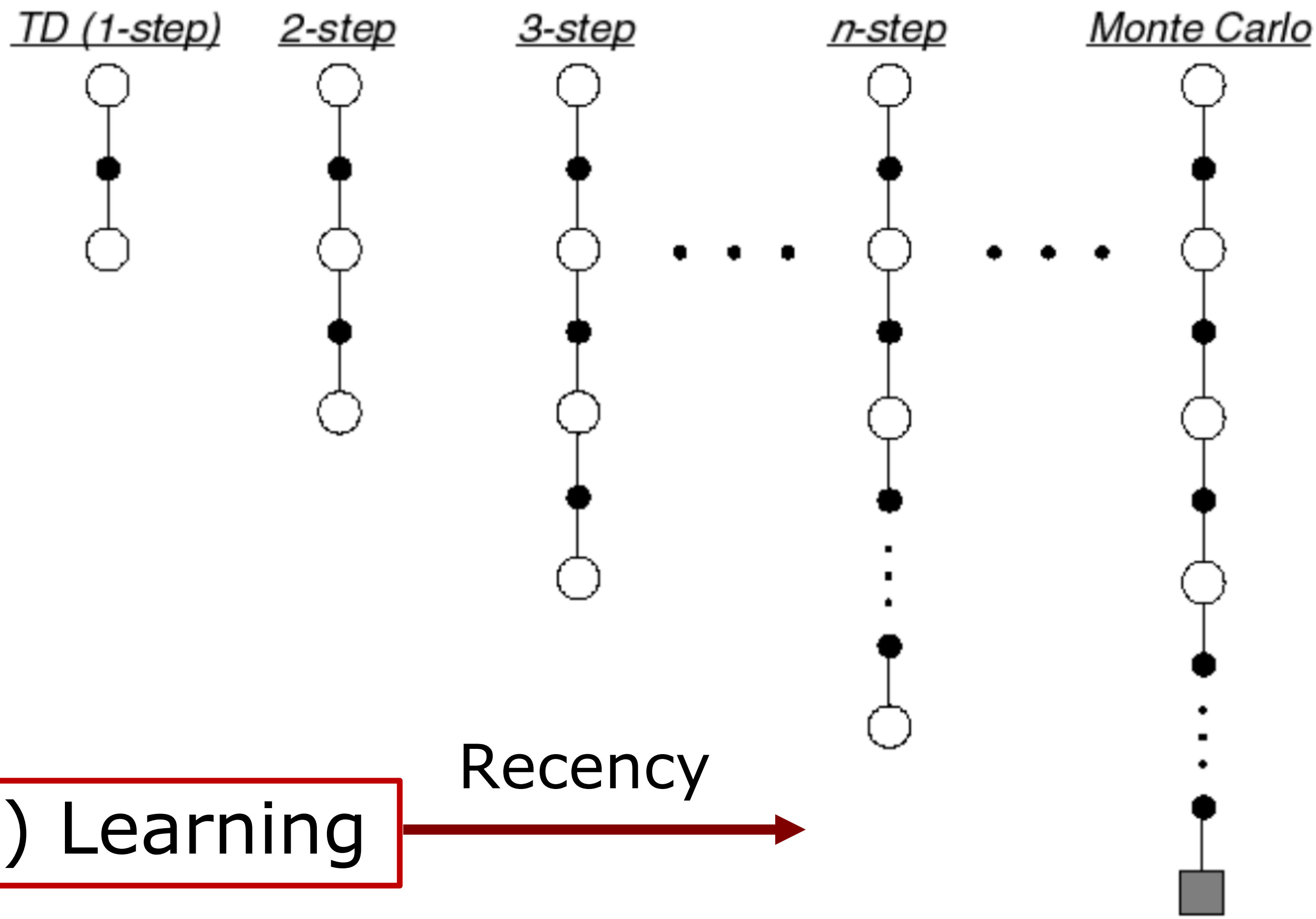
Recap

- **Model-free prediction**



Recap

- **Model-free prediction**

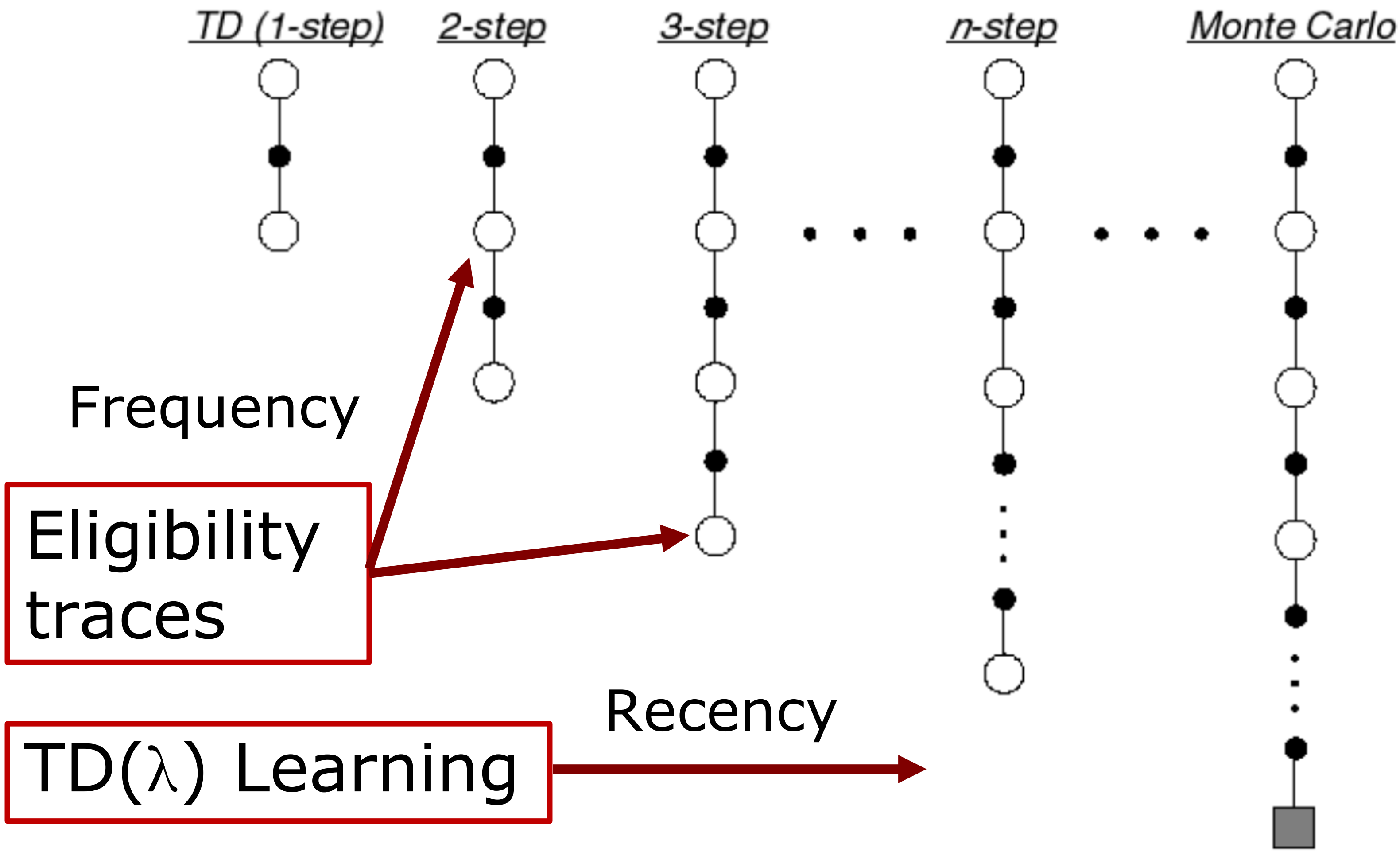


TD(λ) Learning

Recency

Recap

- **Model-free prediction**



Recap

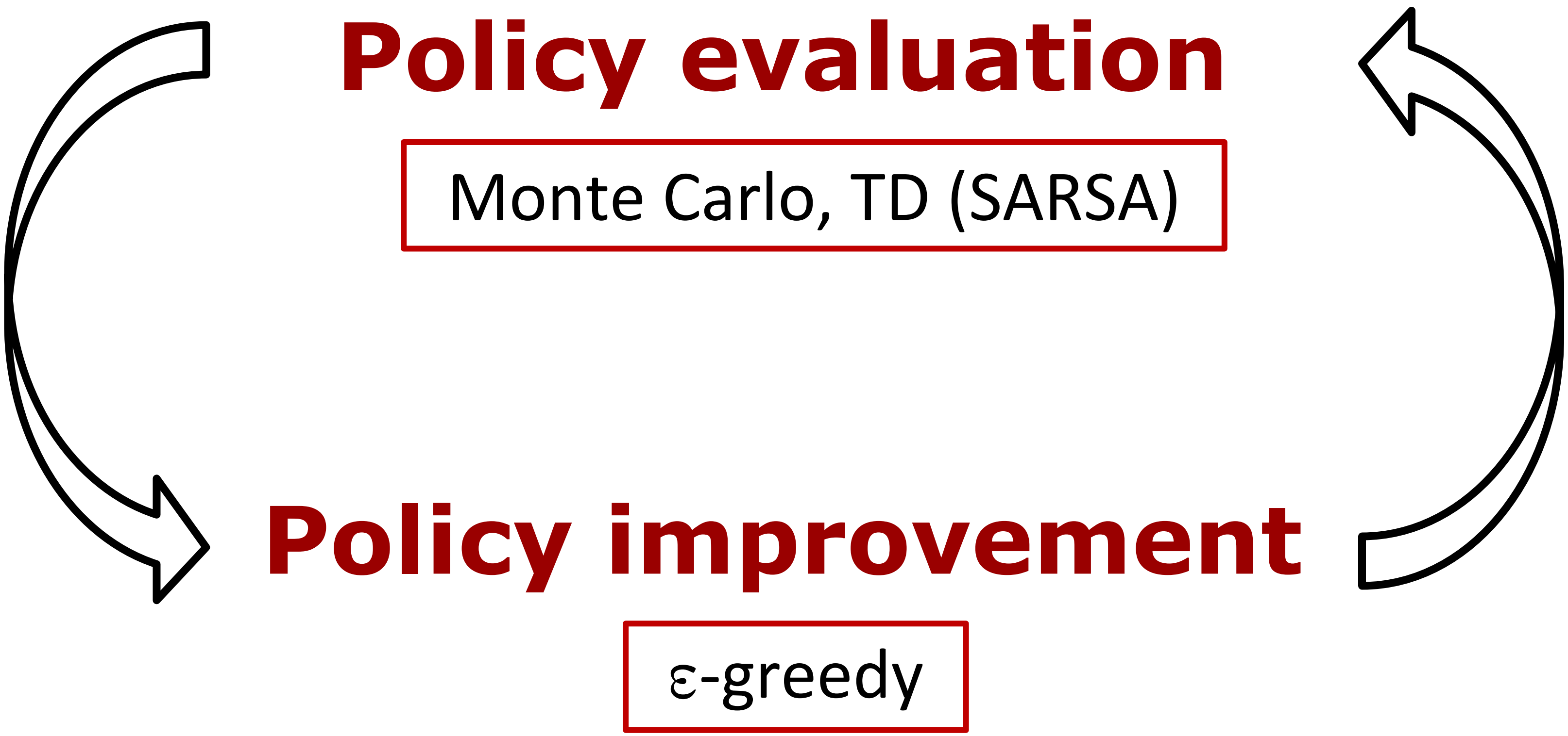
- **Model-free control**

Policy improvement

ε-greedy

Recap

- **Model-free control**



Recap

- **Model-free control**

Policy evaluation

Monte Carlo, TD (SARSA)
SARSA(λ) + eligibility

Policy improvement

ε-greedy

Recap

- **Model-free control**

Policies evaluation

Off-policy: Q-learning

Policy improvement

Recap

- **Model-free control**

Policies evaluation

Off-policy: Q-learning

Policy improvement

Summary

- Model-free prediction
  - Monte Carlo methods
  - Temporal Difference learning
  - n-step and λ-returns
  - Eligibility traces
- Model-free control
  - On-policy: SARSA algorithm
  - Off-policy: Q-learning

Further Reading

- Introduction to Reinforcement Learning with David Silver | DeepMind
- Reinforcement Learning: A Gentle Introduction - Become Sentient
- The Ingredients of Real World Robotic Reinforcement Learning – The Berkeley Artificial Intelligence Research Blog
- Reinforcement Learning, Part 1: A Brief Introduction | by dan lee | AI³ | Theory, Practice, Business | Medium
- https://gym.openai.com/
- A (Long) Peek into Reinforcement Learning (lilianweng.github.io)
- Reinforcement Learning: An Introduction – Barto and Sutton (1992)

Thank you for your attention