Reinforcement Learning in Finance

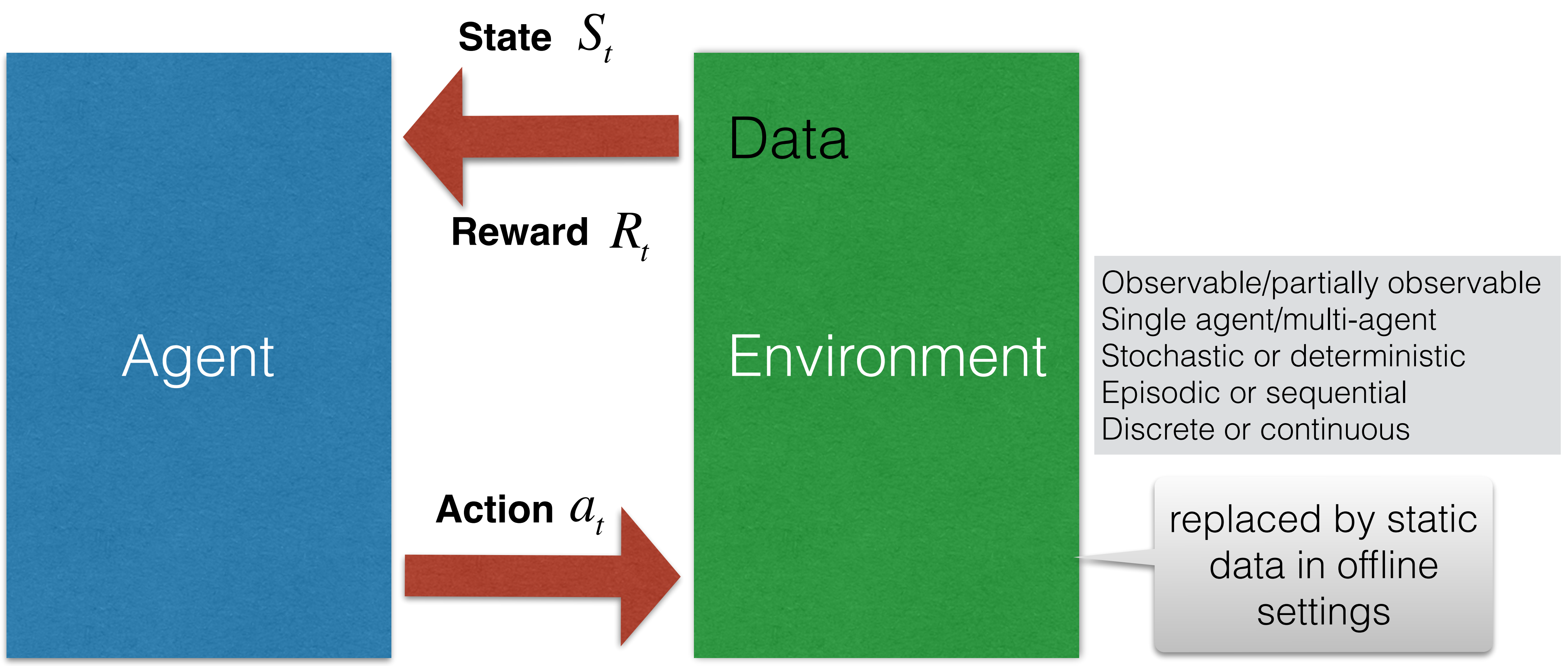
Week 1: Reinforcement Learning

Lesson 1: MDP-and-RL-intro

Igor Halperin

NYU Tandon School of Engineering, 2017

Reinforcement Learning



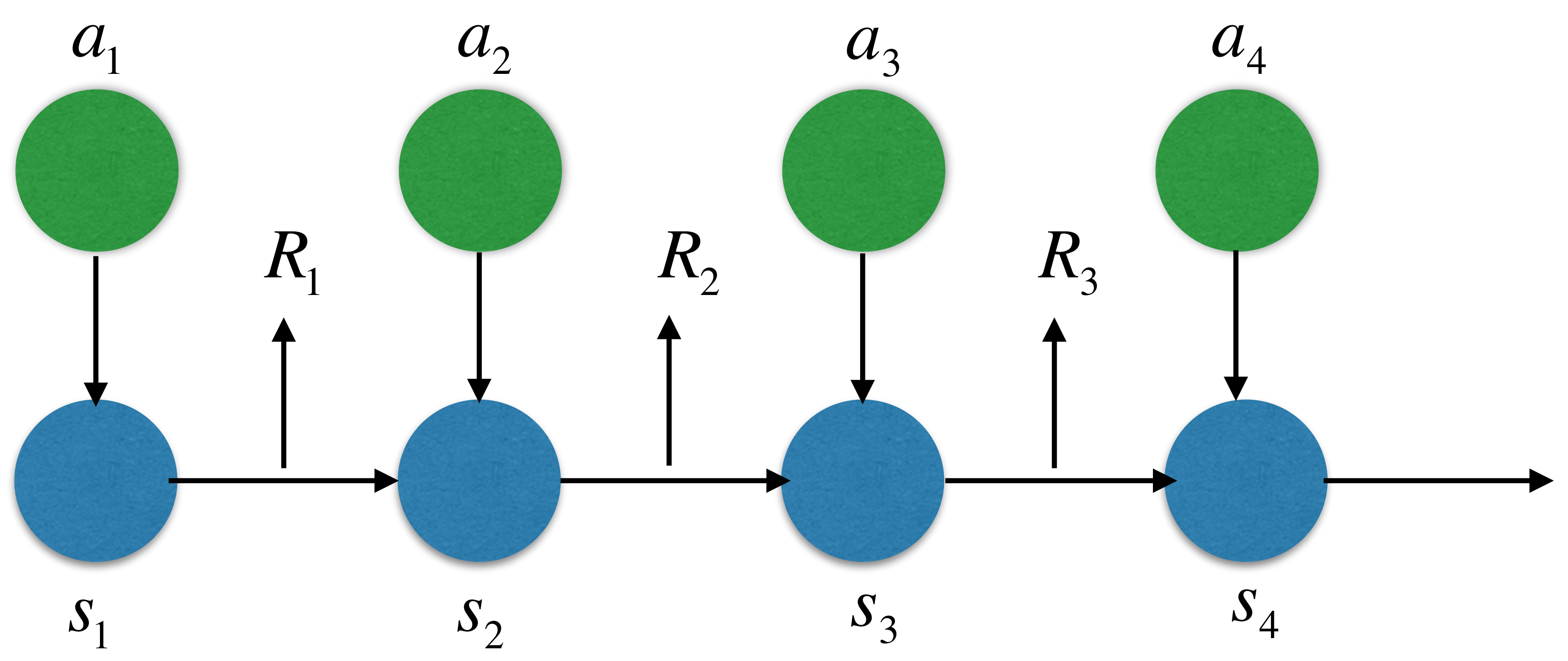
Reinforcement Learning ("**Action tasks**"): sequential (multi-step) decision-making by choosing multiple possible actions. As the state of the environment may change with time, RL involves planning and forecasting the future.

A **Feedback loop** is unique to RL, not encountered in SL or UL

The agent may not have access to streaming data from the environment (on-line learning) and learn instead in a batch mode (off-line) from data obtained from this environment.

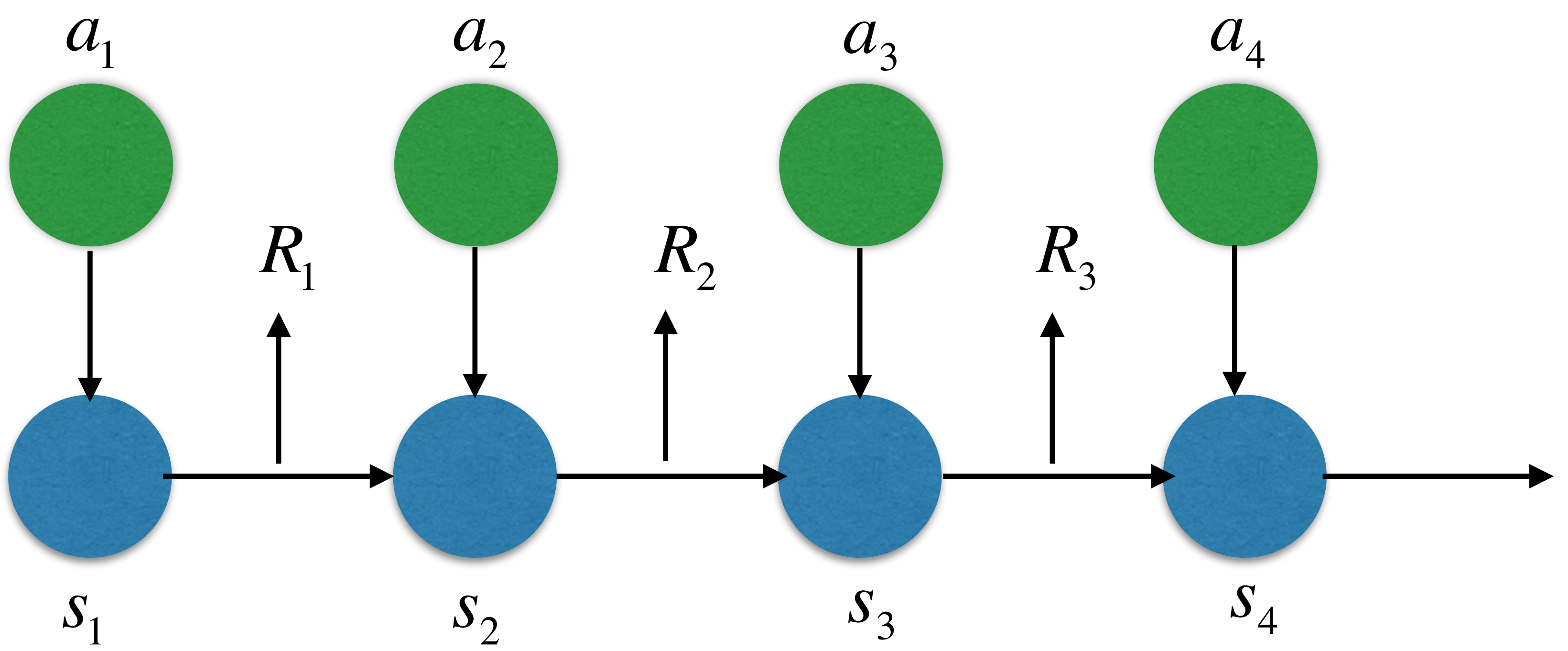
Markov Decision Processes

Markov Decision Processes: s_t - the observable environment whose dynamics can be modulated by agent's actions a_t



Markov Decision Processes

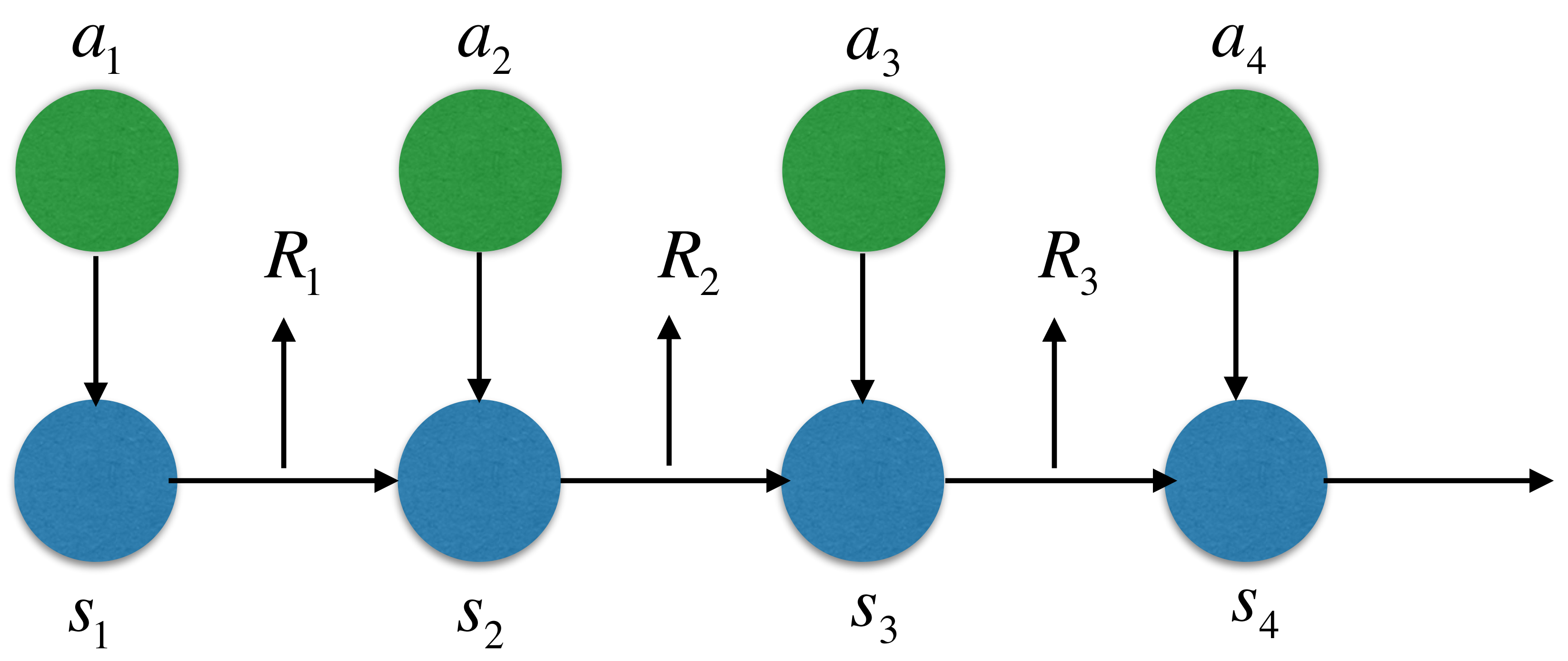
Markov Decision Processes: s_t - the observable environment whose dynamics can be modulated by agent's actions a_t



- $s_t \in S$: S is a set of **states** (discrete or continuous)
- $a_t \in A$: A is a set of **actions** (discrete or continuous)
- $p(s_{t+1} | s_t, a_t)$ are **transition probabilities**
- $R : S \times A \mapsto \mathbb{R}$ is a **reward function** (can depend on both state and action)
- $\gamma \in [0,1]$ is a **discount factor**

Markov Decision Processes

Markov Decision Processes: s_t - the observable environment whose dynamics can be modulated by agent's actions a_t

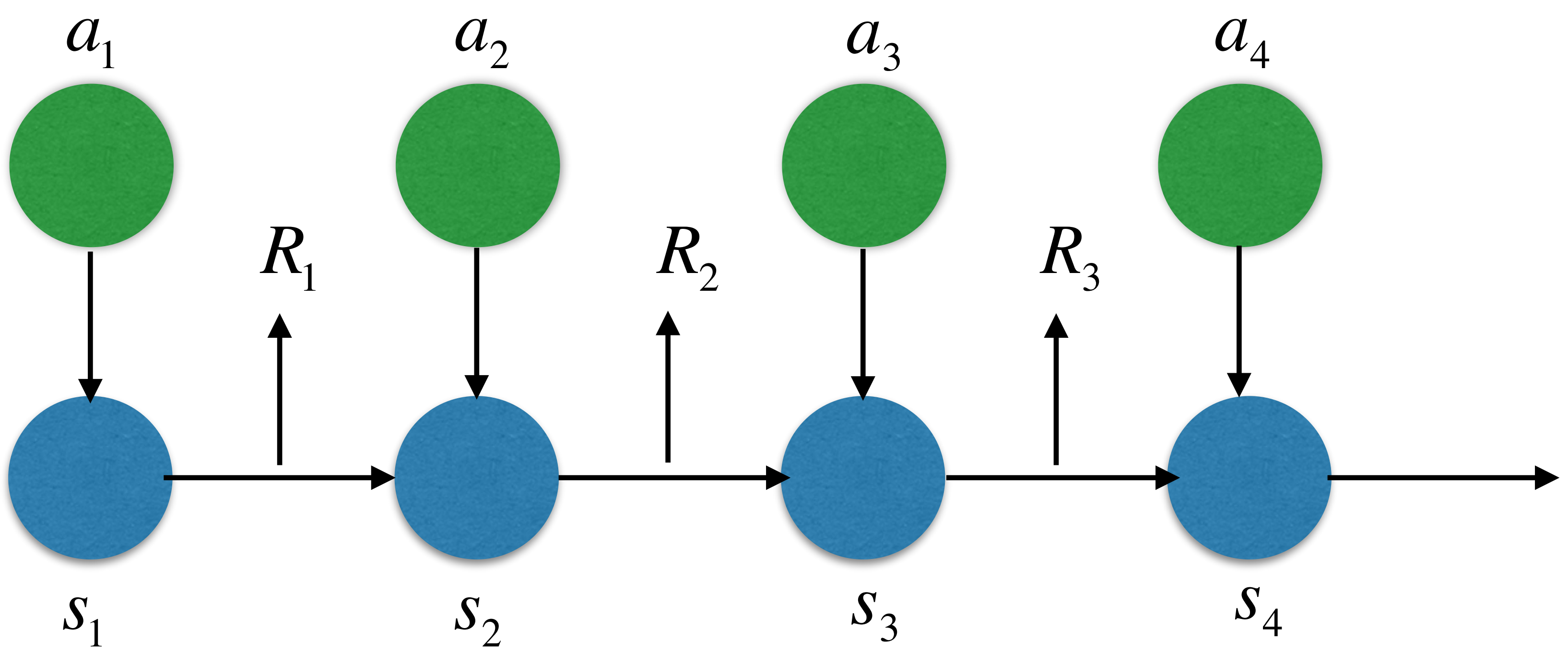


- $s_t \in S$: S is a set of **states** (discrete or continuous)
- $a_t \in A$: A is a set of **actions** (discrete or continuous)
- $p\left(s_{t+1} \mid s_t, a_t\right)$ are **transition probabilities**
- $R : S \times A \mapsto \mathbb{R}$ is a **reward function** (can depend on both state and action)
- $\gamma \in [0,1]$ is a **discount factor**
- **Cumulative total reward**

$$R(s_0,a_0)+\gamma R(s_1,a_1)+\gamma^2 R(s_2,a_2)+\ldots=\sum_n \gamma^n R(s_n,a_n)$$

RL: the objective

$$R(s_0,a_0)+\gamma R(s_1,a_1)+\gamma^2 R(s_2,a_2)+\ldots=\sum_n \gamma^n R(s_n,a_n)$$



- The **goal** in Reinforcement Learning is to **maximize the expected total reward**

$$\mathbb{E}\left[R(s_0,a_0)+\gamma R(s_1,a_1)+\gamma^2 R(s_2,a_2)+\ldots\right]=\mathbb{E}\left[\sum_n \gamma^n R(s_n,a_n)\right]$$

- This is **risk-neutral** RL (looks only at a mean of the distribution of total reward
- **Risk-sensitive** RL looks at **risk** (e.g. the variance of the total reward) as well…

Control question

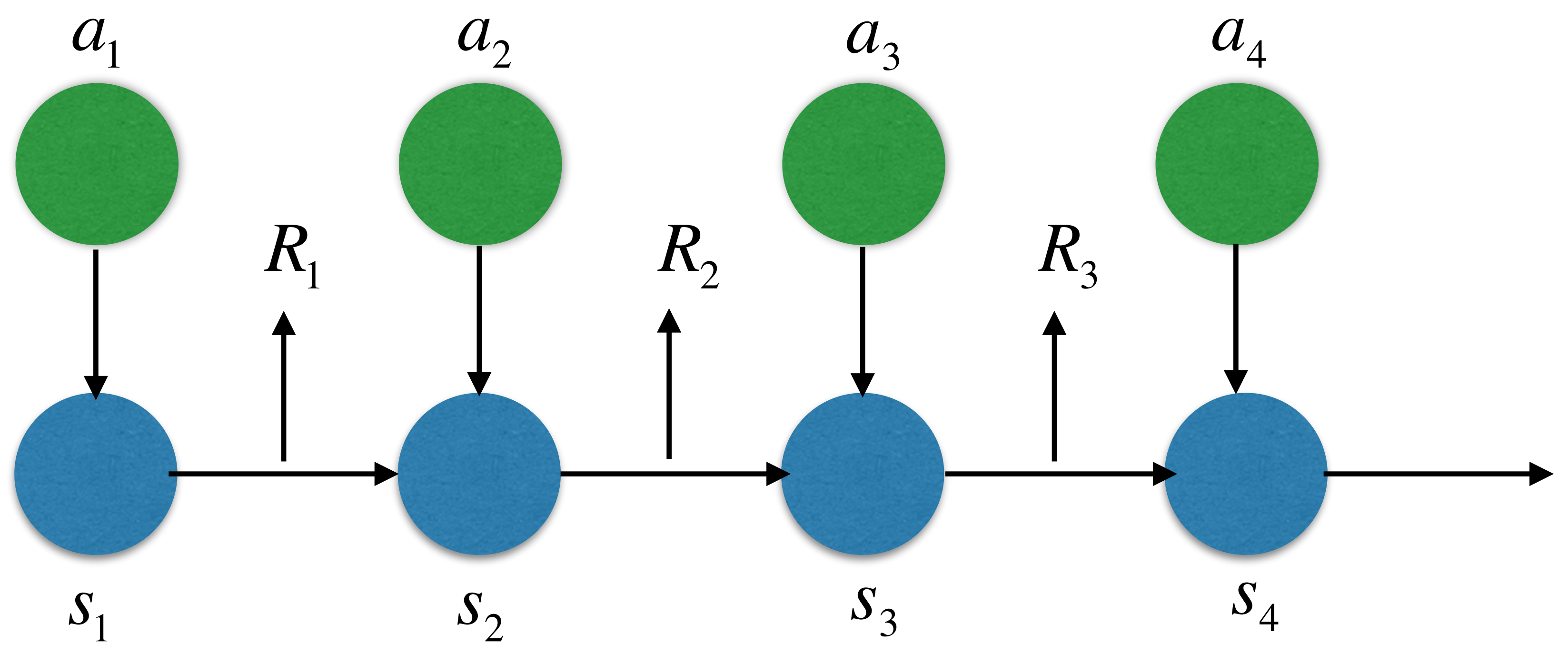
Select all correct answers

1. The name “Markov Processes” first historically appeared as a result of a misspelled name “Mark-Off Processes” that was previously used for random processes that describe learning in certain types of video games, but has become a standard terminology since then.
2. The goal of (risk-neutral) Reinforcement Learning is to maximize the expected total reward by choosing an optimal policy.
3. The goal of (risk-neutral) Reinforcement Learning is to neutralize risk, i.e. make the variance of the total reward equal zero.
4. To goal of risk-sensitive Reinforcement Learning is to teach a RL agent to pick action policies that are most prone to risk of failure. Risk-sensitive RL is used e.g. by venture capitalists and other sponsors of RL research, as a quick tool to access the feasibility of new RL projects.

Correct answers: 2

Decision policy

$$R(s_0,a_0)+\gamma R(s_1,a_1)+\gamma^2 R(s_2,a_2)+\ldots=\sum_n \gamma^n R(s_n,a_n)$$



- The **goal** in Reinforcement Learning is to **maximize the expected total reward**

$$\mathbb{E}\left[R(s_0,a_0)+\gamma R(s_1,a_1)+\gamma^2 R(s_2,a_2)+\ldots\right]=\mathbb{E}\left[\sum_n \gamma^n R(s_n,a_n)\right]$$

- This is achieved by an **optimal choice of a policy** $\pi : S \mapsto A$
- Whenever in state s_t, we take action $a_t=\pi(s_t)$
- Policy π can be deterministic or stochastic (then $\pi(s_t)$ is a probability distribution).

Deterministic vs Stochastic Policies

- **Deterministic policies**

$$a_t = \pi(s_t)$$

  - Always gives the same answer for a given state
  - In general, can depend on previous states and actions
  - For a MDP, deterministic policies depend only on the current state, because state transitions also depend only on the current state
  - For a MDP, an optimal deterministic policy π always exists

Deterministic vs Stochastic Policies

- **Deterministic policies**

$$a_t = \pi(s_t)$$

  - Always gives the same answer for a given state
  - In general, can depend on previous states and actions
  - For a MDP, deterministic policies depend only on the current state, because state transitions also depend only on the current state
  - For a MDP, an optimal deterministic policy π always exists

- **Stochastic (randomized) policies**

$$a_t \sim \pi(a \mid s_t), \quad \sum_a \pi(a \mid s_t) = 1$$

  - Generalize deterministic policies, different actions for the same state
  - For a MDP with known transition probabilities, we only need to consider deterministic policies π
  - If transition probabilities are not known, randomization of actions allow exploration for a better estimation of the model.
  - May work better than a deterministic policy for a Partially Observed Markov Decision Process (POMDP)

Exploration vs Exploitation

- An exploration-exploitation dilemma is specific to RL and does not arise for Supervised Learning or Unsupervised Learning:
- In RL, we need to take actions to collect a maximum total reward
- As the environment is unknown, we need to try different actions/states
- **Exporation:** keep looking for better actions/states, and accumulate inferior rewards
- But if we keep searching for good action/states by a trial-and-error method, we may have bad actions with low rewards

Exploration vs Exploitation

- An exploration-exploitation dilemma is specific to RL and does not arise for Supervised Learning or Unsupervised Learning:
- In RL, we need to take actions to collect a maximum total reward
- As the environment is unknown, we need to try different actions/states
- **Exporation:** keep looking for better actions/states, and accumulate inferior rewards
- But if we keep searching for good action/states by a trial-and-error method, we may have bad actions with low rewards
- **Exploitation:** keep repeating actions/revisit state with high reward - at at list of missing more rewarding actions/states
- RL should combine exploration and exploitation in some manner - how to do it optimally?
- If you deal with a batch-mode RL, someone has already solved this problem for you, though perhaps not optimally.

Control question

Select all correct answers

1. Deterministic policy is a policy $a_t = \pi\left(s_t\right)$ that gives a fixed action for each state of the world.
2. An optimal deterministic policy always exists for any Markov Decision Process (MDP)
3. An optimal deterministic policy always exists for any Partially Observable Markov Decision Process (POMDP)
4. Stochastic policies can be used for Exploration, or in settings where transition probabilities of a MDP are unknown.
5. Deterministic policies can be easily obtained from stochastic policies by fixing a random seed.
6. The Exploration-Exploitation dilemma refers to the need to both collect rewards from good known combinations of actions and states, and keep trying new actions to see if they can produce yet better rewards.

Correct answers: 1, 2, 4, 6.

Value function

- The goal in Reinforcement Learning is to maximize the expected total reward
- The value function for policy π (conditioning on the current state):

$$V_0^\pi(s) = \mathbb{E}\left[R_0(s_0, a_0) + \gamma R_1(s_1, a_1) + \ldots \mid s_0 = s, \pi\right] = \mathbb{E}\left[\sum_n \gamma^n R_n(s_n, a_n) \mid s_0 = s, \pi\right]$$

Value function and Bellman equation

- The goal in Reinforcement Learning is to maximize the expected total reward
- The value function for policy π (conditioning on the current state):

$$V_0^\pi(s) = \mathbb{E}\left[R_0(s_0,a_0) + \gamma R_1(s_1,a_1) + \ldots \mid s_0 = s, \pi\right] = \mathbb{E}\left[\sum_n \gamma^n R_n(s_n,a_n) \mid s_0 = s, \pi\right]$$

- Separate the first term from the sum:

$$V_0^\pi(s) = R_0(s_0, a_0 = \pi(s_0)) + \mathbb{E}\left[\sum_{n=1} \gamma^n R_n(s_n,a_n) \mid s_0 = s, \pi\right] \rightarrow R_0(s) + \mathbb{E}_{S'}\left[\gamma V_1^\pi(s')\right]$$

Value function and Bellman equation

- The goal in Reinforcement Learning is to maximize the expected total reward
- The value function for policy π (conditioning on the current state):

$$V_0^{\pi}(s) = \mathbb{E}\left[R_0(s_0,a_0) + \gamma R_1(s_1,a_1) + \ldots \mid s_0 = s, \pi\right] = \mathbb{E}\left[\sum_n \gamma^n R_n(s_n,a_n) \mid s_0 = s, \pi\right]$$

- Separate the first term from the sum:

$$V_0^{\pi}(s) = R_0(s_0, a_0 = \pi(s_0)) + \mathbb{E}\left[\sum_{n=1} \gamma^n R_n(s_n,a_n) \mid s_0 = s, \pi\right] \to R_0(s) + \mathbb{E}_{S'}\left[\gamma V_1^{\pi}(s')\right]$$

- The Bellman equation for value function

$$\boxed{V_t^{\pi}(s) = R_t(s_t) + \gamma \sum_{s_{t+1} \in S} p(s_{t+1} \mid s_t, a_t = \pi(s_t)) V_{t+1}^{\pi}(s_{t+1})}$$

Value function and Bellman equation

- The goal in Reinforcement Learning is to maximize the expected total reward
- The value function for policy π (conditioning on the current state):

$$V_0^\pi(s) = \mathbb{E}\left[R_0(s_0,a_0) + \gamma R_1(s_1,a_1) + \ldots \mid s_0 = s, \pi\right] = \mathbb{E}\left[\sum_n \gamma^n R_n(s_n,a_n) \mid s_0 = s, \pi\right]$$

- Separate the first term from the sum:

$$V_0^\pi(s) = R_0(s_0, a_0 = \pi(s_0)) + \mathbb{E}\left[\sum_{n=1} \gamma^n R_n(s_n,a_n) \mid s_0 = s, \pi\right] \rightarrow R_0(s) + \mathbb{E}_{S'}\left[\gamma V_1^\pi(s')\right]$$

- The Bellman equation for value function

$$\boxed{V_t^\pi(s_t) = R_t(s_t) + \gamma \sum_{s_{t+1}\in S} p(s_{t+1} \mid s_t, a_t = \pi(s_t)) V_{t+1}^\pi(s_{t+1})}$$

- The Bellman equation for value function when the reward depends on the next state S'

$$V_t^\pi(s_t) = \sum_{s_{t+1}\in S} p(s_{t+1} \mid s_t, a_t = \pi(s_t)) \left(R_t(s_t, s_{t+1}) + \gamma V_{t+1}^\pi(s_{t+1})\right)$$

Bellman equation for time-homogeneous MDPs

- The Bellman equation for value function

$$V_t^{\pi}(s) = R_t(s_t) + \gamma \sum_{s_{t+1} \in S} p(s_{t+1} \mid s_t, a_t = \pi(s_t)) V_{t+1}^{\pi}(s_{t+1})$$

- For time-homogeneous MDPs, transition probabilities and rewards do not depend on time, and the time horizon is infinite. In such cases, the value function does not depend on time $V_t^{\pi}(s) \rightarrow V^{\pi}(s)$

- The Bellman equation for a time-independent value function is equivalent to a system of N linear equations (where N is the number of discrete states)

$$\boxed{V^{\pi}(s_t) = R(s_t) + \gamma \sum_{s_{t+1} \in S} p(s_{t+1} \mid s_t, a_t = \pi(s_t)) V^{\pi}(s_{t+1})}$$

The optimal value function

- The value function for policy π

$$V_0^\pi(s_0) = \mathbb{E}\left[R_0(s_0, a_0) + \gamma R_1(s_1, a_1) + \ldots \mid s_0 = s, \pi\right] = \mathbb{E}\left[\sum_n \gamma^n R_n(s_n, a_n) \mid s_0 = s, \pi\right]$$

- The Bellman equation for value function

$$V_t^\pi(s) = R_t(s_t) + \gamma \sum_{s_{t+1} \in S} p(s_{t+1} \mid s_t, a_t = \pi(s_t)) V_{t+1}^\pi(s_{t+1})$$

- The **optimal value function:** $V_t^*(s_t) = \max_\pi V_t^\pi(s_t)$

- Optimality means that

$$V^*(s) = V^{\pi^*}(s) \geq V^\pi(s), \boxed{\forall \pi \neq \pi^*}$$

The optimal value function

- The value function for policy π

$$V_0^{\pi}(s_0) = \mathbb{E}\left[R_0(s_0, a_0) + \gamma R_1(s_1, a_1) + \ldots \mid s_0 = s, \pi\right] = \mathbb{E}\left[\sum_n \gamma^n R_n(s_n, a_n) \mid s_0 = s, \pi\right]$$

- The Bellman equation for value function

$$V_t^{\pi}(s) = R_t(s_t) + \gamma \sum_{s_{t+1} \in S} p(s_{t+1} \mid s_t, a_t = \pi(s_t)) V_{t+1}^{\pi}(s_{t+1})$$

- The **optimal value function:** $V_t^*(s_t) = \max_{\pi} V_t^{\pi}(s_t)$

- The **Bellman optimality equation for optimal value function** $V^*(s)$

$$V_t^*(s_t) = R_t(s_t) + \max_{a_t \in A} \gamma \sum_{s_{t+1} \in S} p(s_{t+1} \mid s_t, a_t) V_{t+1}^*(s_{t+1})$$

The optimal value function

- The value function for policy π

$$V_0^{\pi}(s_0) = \mathbb{E}\left[R_0(s_0,a_0) + \gamma R_1(s_1,a_1) + \ldots \mid s_0 = s, \pi\right] = \mathbb{E}\left[\sum_n \gamma^n R_n(s_n,a_n) \mid s_0 = s, \pi\right]$$

- The Bellman equation for value function

$$V_t^{\pi}(s) = R_t(s_t) + \gamma \sum_{s_{t+1}\in S} p(s_{t+1} \mid s_t, a_t = \pi(s_t)) V_{t+1}^{\pi}(s_{t+1})$$

- The **optimal value function:** $V_t^*(s_t) = \max_{\pi} V_t^{\pi}(s_t)$

- The **Bellman optimality equation for optimal value function** $V^*(s)$

$$V_t^*(s_t) = R_t(s_t) + \max_{a_t\in A} \gamma \sum_{s_{t+1}\in S} p(s_{t+1} \mid s_t, a_t) V_{t+1}^*(s_{t+1})$$

- The **optimal policy:**

$$\pi_t(s_t) = \arg\max_{a_t\in A} \sum_{s_{t+1}\in S} p(s_{t+1} \mid s_t, a_t) V_t^*(s_{t+1})$$

Control question

Select all correct answers

1. The value function depends on the current state and the policy used to collect rewards
2. The value function depends on the whole history of state transitions.
3. The optimal value function is a value function that has the lowest complexity among all value function, to prevent overfitting.
4. The optimal value function is a maximum of all possible value functions for different policies among all possible policies
5. The Bellman optimality equation is a system of linear equations
6. The Bellman optimality equation is a non-linear equation that generally needs to be solved numerically

Correct answers: 1, 4, 6

Value iteration

- The **Bellman equation for optimal value function**

$$V^*(s) = R(s) + \max_{a\in A} \gamma \sum_{s'\in S} p(s' \mid s,a) V^*(s')$$

- **Value Iteration** algorithm (for discrete state-action space)**:**
  - Initialize the value function for each state $V(s) = V^{(0)}(s)$
  - Repeat the update of the value function until convergence:

$$V^{(k+1)}(s) = R(s) + \max_{a\in A} \gamma \sum_{s'\in S} p(s' \mid s,a) V^{(k)}(s')$$

Value iteration

- The **Bellman equation for optimal value function**

$$V^*(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} p(s' \mid s, a) V^*(s')$$

- **Value Iteration** algorithm (for discrete state-action space)**:**
  - Initialize the value function for each state $V(s) = V^{(0)}(s)$
  - Repeat the update of the value function until convergence:

$$V^{(k+1)}(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} p(s' \mid s, a) V^{(k)}(s')$$

- Synchronous updates: finish until the end of iteration, then update the value function for all states at once
- Asynchronous updates: update the value function on the fly

Value iteration

- The **Bellman equation for optimal value function**

$$V^*(s) = R(s) + \max_{a\in A} \gamma \sum_{s'\in S} p(s' \mid s,a) V^*(s')$$

- **Value Iteration** algorithm (for discrete state-action space)**:**
  - Initialize the value function for each state $V(s) = V^{(0)}(s)$
  - Repeat the update of the value function until convergence:

$$V^{(k+1)}(s) = R(s) + \max_{a\in A} \gamma \sum_{s'\in S} p(s' \mid s,a) V^{(k)}(s')$$

- Synchronous updates: finish until the end of iteration, then update the value function for all states at once
- Asynchronous updates: update the value function on the fly

- Optimal policy: $\pi^*(s) = \arg\max_{a\in A} \sum_{s'\in S} p(s' \mid s,a) V^*(s')$

Policy iteration

- The Bellman equation for the value function

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} p(s' | s, a = \pi(s)) V^{\pi}(s')$$

- **Policy Iteration** algorithm**:**
  - Initialize policy randomly $\pi(s) = \pi^{(0)}(s)$
  - Repeat the update of the value function until convergence:
    - Policy evaluation: Compute $V^{\pi}(s)$ from the Bellman equation for the value function
    - Iterate policy $\pi^{(k+1)}(s) = \arg\max_{a \in A} \sum_{s' \in S} p(s' | s, a) V^{(\pi)}(s')$

Policy iteration

- The Bellman equation for the value function

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} p(s' \mid s, a = \pi(s)) V^{\pi}(s')$$

- **Policy Iteration** algorithm**:**
  - Initialize policy randomly $\pi(s) = \pi^{(0)}(s)$
  - Repeat the update of the value function until convergence:
    - Policy evaluation: Compute $V^{\pi}(s)$ from the Bellman equation for the value function
    - Iterate policy $\pi^{(k+1)}(s) = \arg\max_{a \in A} \sum_{s' \in S} p(s' \mid s, a) V^{(\pi)}(s')$

- The policy iteration step can be done using standard optimization software
- The policy evaluation is critical, as it requires solving Bellman equation multiply times - can be costly for large state spaces

Control question

Select all correct answers

1. The optimal policy is a policy for which the Bellman equation is fastest to solve.
2. The policy evaluation step can be done using standard optimization software
3. Optimal policy calculation in Value Iteration is computationally the same as a policy iteration step in Policy Iteration method.

Correct answers: 3

Action-State value function

- The value function for policy π depends only on the current state, does not tell a RL agent what it should do

$$V_t^\pi(s) = \mathbb{E}\left[\sum_n \gamma^n R(s_n, a_n) \mid s_0 = s, \pi\right]$$

- The action-state value function (the Q-function): take action a, then follow π

$$Q_0^\pi(s,a) = \mathbb{E}\left[\sum_n \gamma^n R_n(s_n, a_n) \mid s_0 = s,\, a_0 = a, \pi\right] = R_0(s,a) + \mathbb{E}\left[\sum_{n=1} \gamma^n R_n(s_n, a_n) \mid \pi\right]$$

Action-State value function

- The value function for policy π depends only on the current state, does not tell a RL agent what it should do

$$V_t^{\pi}(s) = \mathbb{E}\left[\sum_n \gamma^n R(s_n, a_n) \mid s_0 = s, \pi\right]$$

- The action-state value function (the Q-function): take action a, then follow π

$$Q_0^{\pi}(s,a) = \mathbb{E}\left[\sum_n \gamma^n R_n(s_n, a_n) \mid s_0 = s,\, a_0 = a, \pi\right] = R_0(s,a) + \mathbb{E}\left[\sum_{n=1} \gamma^n R_n(s_n, a_n) \mid \pi\right]$$

- The Bellman equation for the Q-value function

$$Q_t^{\pi}(s_t, a_t) = R_t(s_t, a_t) + \mathbb{E}_{S_{t+1}}\left[\gamma V_{t+1}^{\pi}(s_{t+1})\right]$$

Action-State value function

- The value function for policy π depends only on the current state, does not tell a RL agent what it should do

$$V_t^{\pi}(s) = \mathbb{E}\left[\sum_n \gamma^n R(s_n, a_n) \mid s_0 = s, \pi\right]$$

- The action-state value function (the Q-function): take action a, then follow π

$$Q_0^{\pi}(s,a) = \mathbb{E}\left[\sum_n \gamma^n R_n(s_n, a_n) \mid s_0 = s,\ a_0 = a, \pi\right] = R_0(s,a) + \mathbb{E}\left[\sum_{n=1} \gamma^n R_n(s_n, a_n) \mid \pi\right]$$

- The Bellman equation for the Q-value function

$$Q_t^{\pi}(s_t, a_t) = R_t(s_t, a_t) + \mathbb{E}_{S_{t+1}}\left[\gamma V_{t+1}^{\pi}(s_{t+1})\right]$$

- The optimal Q-function:

$$Q_t^{*}(s_t, a_t) = \max_{\pi} Q_t^{\pi}(s_t, a_t)$$

Action-State value function

- The value function for policy π depends only on the current state, does not tell a RL agent what it should do

$$V_t^{\pi}(s) = \mathbb{E}\left[\sum_n \gamma^n R(s_n, a_n) \mid s_0 = s, \pi\right]$$

- The action-state value function (the Q-function): take action a, then follow π

$$Q_0^{\pi}(s,a) = \mathbb{E}\left[\sum_n \gamma^n R_n(s_n, a_n) \mid s_0 = s,\, a_0 = a, \pi\right] = R_0(s,a) + \mathbb{E}\left[\sum_{n=1} \gamma^n R_n(s_n, a_n) \mid \pi\right]$$

- The Bellman equation for the Q-value function

$$Q_t^{\pi}(s_t, a_t) = R_t(s_t, a_t) + \mathbb{E}_{S_{t+1}}\left[\gamma V_{t+1}^{\pi}(s_{t+1})\right]$$

- The optimal Q-function:

$$Q_t^{*}(s_t, a_t) = \max_{\pi} Q_t^{\pi}(s_t, a_t)$$

- The optimal V-function:

$$V_t^{*}(s_t) = \max_{a_t \in A} Q_t^{*}(s_t, a_t)$$

- The Bellman equation for the optimal Q-function

$$Q_t^{*}(s_t, a_t) = R_t(s_t, a_t) + \mathbb{E}_{S'}\left[\gamma V_{t+1}^{*}(s_{t+1}, a_t)\right]$$

$$Q_t^{*}(s_t, a_t) = R_t(s_t, a_t) + \max_{a_t \in A} \mathbb{E}_{S'}\left[\gamma Q_{t+1}^{*}(s_{t+1}, a_t)\right]$$

Learning from experience

- Both the State Value function $V^{\pi}(s)$ and Action-State Value function $Q^{\pi}(s,a)$ can be estimated from experience

- An Agent can follow policy π and compute the average reward for each visited state s. This average would converge to the value function $V^{\pi}(s)$ at this state.

- If averages are computed separately for each action taken, we can estimate the action-state value function $Q^{\pi}(s,a)$ in the same way.

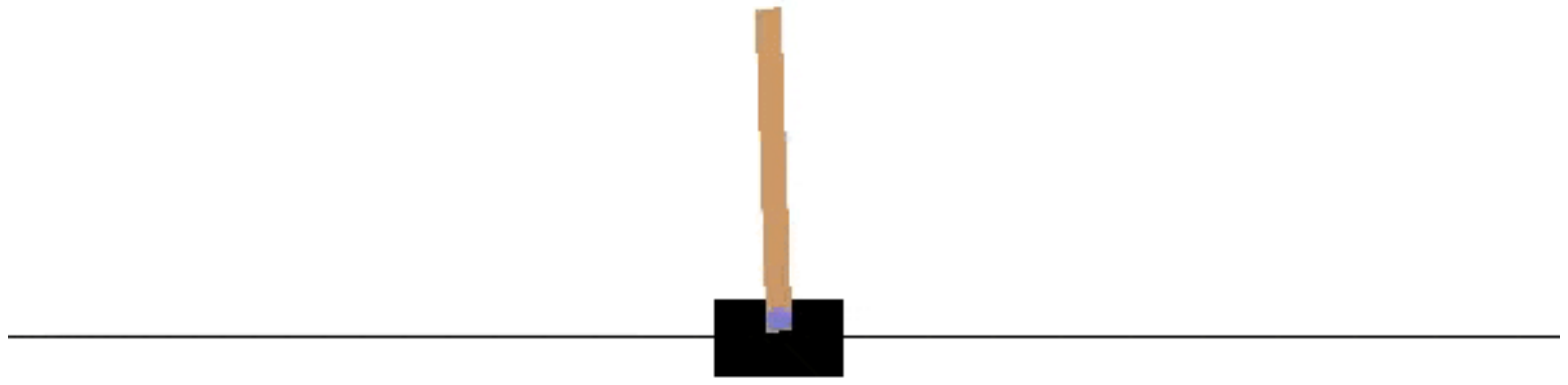
- Estimations of this kind are called Monte Carlo methods in Reinforcement Learning.

- If there are many states, it may be not practical to keep separate averages for each state - need to rely on Function Approximation.

- Use parameterized functions for Value Functions in such cases.

Pole balancing example

- The task: apply forces to a cart moving along a track so as to keep a pole hinged to the cart from falling over.
- A failure: the pole falls past a given angle from the vertical
- This is the **episodic task**: episodes are repeated attempts to balance the pole, for a given time, or until it falls
- Rewards: +1 for each step failure did not occur.
- Can also be formulated as a **continuing task** (no fixed end time), with a discounting. Reward is -1 for each failure, and zero otherwise.
- The MDP formulation: States are positions and velocities of the cart and the pole. Actions are forces applied to the cart.



Source: https://cdn-images-1.medium.com/max/1600/1*oMSg2_mKguAGKy1C64UFlw.gif

Control question

Select all correct answers

1. Monte Carlo methods in Reinforcement Learning directly estimate the Q-function by summing observed rewards along trajectories.
2. An action-value function (Q-function) is a function of a current state and current action, unlike a value function that is only a function of a current state.
3. An optimal value function is obtained by maximization of a Q-function with respect to the state variable.
4. An optimal value function is obtained by maximization of a Q-function with respect to the action variable.
5. Episodic tasks continue until a Q-function keeps maximizing. Because the time it takes depends on initial conditions, different episodes generally take different times.

Correct answers: 1,2,4