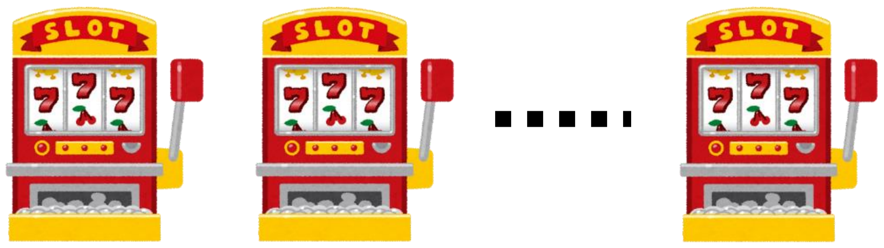
Brain Inspired Artificial Intelligence
2: Markov Decision Process

Eiji Uchibe
Dept. of Brain Robot Interface
ATR Computational Neuroscience Labs.

Reminder: Bandit Problem

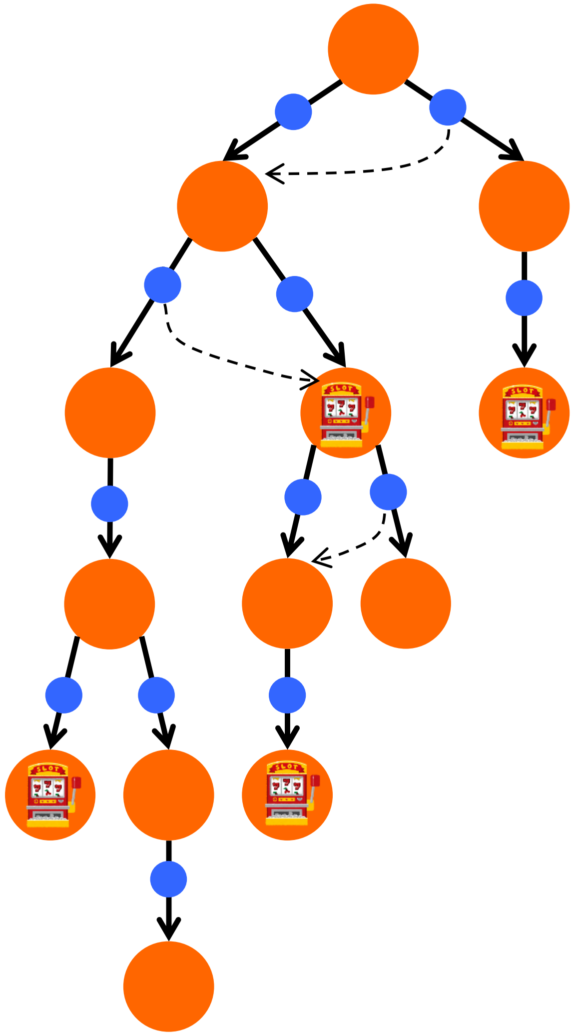
- Bandit problems: one state, many actions
- The goal is to find an optimal policy $\pi(a)$ that maximize the expected reward
- The value based approach estimates the value function $q_\pi(a)$
- The policy based approach directly search the optimal policy



- Now, we will study Markov Decision Processes (MDPs) for sequential decision making
  - MDPs formally describe an environment for reinforcement learning

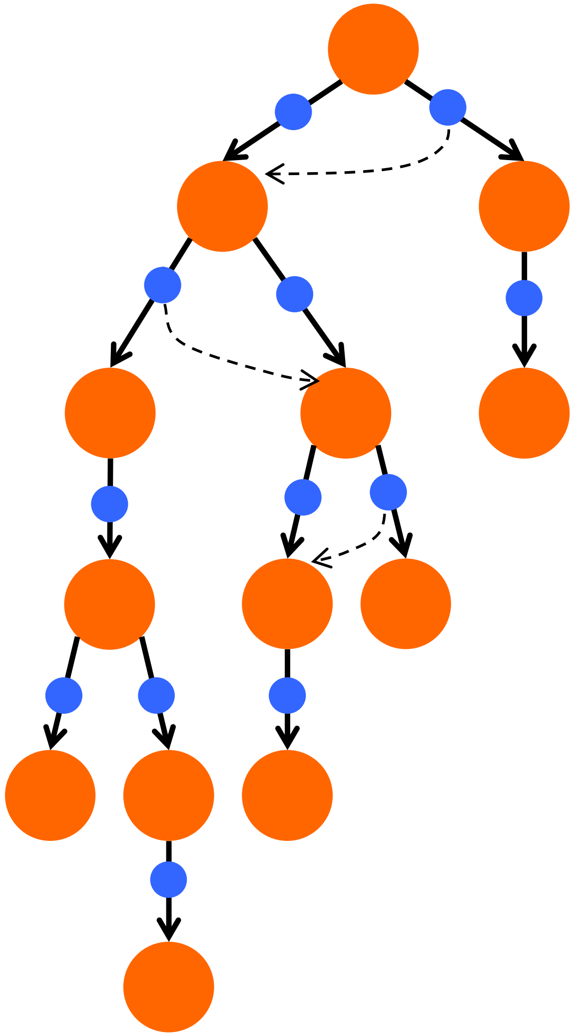
Extension to Sequential Decision Making Problems

- A learning agent is not in front of gambling machines
- To get non-zero rewards, the agent has to move to one of machines
- A current position is considered as a state
  - The state is a sufficient statistics to describe the dynamics of the environment
  - Bandits are MDPs with one state
- The action of the agent influences the environment, causing a state transition

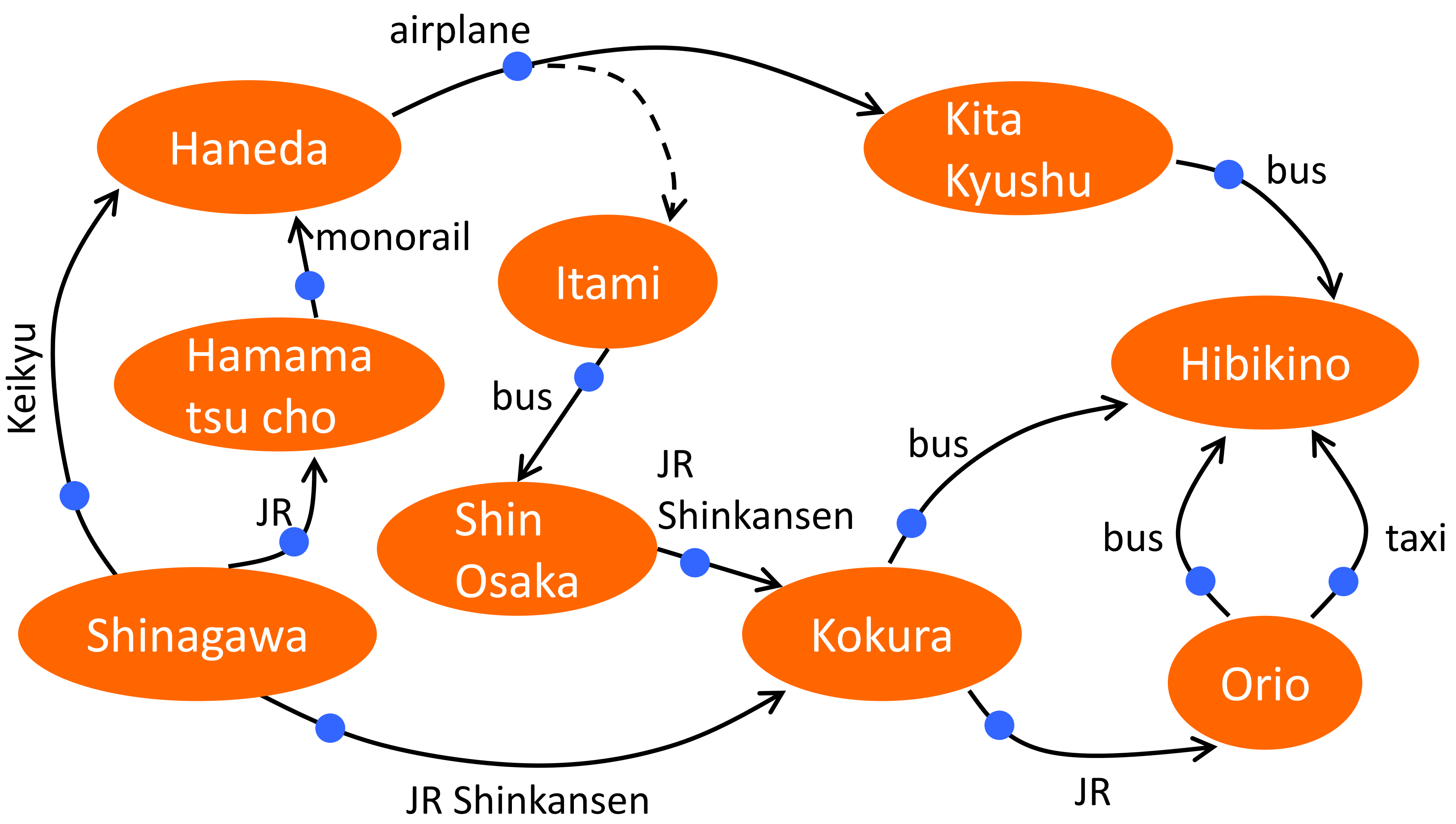


Extension to the Sequential Decision Making Problems

- s: state,
- a: action,
- State set: $\mathcal{S} = \{s_1, s_2, \dots, s_{|\mathcal{S}|}\}$
- Action set: $\mathcal{A} = \{a_1, a_2, \dots, a_{|\mathcal{A}|}\}$
- $p_0(s)$: initial state distribution
- $p(s', r \mid s, a)$: stochastic environmental dynamics
- $\pi(\, a \mid s \,)$: policy, probability to select an action a for state s

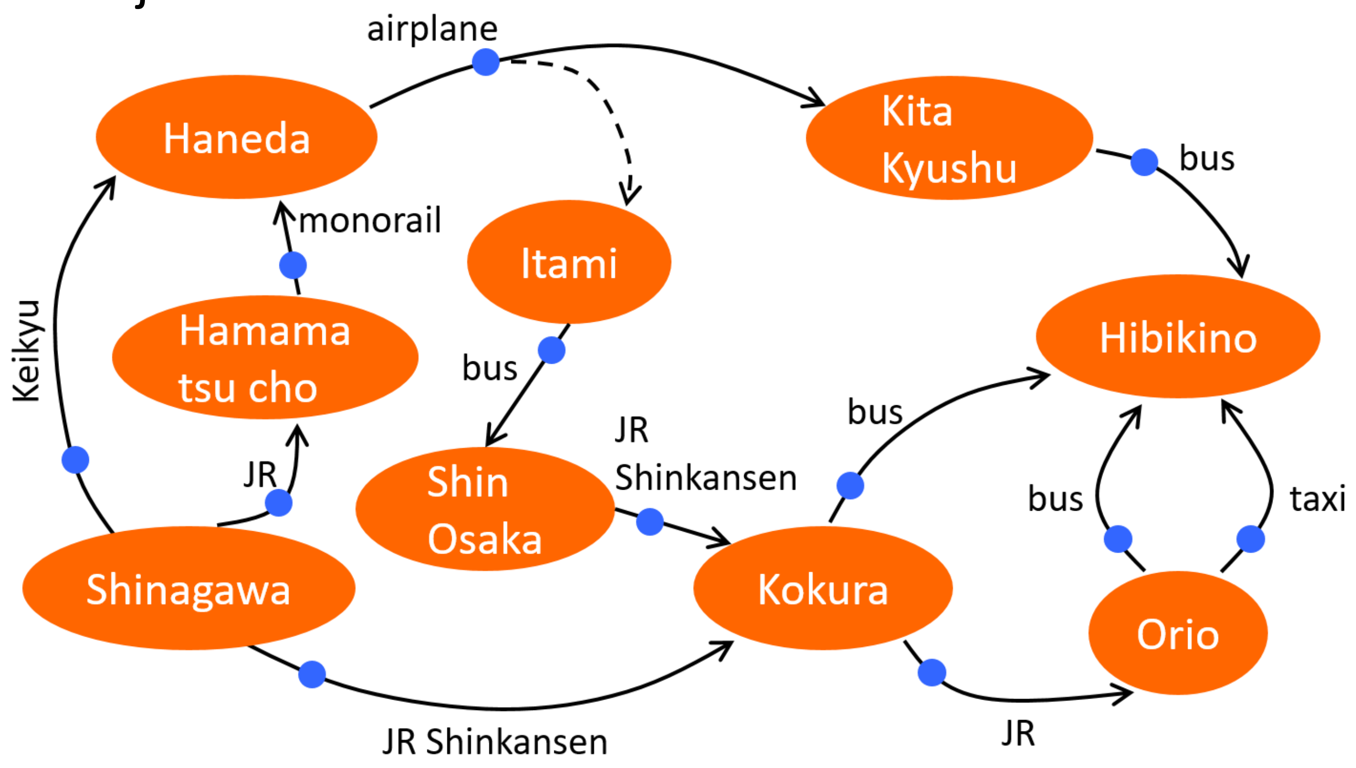


Example: From Shinagawa to Hibikino Campus



States and Actions in the Example

- $\mathcal{S}$={Shinagawa, Hamamatsu cho, Haneda, Itami, Shin Osaka, Kitakyushu, Kokura, Orio, Hibikino}
- Action set is state dependent in this case
  - $\mathcal{A}$(Shinagawa) = {Keikyu, JR Shinkansen}
  - $\mathcal{A}$(Kokura) = {bus, JR Shinkansen}
  - $\mathcal{A}$(Shinagawa) = {Airplane, JR Shinkansen}
  - ⋮



Markovian Assumption

- Consider a sequence of states, action, and rewards from $t = 0$ to t

$$S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{t-1}, A_{t-1}, R_t, S_t, A_t$$

- How can we model the next state S_{t+1} and the reward R_{t+1}? The most general way is to introduce a conditional probability distribution

$$\Pr(S_{t+1} = s', R_{t+1} = r \mid S_0, A_0, R_1, \ldots, S_{t-1}, A_{t-1}, R_t, S_t, A_t)$$

- It is too complicated! We usually **assume** that the environment satisfies the following Markov property:

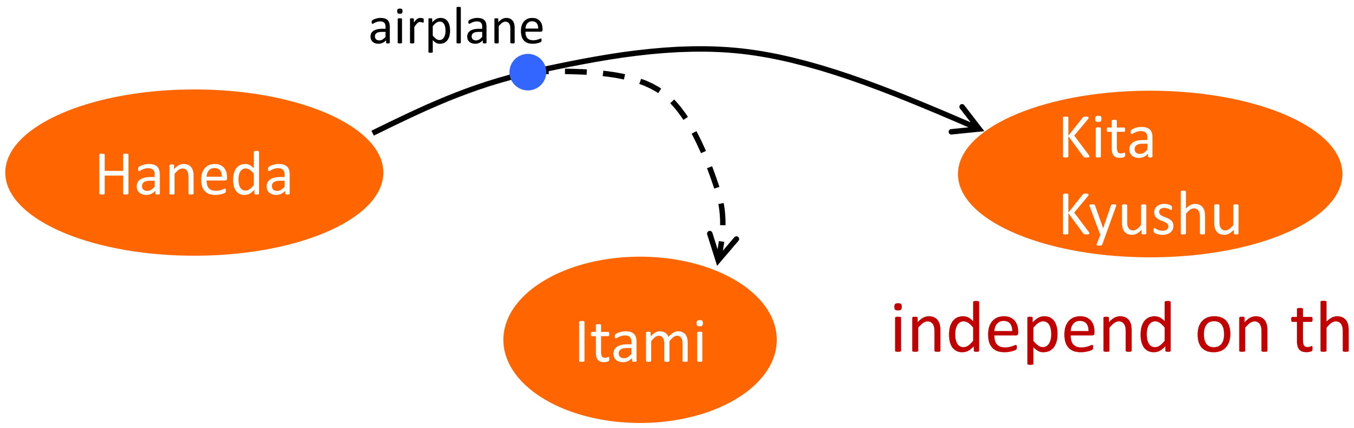
$$p(s', r \mid s, a) = \Pr(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a)$$

State Representation is Important

- Consider a stochastic state transition
  - Pr(Kita Kyushu|Haneda, airplane) = 0.9
  - Pr(Itami|Haneda, airplane)=0.1
- Markovian property
  - Pr(Kita Kyushu|Haneda, airplane)

    =Pr(Kita Kyushu|Haneda, airplane, Shinagawa, Keikyu)

    =Pr(Kita Kyushu|Haneda, airplane, Hamamatsu cho, monorail, Shinagawa, JR)



independ on the route

State Representation is Important

- Mario's position $\boldsymbol{p}_t$ at time t is not uniquely determined if we use the game screen as state

$$\Pr(\boldsymbol{p}_{t+1} \mid \boldsymbol{p}_t, a_t) \neq \Pr(\boldsymbol{p}_{t+1} \mid \boldsymbol{p}_t, a_t, \boldsymbol{p}_{t-1})$$

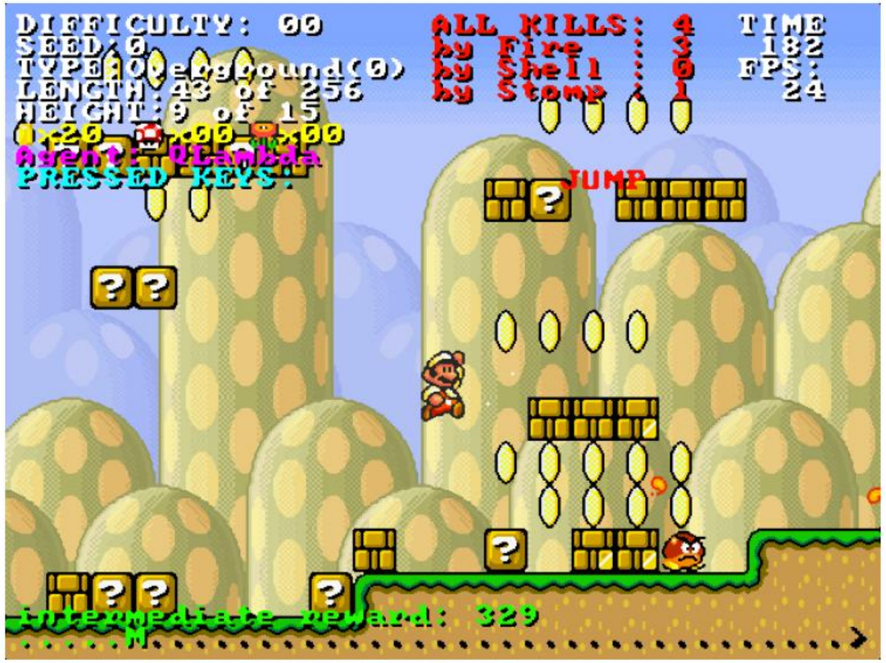
  – For example, one defines the state by

$$s_t = (\boldsymbol{p}_t, \boldsymbol{p}_{t-1})$$

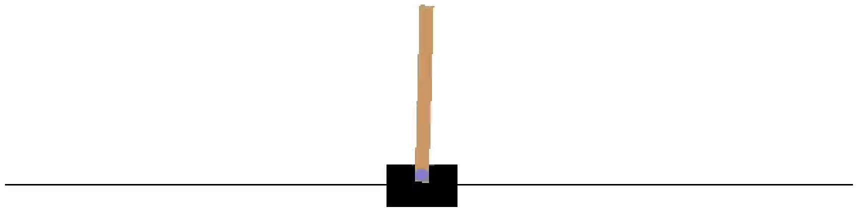
  – s_t contains the information of the velocity of Mario

- If a state is given by the position and velocity of the cart and the angle and angular velocity of the pole, it satisfies Markov assumption

$$s_t = (x_t, \dot{x}_t, \theta, \dot{\theta})$$

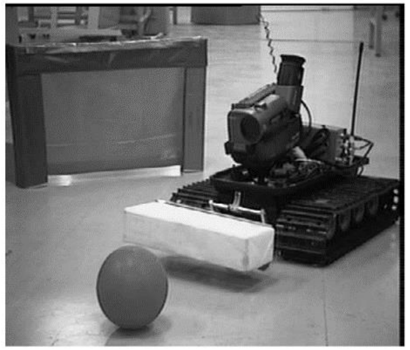


Mario benchmark

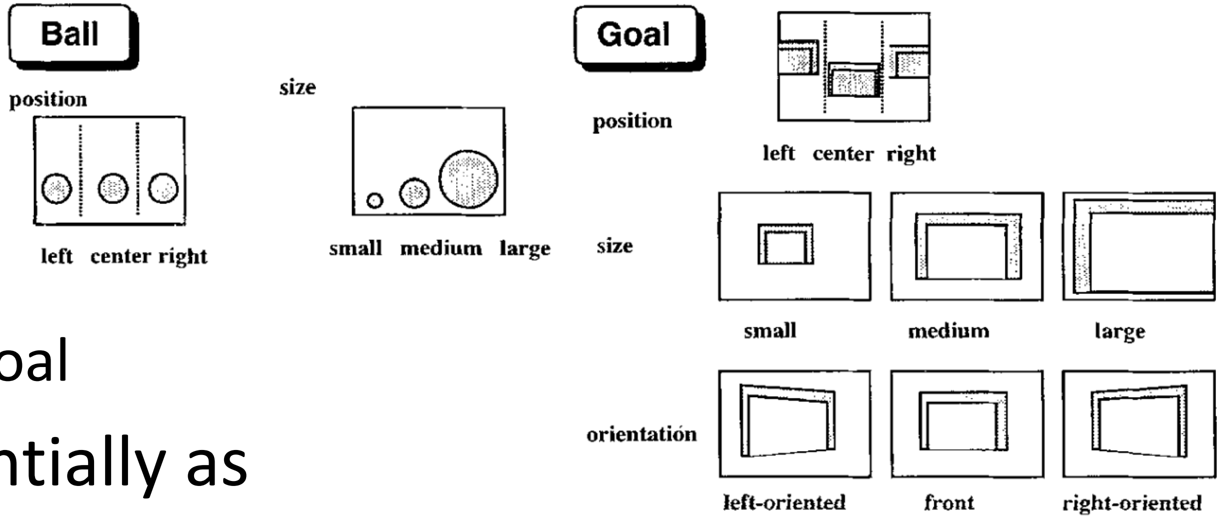


cart-pole system

Discretizing State Space



- When a state is given by a continuous vector, it is discretized manually
  - Example of the soccer robot
  - Task is to shoot a ball into a goal
  - 5 state variables:
    x-position and size of the ball and
    x-position, size and orientation of the goal



- The number of states grows exponentially as the number of features increases

$$3^2 \times 3^3 = 243$$

the number of ball states

the number of goal states

the total number of states

[Asada et al., 1996]

Components

- State transition probability: $p(s' \mid s, a) = \sum_r p(s', r \mid s, a)$

- Expected reward for state-action pairs:

$$r(s,a) = \sum_r r p(r \mid s, a) \quad \text{where} \quad p(r \mid s, a) = \sum_{s'} p(s', r \mid s, a)$$

- Expected reward for state-action-next-state triples

$$r(s,a,s') = \sum_r r p(r \mid s, a, s') \quad \text{where} \quad p(r \mid s, a, s') = \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}$$

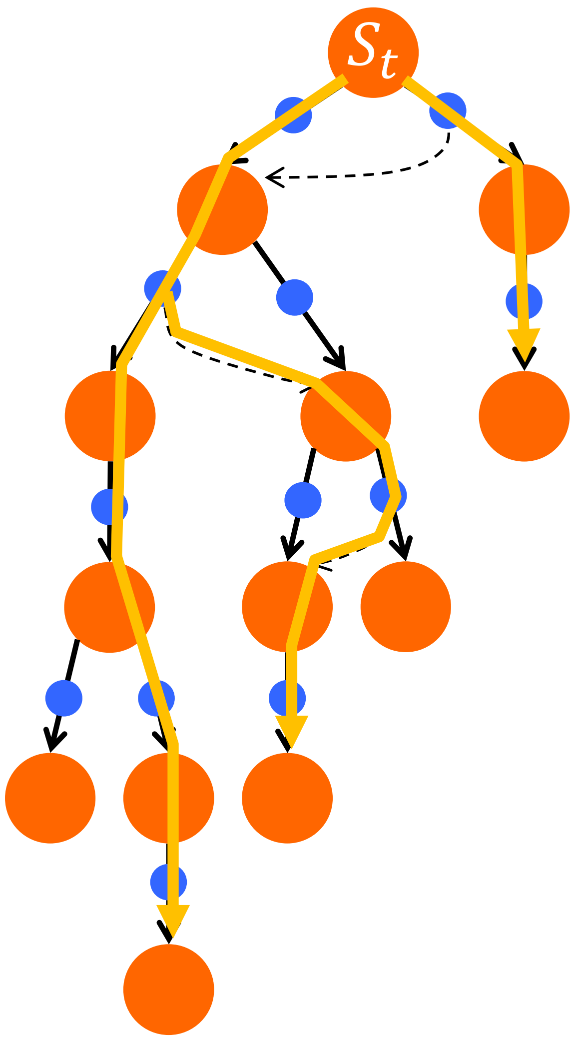
Return

- Consider a sequence of states, action, and rewards
- The return G_t is defined as the sum of discounted rewards from time-step t

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

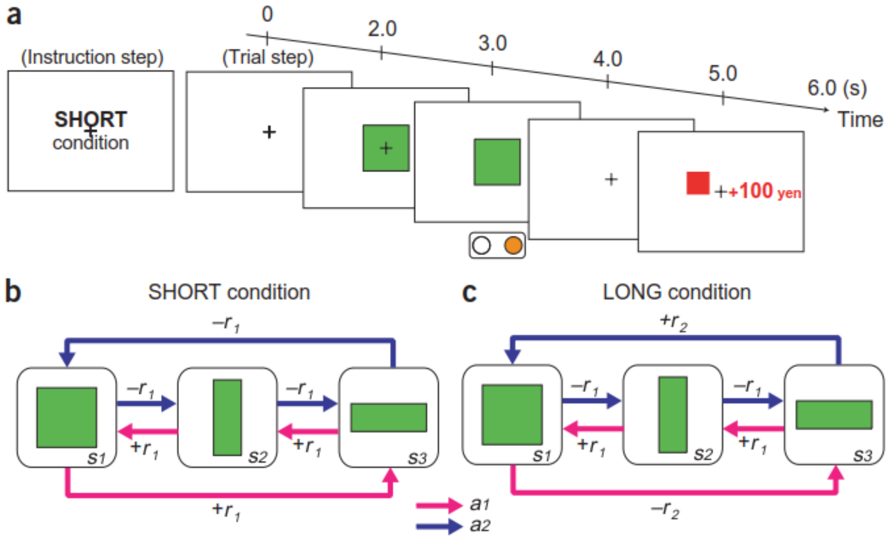
$$= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

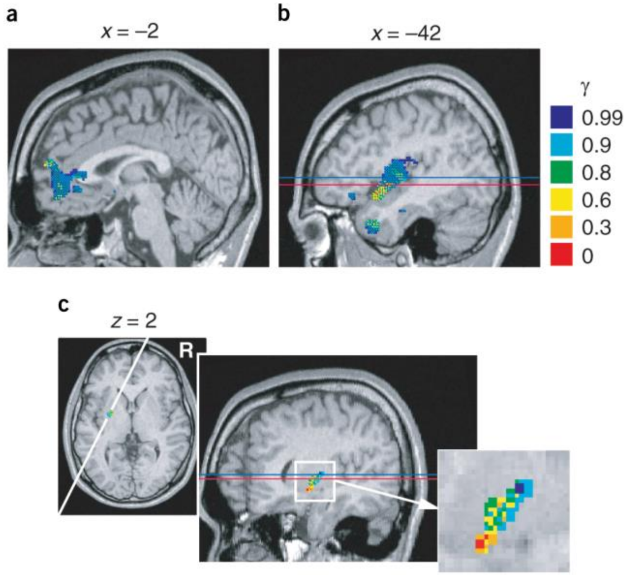
  - G_t is a random variable
- $\gamma \in [0, 1)$ is the discount rate
  - γ close to 0: myopic evaluation
  - γ close to 1: far-sighted evaluation



Why Discount?

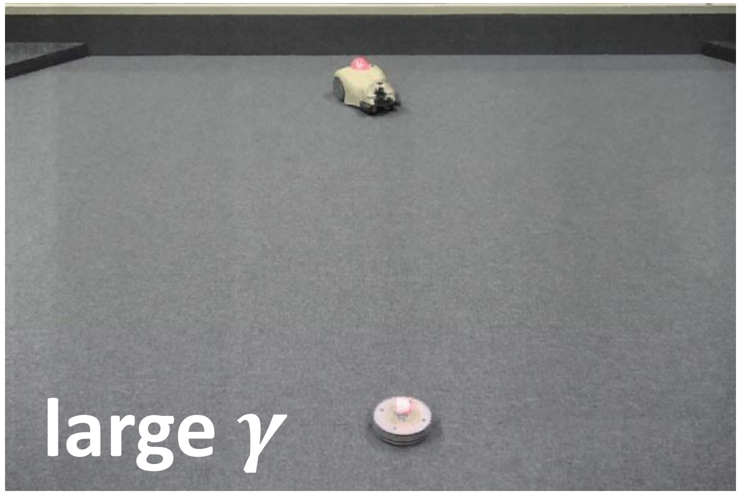
- If the reward is bounded, the return is also bounded: $|R| \leq R_{\max} \Rightarrow |G| \leq \dfrac{R_{\max}}{1-\gamma}$
- Prediction of immediate and future rewards differentially recruits cortico-basal ganglia loops





[Tanaka et al., 2004]

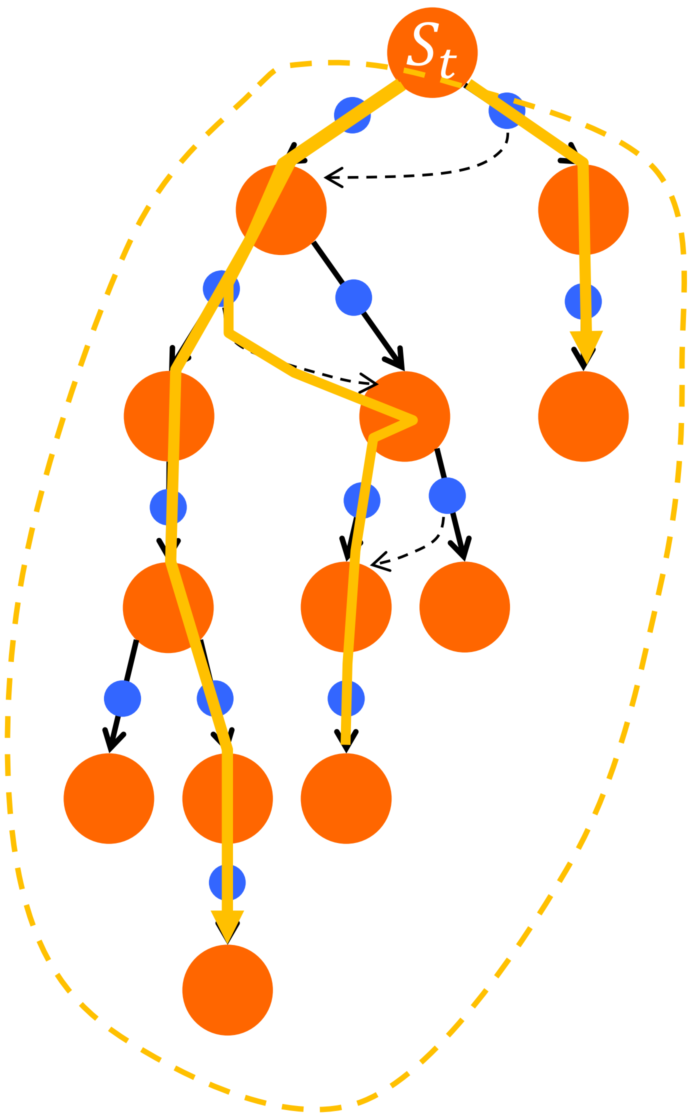
The robot does not move towards the battery



The robot tries to catch the battery

State Value Function

- A state value function evaluates the policy in each state
- For a given stationary policy π, consider a state-action-reward sequence
$$S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, S_{t+2} \dots$$
generated by
  - $A_t \sim \pi(a \mid S_t)$
  - $S_{t+1}, R_{t+1} \sim p(s', r \mid, S_t, A_t)$
- The state value function is the expected return given by
$$v_\pi(s) \triangleq \mathbb{E}_\pi[G_t \mid S_t = s, A_{t:\infty} \sim \pi]$$



State Value Function

- The value of a state s under a policy π is defined by

$$v_\pi(s) \triangleq \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,|S_t = s\right]$$

- $\mathbb{E}_\pi\{\quad\}$ denotes the expectation which is taken over the probability distribution

$$P(\,A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, S_{t+2}, \ldots \mid S_t = s\,)$$

$$= \prod_{k=0}^{\infty} \pi(A_{t+k}|S_{t+k}) P_T(S_{t+k+1}, R_{t+k+1}|S_{t+k}, A_{t+k})$$

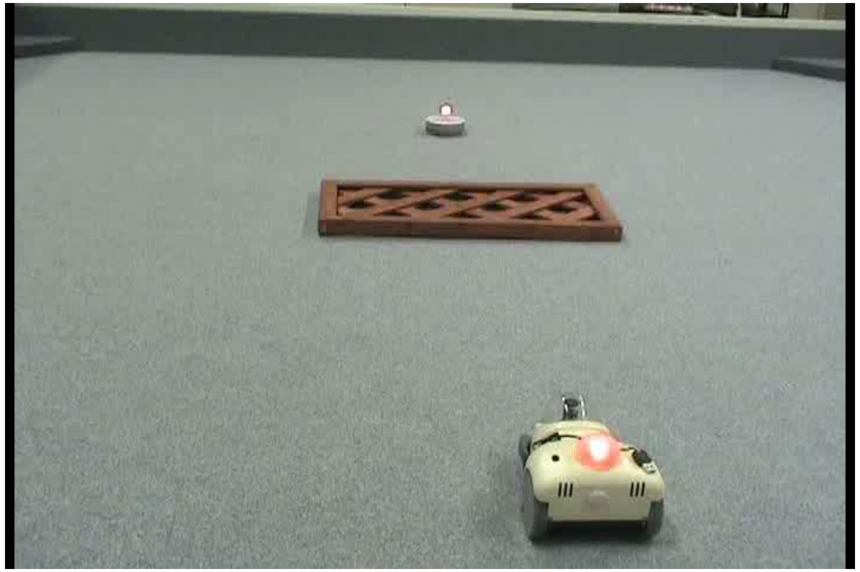
- Therefore,

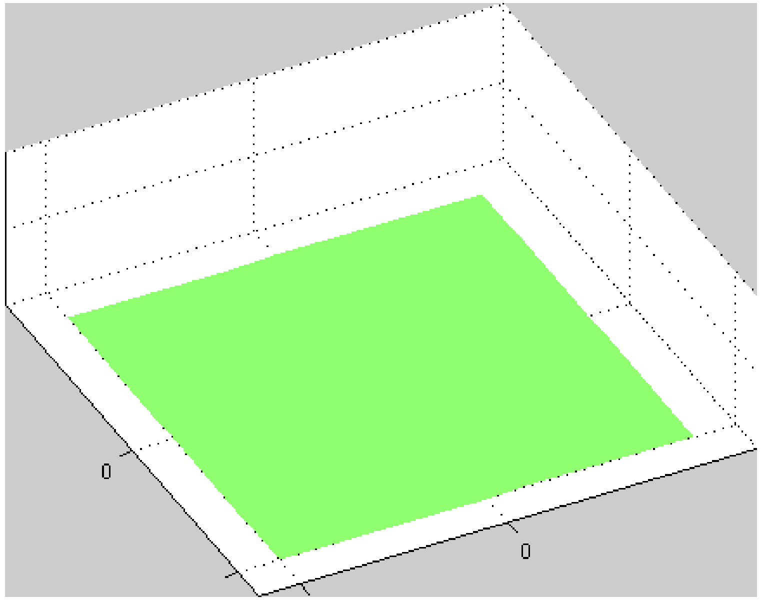
$$v_\pi(s) = \sum_{A_t, R_{t+1}, S_{t+1}, \ldots} P(A_t, R_{t+1}, S_{t+1}, \ldots |S_t = s) \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right]$$

Example of State Value Function

- Task: to get the battery pack while avoiding collisions with an obstacle
  - Positive reward for successful battery catching
  - Negative reward for collisions with obstacles
  - Small negative reward for every time step

The goal state has the highest value



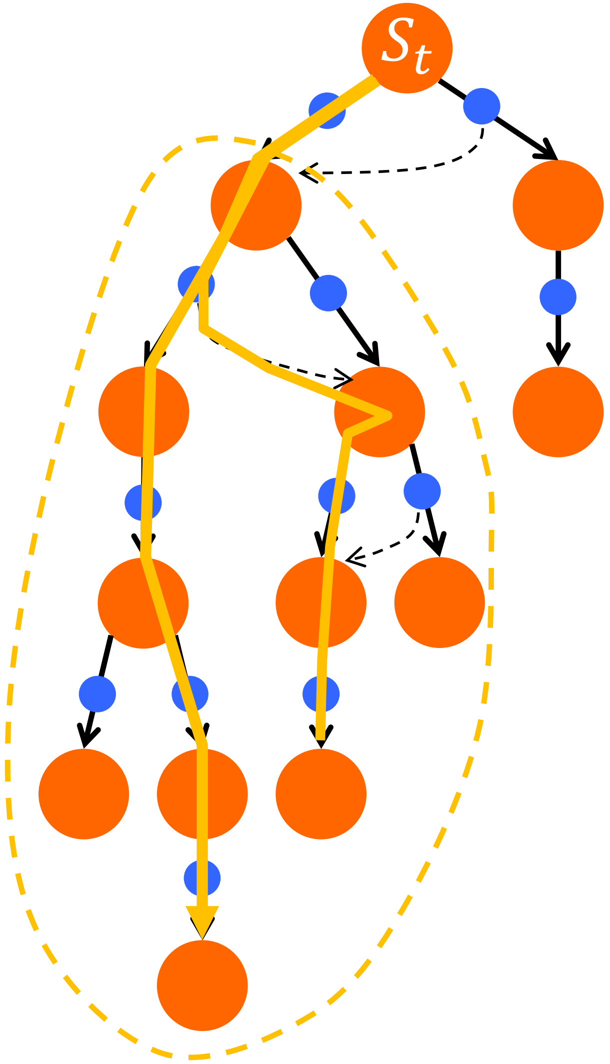


state value function

State-Action Value Function

- A state-action value function used in the bandit problem is extended to a state-action value function
- It evaluates the all actions in each state
- It is the expected return starting from s, taking the action a, and thereafter following policy π
  - $S_{t+1}, R_{t+1} \sim p(s', r \mid, S_t, A_t)$
  - $A_{t+1} \sim \pi(a \mid S_{t+1})$
- The state-action value function is the expected return given by

$$q_\pi(s, a) \triangleq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$



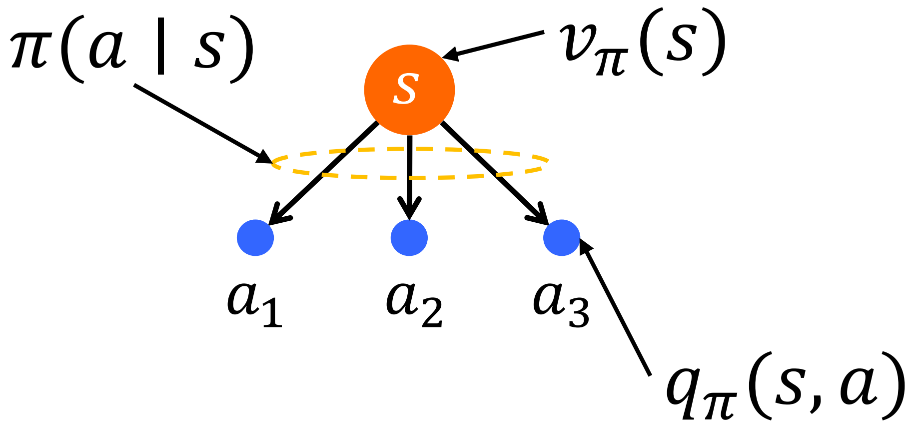
State-Action Value Function

- Similarly, the value of taking action a in a state s under a policy π is defined by
$$q_\pi(s,a) \triangleq \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\,|S_t = s, A_t = a\right]$$
- $\mathbb{E}_\pi\{\quad\}$ denotes the expectation which is taken over the probability distribution
$$P(\,R_{t+1}, S_{t+1}, A_{t+1}, S_{t+2}, \dots \mid S_t = s, A_t = a\,)$$
$$= \prod_{k=0}^{\infty} p(S_{t+k+1}, R_{t+k+1}|S_{t+k}, A_{t+k})\pi(A_{t+k+1}|S_{t+k+1})$$
- Note the difference between the distributions
- Therefore,
$$q_\pi(s,a) = \sum_{R_{t+1},S_{t+1},A_{t+1},\dots} P(R_{t+1}, S_{t+1}, \dots |S_t = s, A_t = a)\left[\sum_{k=0}^{\infty}\gamma^k\, R_{t+k+1}\right]$$

Relations between v_π and q_π

- Relation between $v_\pi(s)$ and $q_\pi(s,a)$

$$v_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a \mid s) q_\pi(s,a)$$
$$= \mathbb{E}_\pi[q_\pi(s,a)]$$



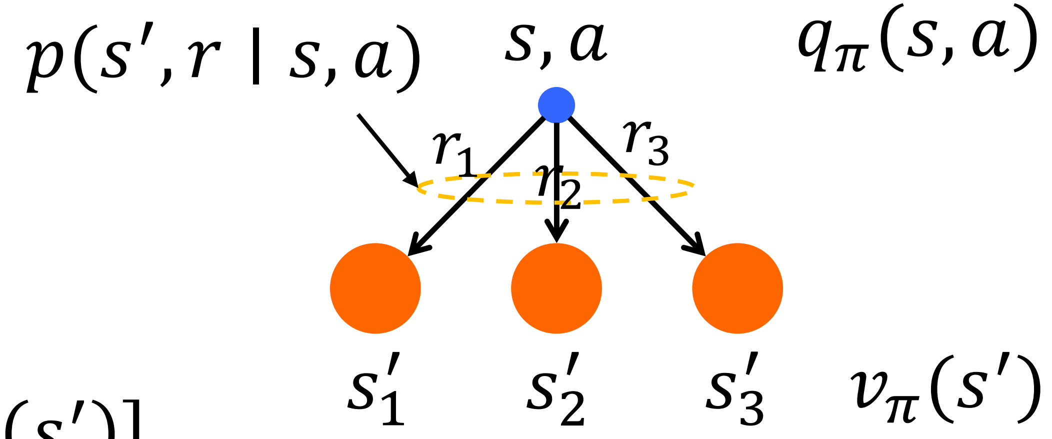
- Advantage function

$$A_\pi(s,a) \triangleq q_\pi(s,a) - v_\pi(s)$$

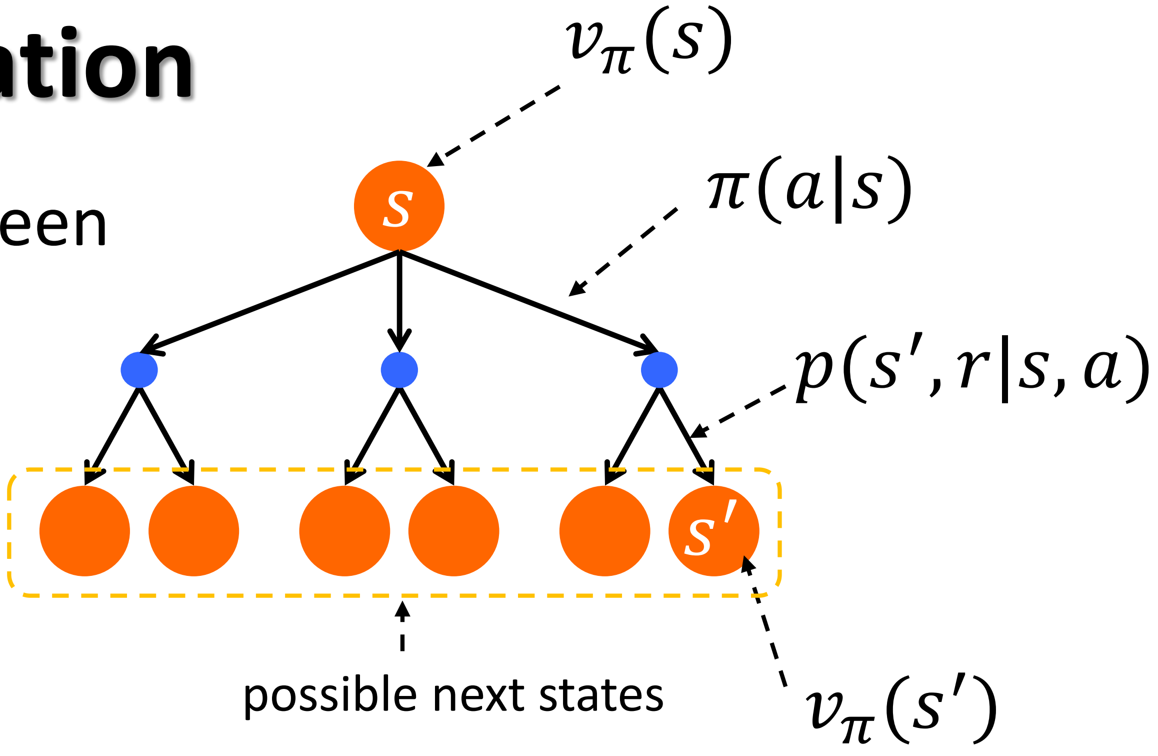
  – From the definition, $\mathbb{E}_\pi[A_\pi(s,a)] = 0$

- Relation between $q_\pi(s,a)$ and $v_\pi(s')$

$$q_\pi(s,a) = \sum_{s',r} p(s',r \mid s,a)[r + \gamma v_\pi(s')]$$



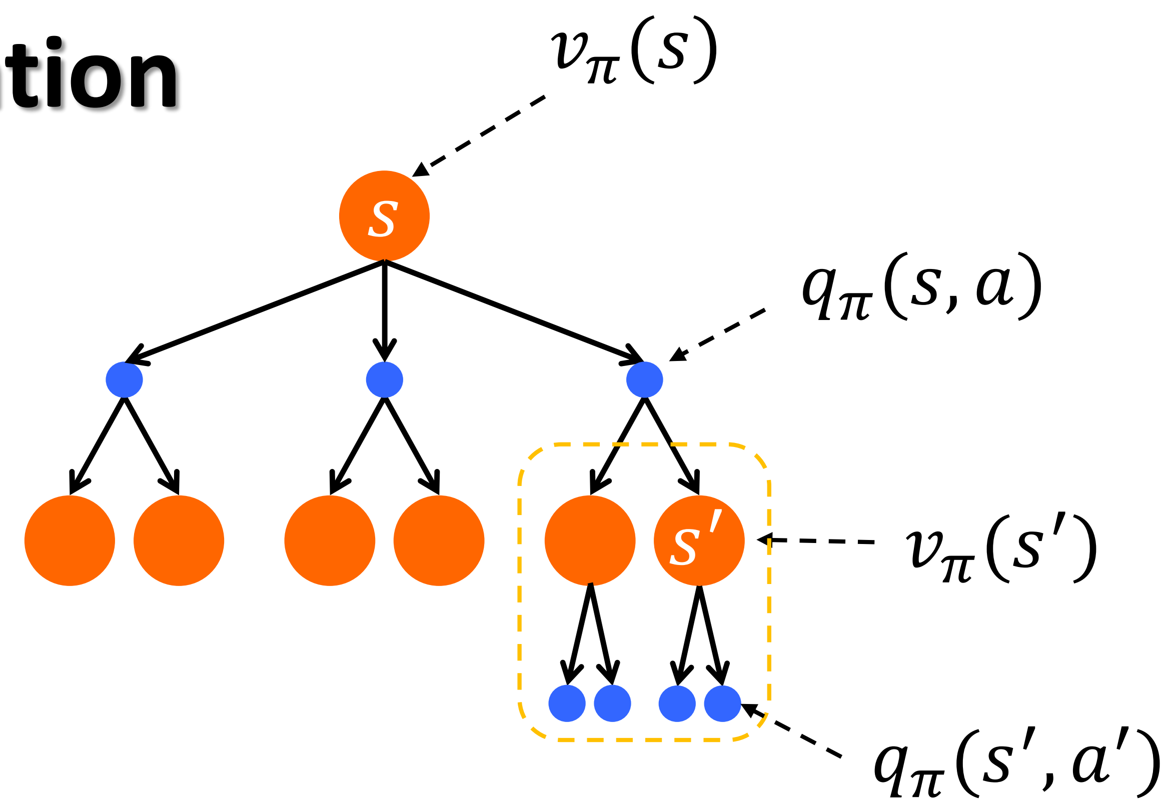
Bellman (Expectation) Equation



- Recursive relationship of values between the current and the next state
- Linear with respect to v_π
- For any policy π and any state s,

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a \mid s) \sum_{s',r} p(s',r \mid s,a)\left[r + \gamma \mathbb{E}_\pi[\,G_{t+1} \mid S_{t+1} = s'\,]\right] \\
&= \sum_a \pi(a \mid s) \sum_{s',r} P(s',r \mid s,a)\left[r + \gamma v_\pi(s')\right]
\end{aligned}
$$

Bellman (Expectation) Equation



- Similarly, we obtain the recursive relationships for q_π
- Linear with respect to q_π
- For any policy π and any state-action pair (s,a),

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

$$= \sum_{s',r} p(s',r \mid s,a)\left[r + \gamma \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s', A_{t+1} = a']\right]$$

$$= \sum_{s',r} P(s',r \mid s,a)\left[r + \gamma \sum_{a'} \pi(a' \mid s') q_\pi(s',a')\right]$$

Optimal Value Functions

- Reminder: The goal of RL is to find an optimal policy π_*
- v_* is an optimal state value function for π^* defined by

$$v_*(s) \triangleq \max_\pi v_\pi(s) \qquad \forall s \in \mathcal{S}$$

- Similarly, an optimal state-action value function is defined by

$$q_*(s,a) \triangleq \max_\pi q_\pi(s,a) \qquad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$$

Theorem

- A value function defines a partial ordering over policies

$$\pi \geq \pi' \quad \text{if} \quad v_\pi(s) \geq v_{\pi'}(s), \quad \text{for all } s$$

- For any Markov Decision Process

1. There exists an optimal policy π_* that is better than or equal to all other policies, $\pi_* \geq \pi$ for all π
2. All optimal policies achieve the optimal value function,

$$v_{\pi_*}(s) = v_*(s)$$

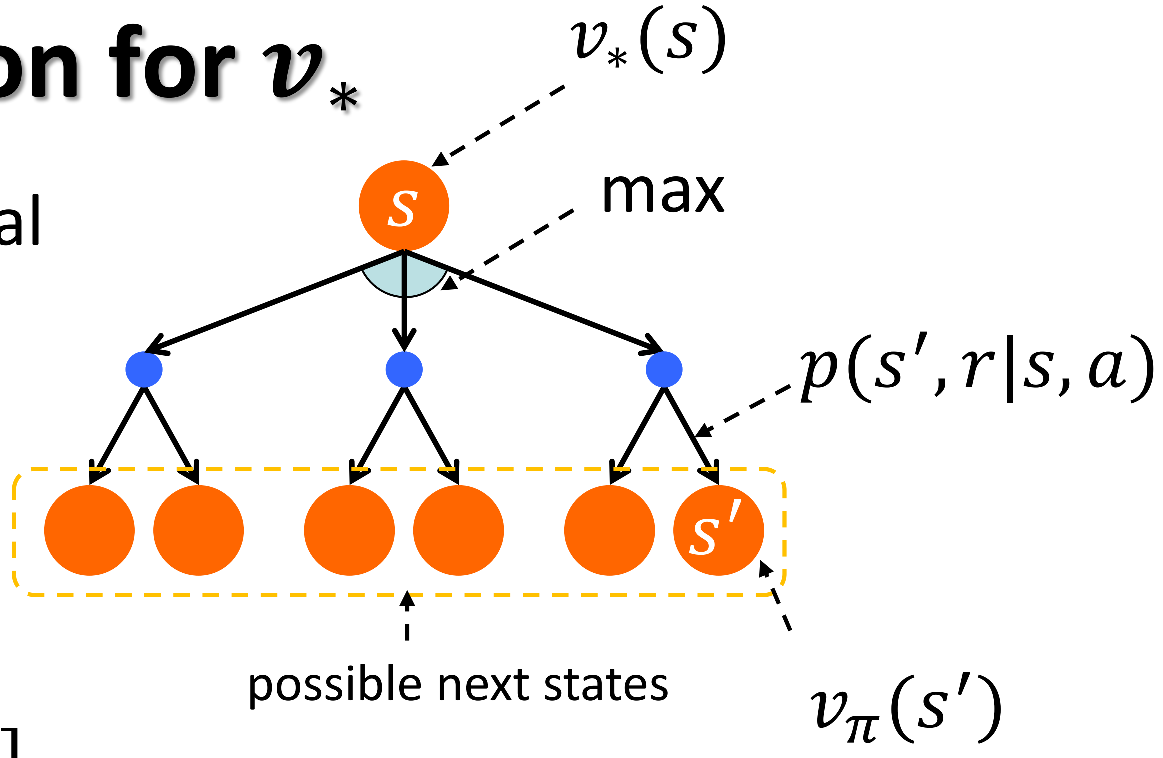
3. All optimal policies achieve the optimal state-action value funciton

$$\underbrace{q_{\pi_*}(s,a)}_{\substack{\text{state-action value} \\ \text{function of the optimal policy}}} = \underbrace{q_*(s,a)}_{\substack{\text{optimal state-action} \\ \text{value function}}}$$

Bellman Optimality Equation for v_*

- Recursive relationships for the optimal state value function
- Nonlinear with respect to v_*
- For any state s,

$$
\begin{aligned}
v_*(s) &= \max_a q_*(s,a) \\
&= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_t \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \max_a \sum_{s',r} p(s',r \mid s,a)[r + \gamma v_*(s')]
\end{aligned}
$$



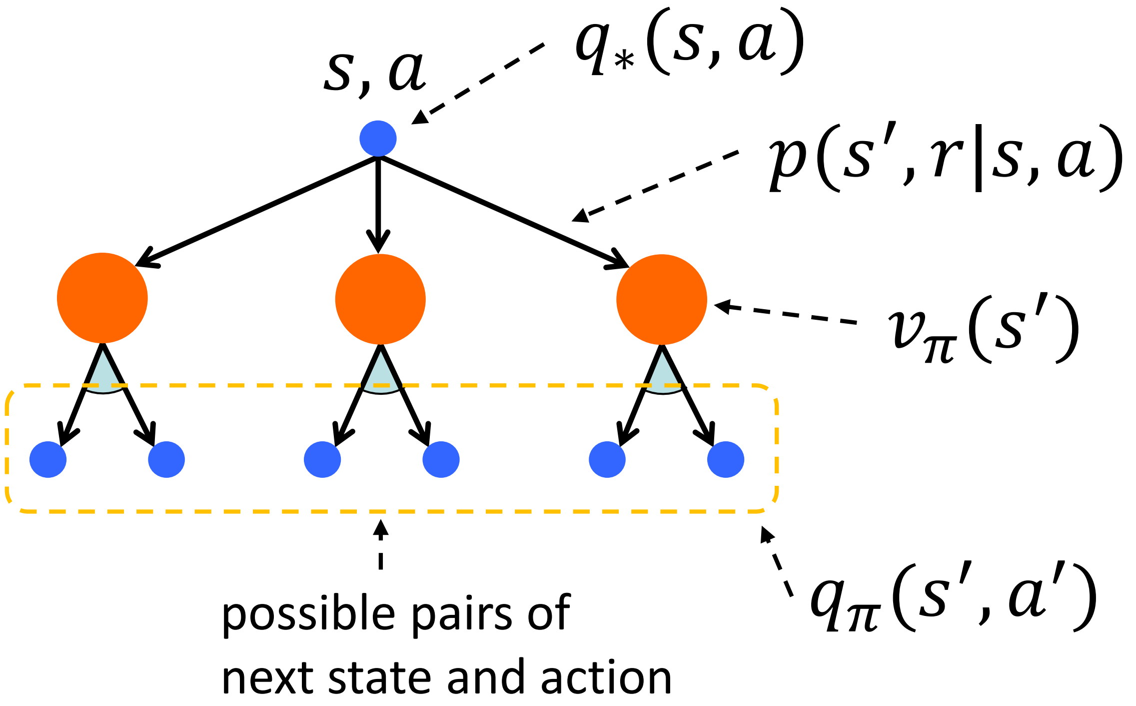
Bellman Optimality Equation for q_*

- Similarly, the Bellman optimality equation is given by

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

$$= \sum_{s',r} p(s',r \mid s,a)\left[r + \gamma \max_{a'} q_*(s',a')\right]$$

- This is also a nonlinear equation with respect to q_π due to the max operator



Optimal Value Functions

- There exists the following relationships:

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

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

Summary

- Bellman expectation equation

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s',r} P(s', r \mid s, a) \, [r + \gamma v_\pi(s')]$$

$$q_\pi(s,a) = \sum_{s',r} P(s', r \mid s, a) \left[r + \gamma \sum_{a'} \pi(\, a' \mid s' \,) q_\pi(s', a') \right]$$

- Bellman optimality equation

$$v_*(s) = \max_a \sum_{s',r} p(\, s', r \mid s, a \,)[r + \gamma v_*(s')]$$

$$q_*(s,a) = \sum_{s',r} p(\, s', r \mid s, a \,) \left[r + \gamma \max_{a'} q_*(s', a') \right]$$

Break!

References

- Asada, M., Noda, S., Tawaratsumida, S., and Hosoda, K. (1996). Purposive behavior acquisition for a real robot by vision-based reinforcement learning. Machine Learning, 23, 279-303, 1996.
- Playing Mario with Deep Reinforcement Learning. https://github.com/aleju/mario-ai
- Tanaka, S.C., Doya, K., Okada, G., Ueda, K., Okamoto, Y., and Yamawaki, S. Prediction of immediate and future rewards differentially recruits cortico-basal ganglia loops. Nature Neuroscience, 7(8): 887-893, 2004.