EE488B Special Topics in EE <Deep Learning and AlphaGo>

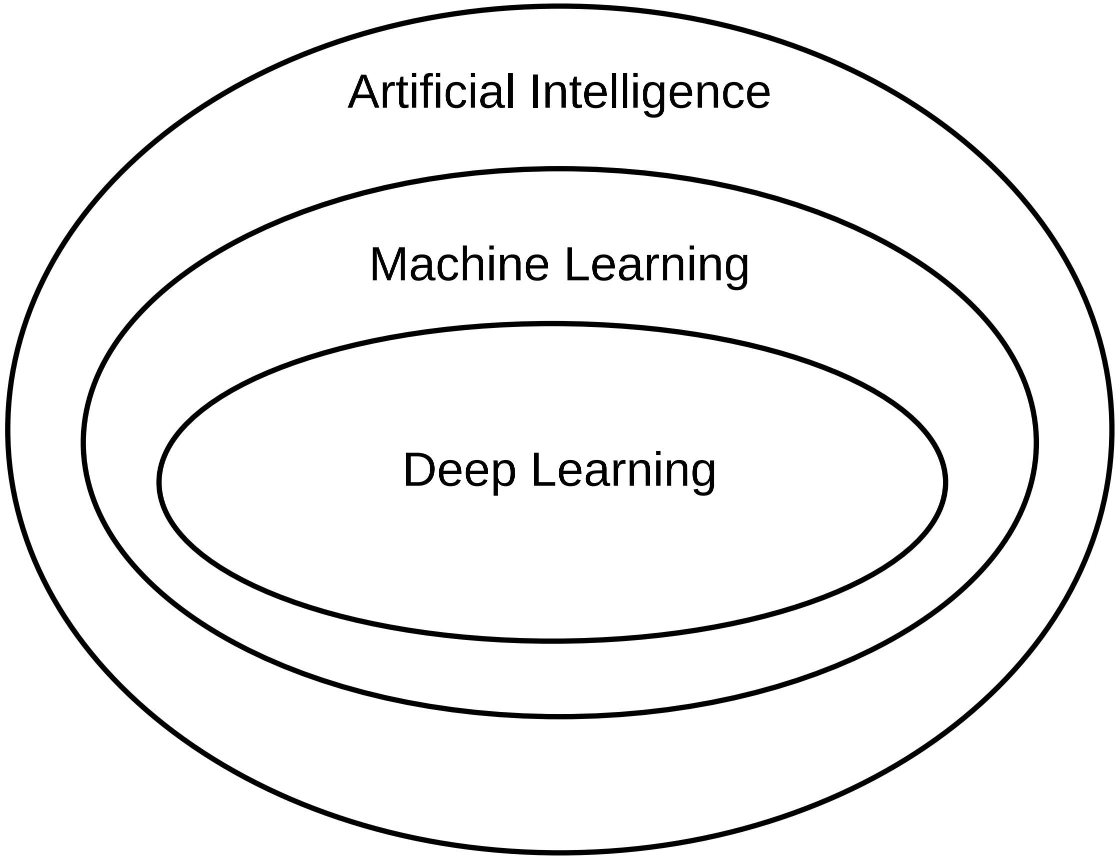
Sae-Young Chung

Lecture 10

October 30, 2017

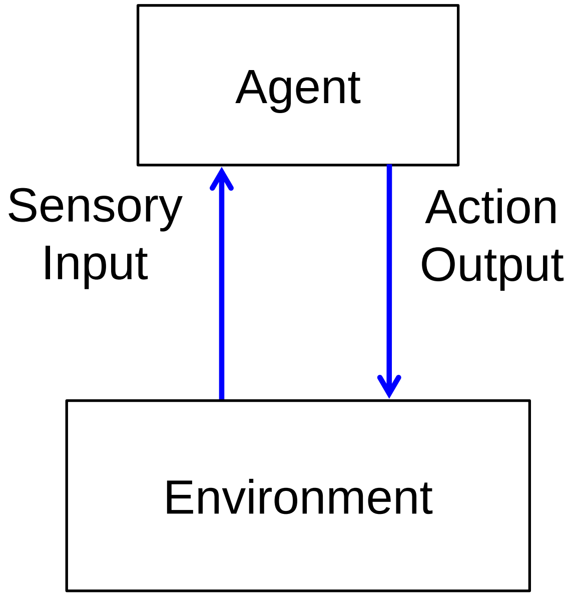
Artificial Intelligence

- Learning
- Logical reasoning and problem solving
- Understanding
- Planning
- Creativity
- Natural language processing (NLP)
- Motion
- Artistic intelligence
- Social intelligence
- Emotional intelligence
- Artificial general intelligence
- Self awareness


Artificial Intelligence
Machine Learning
Deep Learning

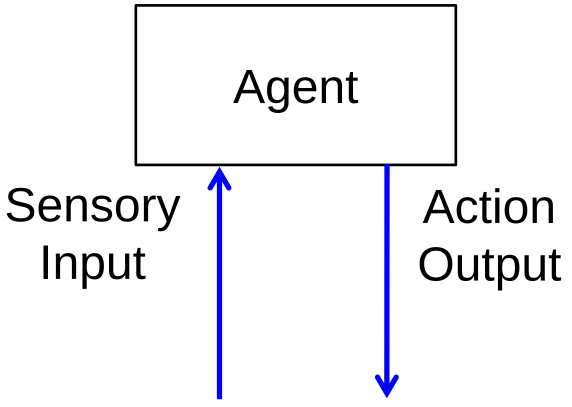
Reinforcement Learning

- Machine learning
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning
- Reinforcement learning
  - Tries to maximize reward
  - Learns which action to take when
  - Environment has memory (state): agent’s action affects future states


Agent
Sensory Input
Action Output
Environment

Non-interactive Task

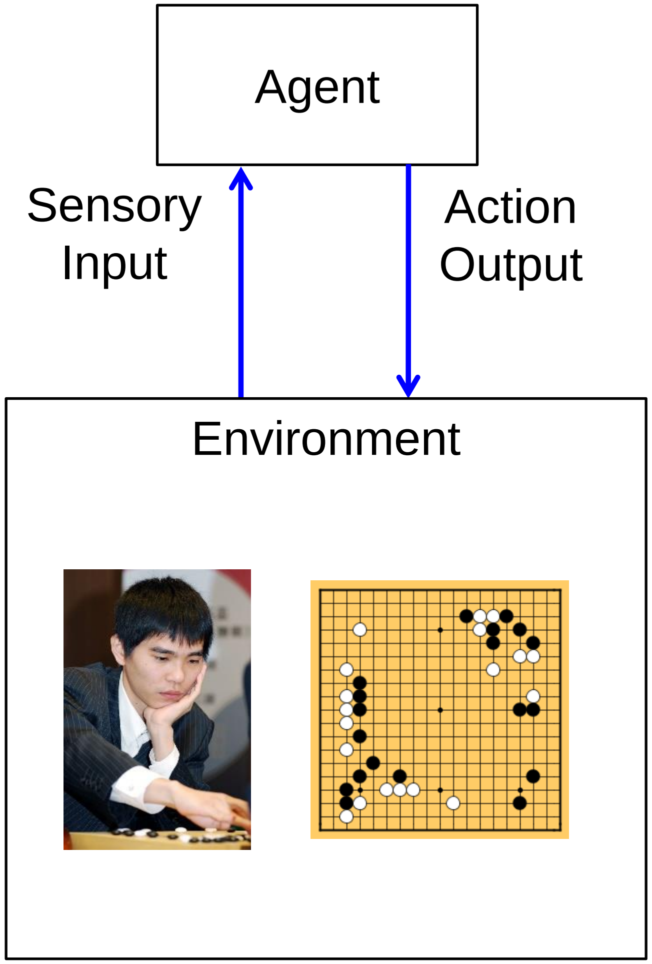
E.g., classification



cat
dog
…

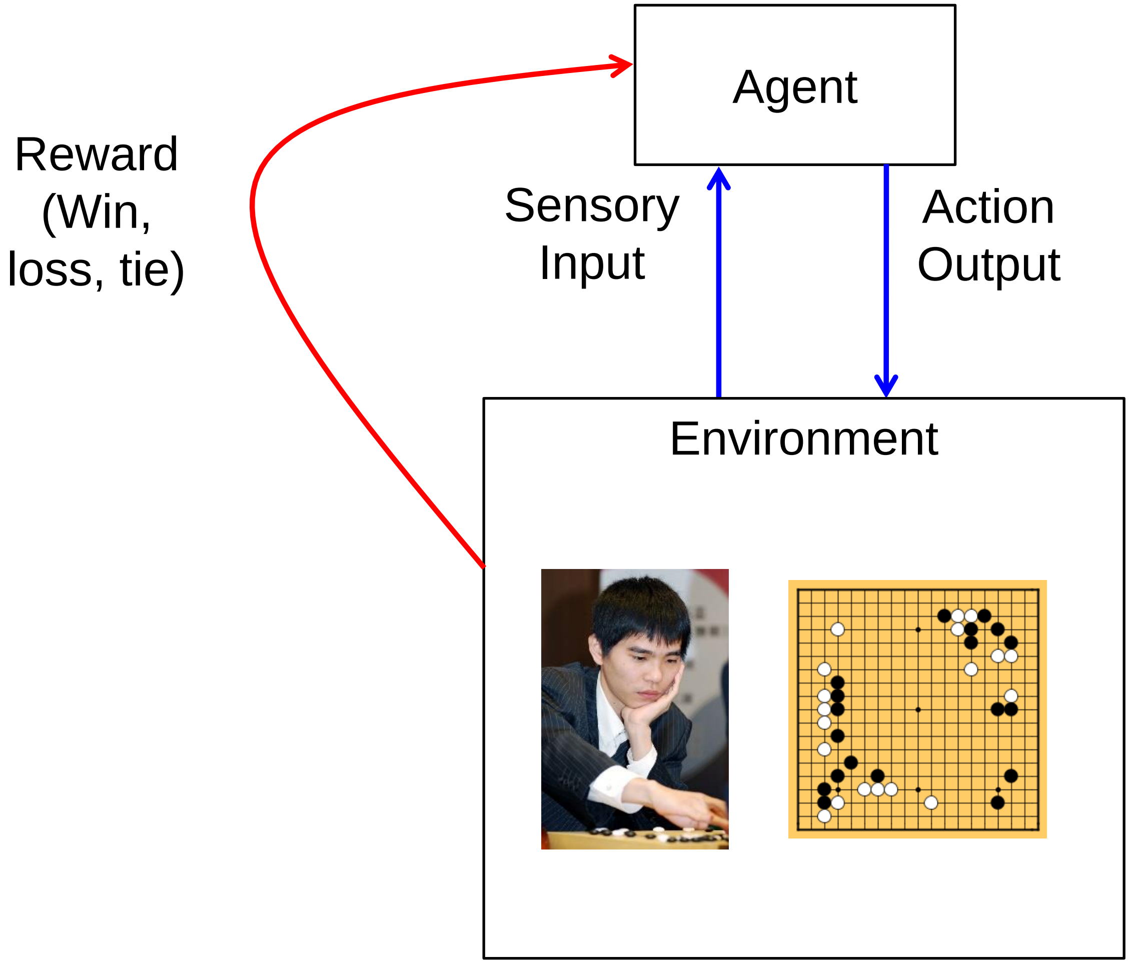
Classification output does NOT affect future inputs

Interactive Task



Agent’s action output affects future inputs. Learning through interactions with environment.

Reinforcement Learning

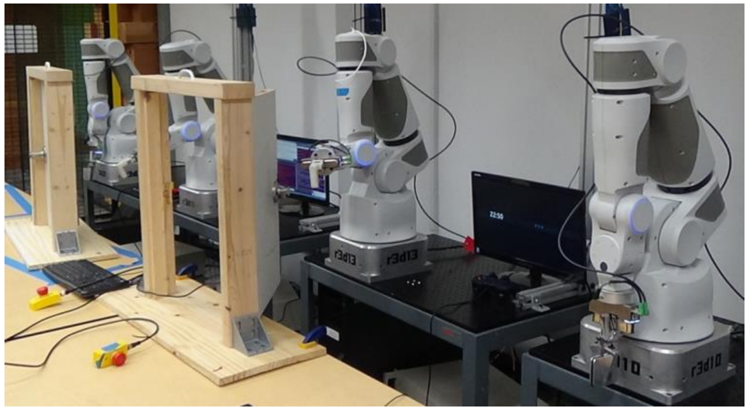


Reinforcement learning: Learning to take actions based on past input and reward to maximize expected future reward (e.g., winning probability)

Agent makes sequential decisions based on sensory input. Agent's action output may affect future inputs and/or future rewards. Learning needs to account for such interactions.

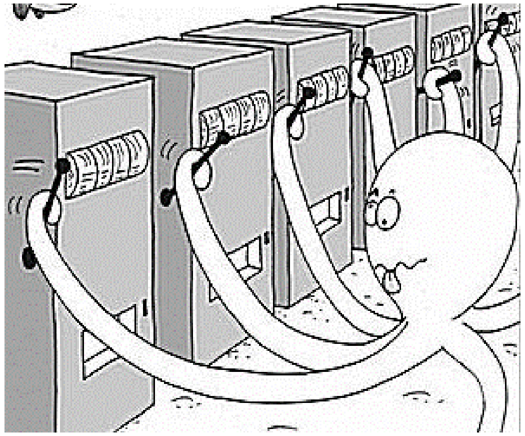
Examples

AlphaGo

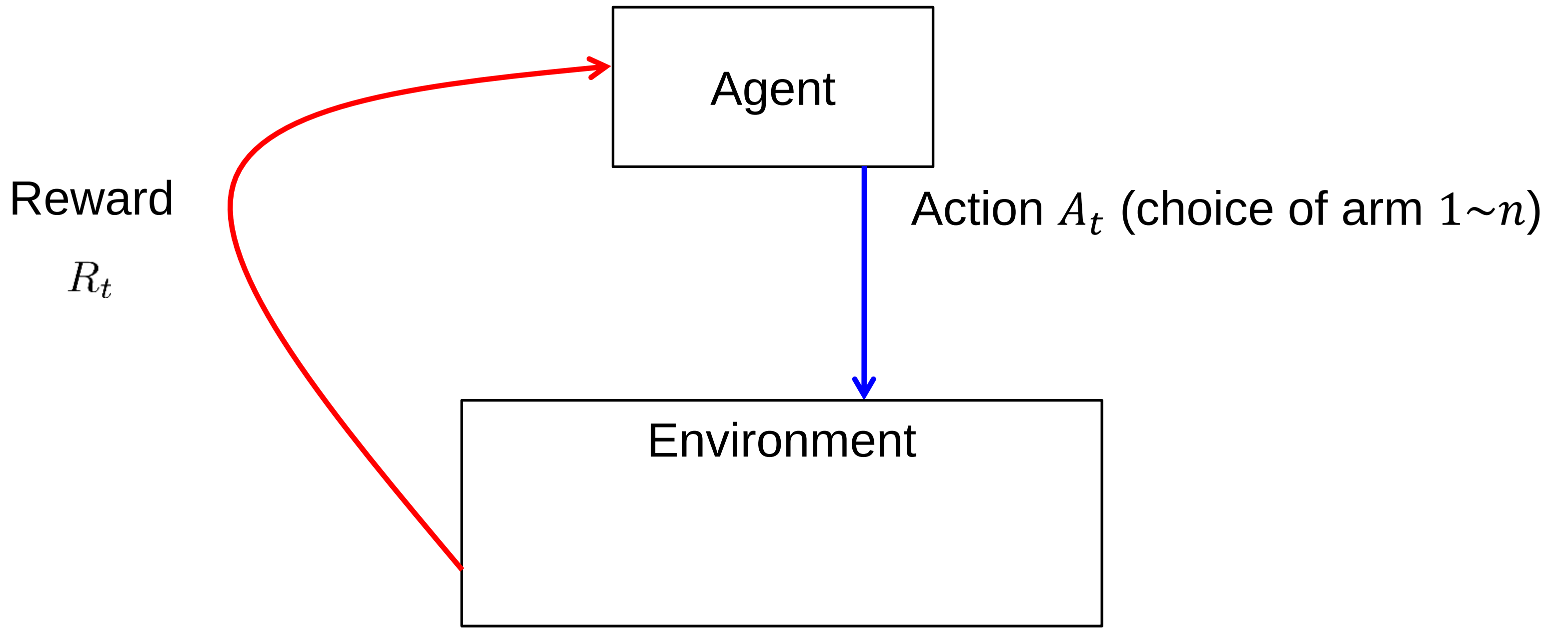


Examples

Multi-armed Bandit Problem



Multi-armed Bandit Problem



- Multi-armed (or n-armed) bandit problem
  - n: number of arms
  - $A_t \in \{1, \ldots, n\}$: action at time t
  - R_t: (random) reward at time t based on action A_t

Multi-armed Bandit Problem

- Value of action a: expected reward when action a is chosen, i.e.,

$$q_*(a) := \mathbb{E}[R_t | A_t = a]$$

- However, $q_*(a)$ is unknown to the agent and therefore it must estimate it from past experience.
  - $Q_t(a)$: estimated value of action a at time t
  - Example) sample average of past rewards for which the action was $A_t = a$

$$Q_t(a) = \frac{1}{n_{a,t}} \sum_{i=1}^{n_{a,t}} R_{t_{a,i}}$$

    - $t_{a,i}$: time index of the i-th occurrence of action a
    - $n_{a,t}$: number of times action a occurred before time t
    - If $n_{a,t} = 0$, then take $Q_t(a) = Q_0$, e.g., $Q_0 = 0$

- Greedy action: action that maximizes the estimated value (there may be more than one)

- ϵ-greedy method: take a random action with probability ϵ (exploration) and a greedy action with probability $1 - \epsilon$ (exploitation)

- ϵ can be either fixed or gradually reduced

ϵ-greedy Method

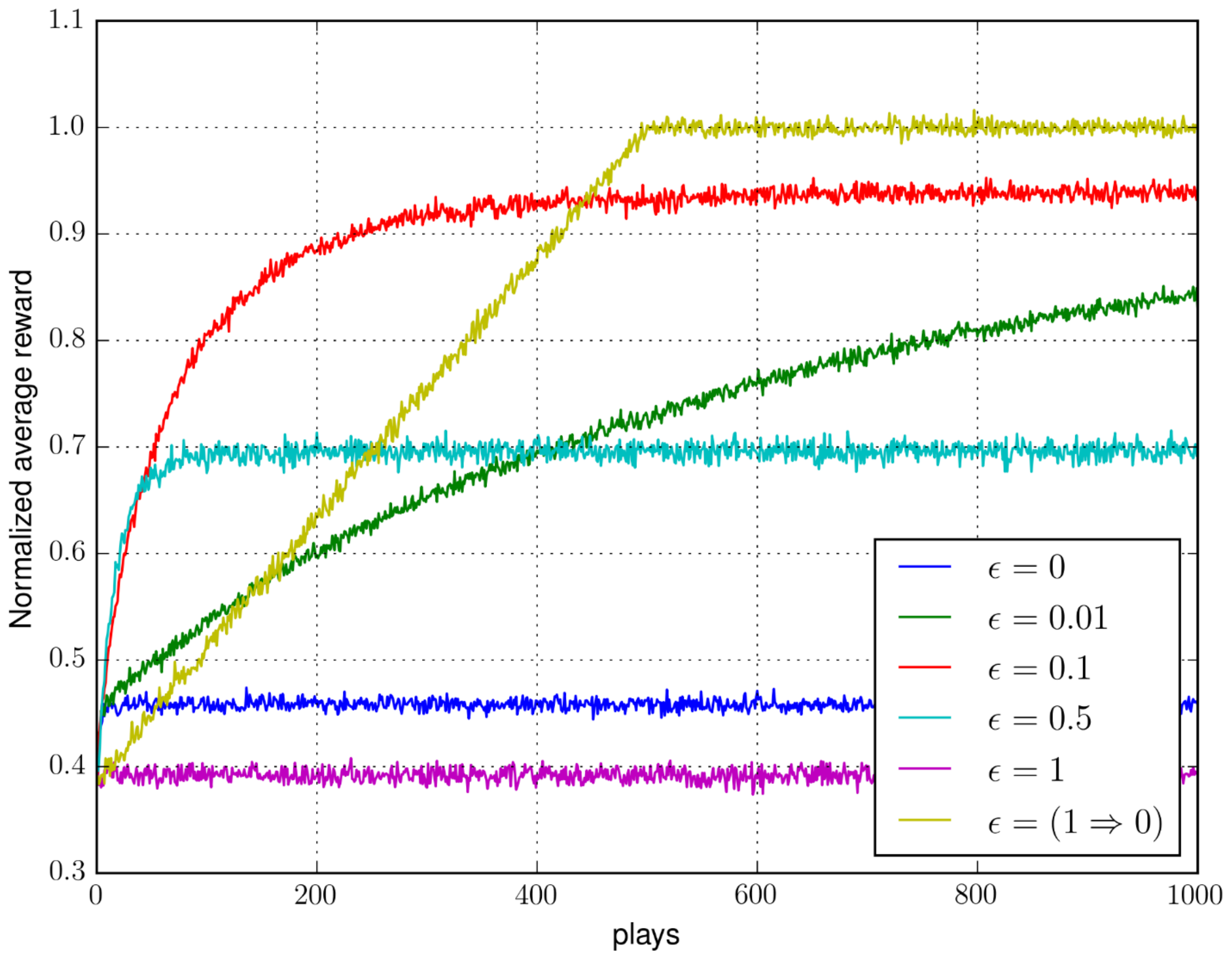
- $\epsilon = 0$: pure exploitation
- $\epsilon = 1$: pure exploration
- Tradeoff between exploitation & exploration

Classics
Fresh Brewed Coffee
Award winning coffee ground daily and brewed fresh.
Americano
Dark roast espresso with hot water; European style coffee.
Steve Smith Tea
Proudly serving Artisan whole-leaf Steven Smith Teamoker teas.
Chai or Mate Latte
Tea and spices served steaming hot or over ice. Chai is available Sugar-Free.
Espresso Shot
Our signature dark roast blend combines dark chocolate with a hardy body and rich smoky accents.
Flavor Infused Tea
Exceptional Black or Fez iced tea steeped fresh with bold and balanced flavors.
Little Beans
Italian Soda or Creamosa
Soda water and sweet flavored syrups; add cream for a Creamosa.
Real Fruit Smoothies
Blended real fruit with no preservatives or artificial flavors.
Not So Hot Chocolate
Rich and famous hot chocolate topped with whipped cream.
Flavored Steamer
Steamed milk flavored with premium syrups.

Example

- Example
  - $n = 10$
  - Reward for the k-th arm: $\mathcal{N}(\mu_k, 0.3^2)$
  - μ_k's: i.i.d. uniform in $[0, 1]$, generated once
  - Number of plays: $1,000$
  - Averaged over 5,000 trials (1,000 plays in each trial)
  - `narmedbandit.py`

ϵ-greedy Method



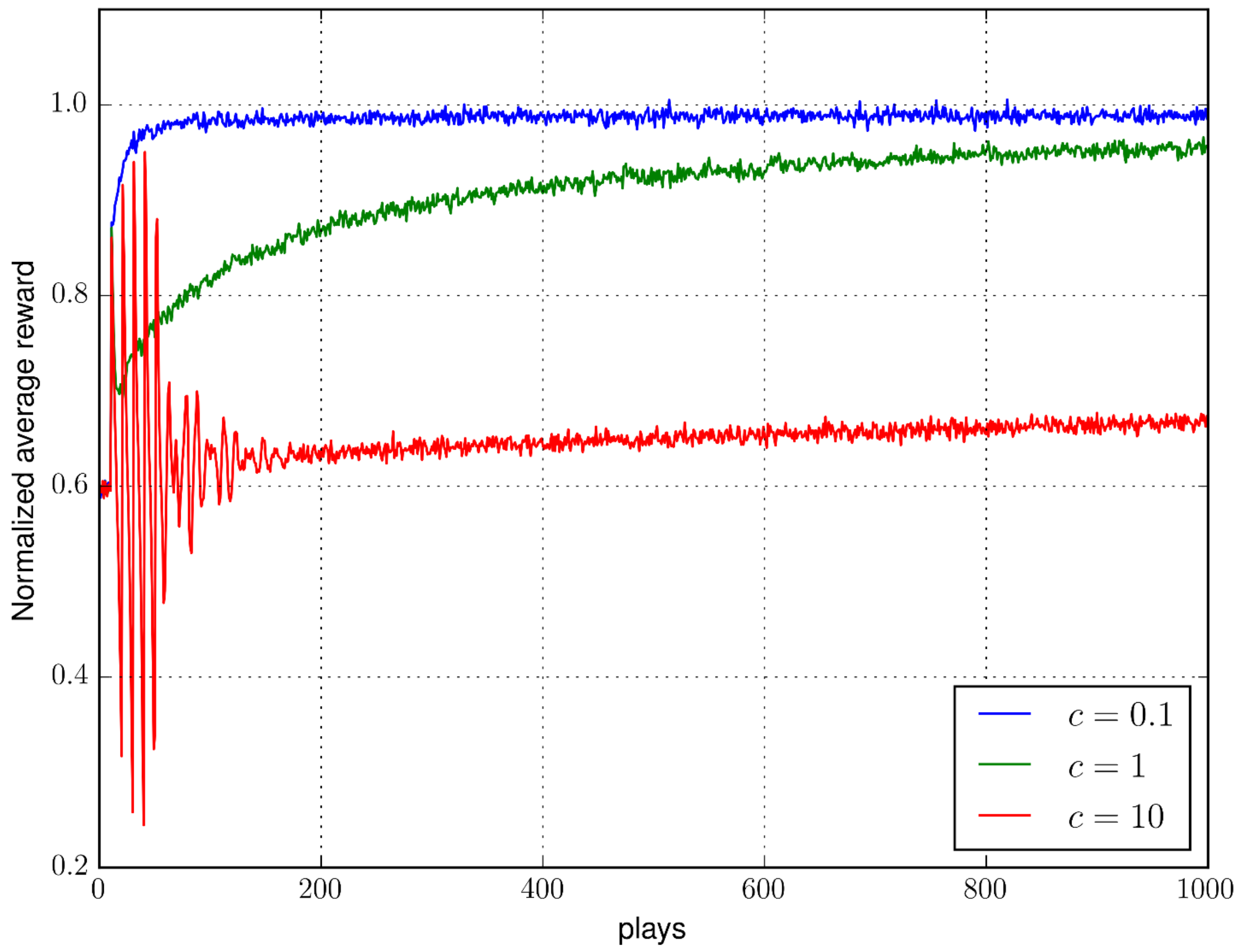
UCB Action Selection

- Upper-Confidence-Bound (UCB) Action Selection

$$A_t := \underset{a}{\operatorname{argmax}} \left[Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}} \right]$$

  - $c > 0$: parameter to control the degree of exploration
  - $N_t(a)$: # of times action a was taken before time t
  - $\sqrt{\frac{\log t}{N_t(a)}}$: measure of uncertainty or variation in the estimate of a's value
    - Higher if less visited
    - Lower if more visited
    - Thus, enables more exploration when less visited
  - $Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}}$: roughly an upperbound on the possible true value of a

UCB Action Selection

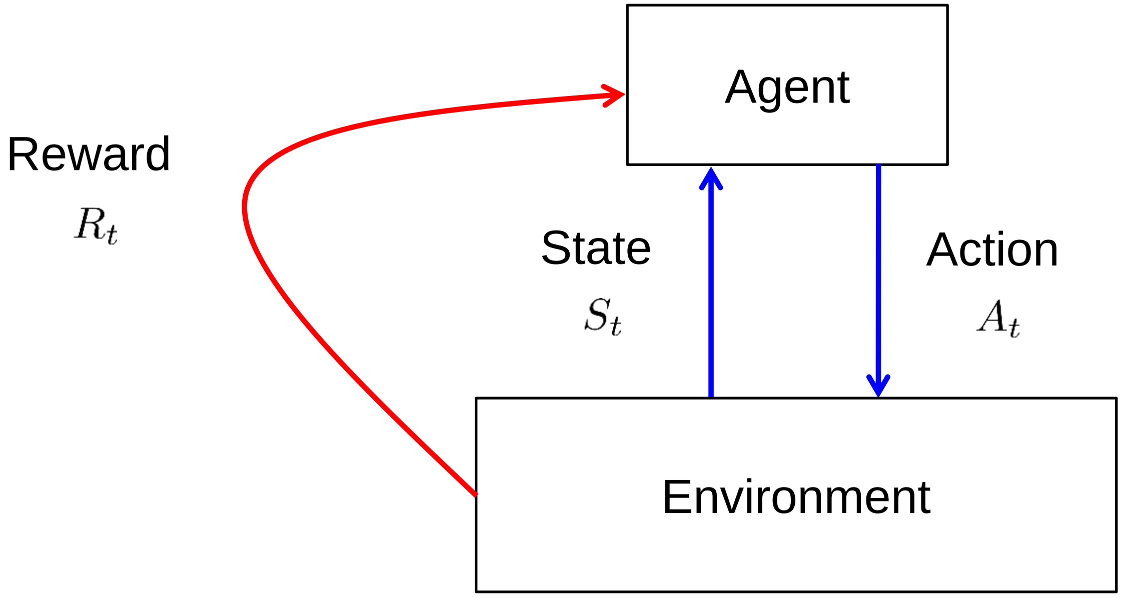


Reinforcement Learning Setup

- General environment defined by the conditional probability of the next state and reward given all history

$$\Pr(S_{t+1}=s', R_{t+1}=r | S_t=s, A_t=a, R_t=r_t, S_{t-1}=s_{t-1}, A_{t-1}=a_{t-1}, \ldots, R_1=r_1, S_0=s_0, A_0=a_0)$$

  - $S_t \in \mathcal{S}$: state at time t
  - $A_t \in \mathcal{A}(s)$: action at time t based on the current state $S_t = s$
  - $R_t \in \mathbb{R}$: reward at time t

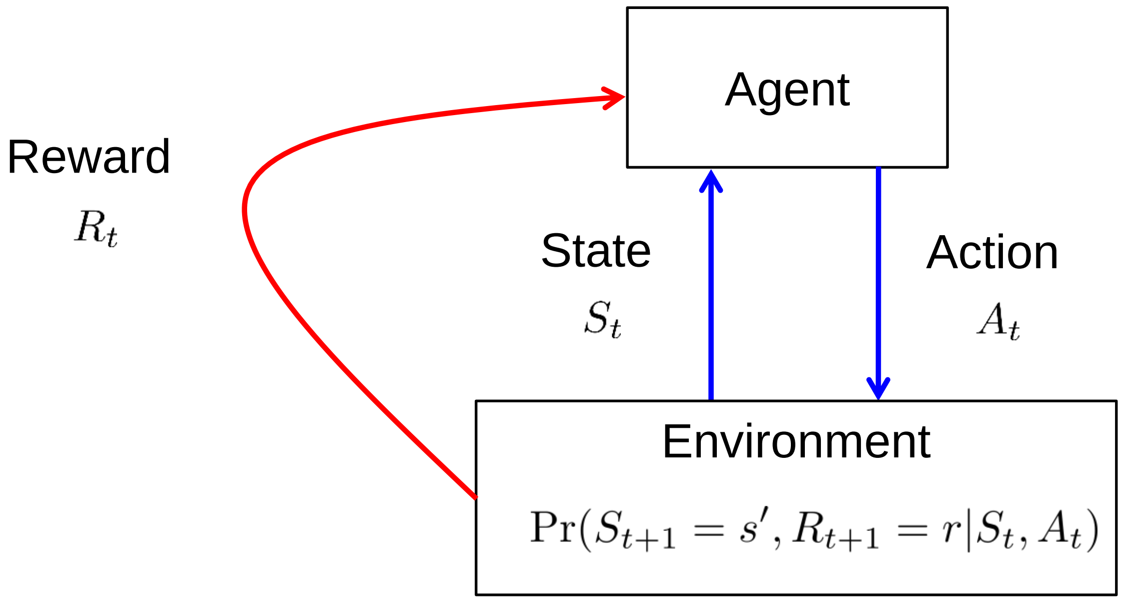


Markov Decision Process

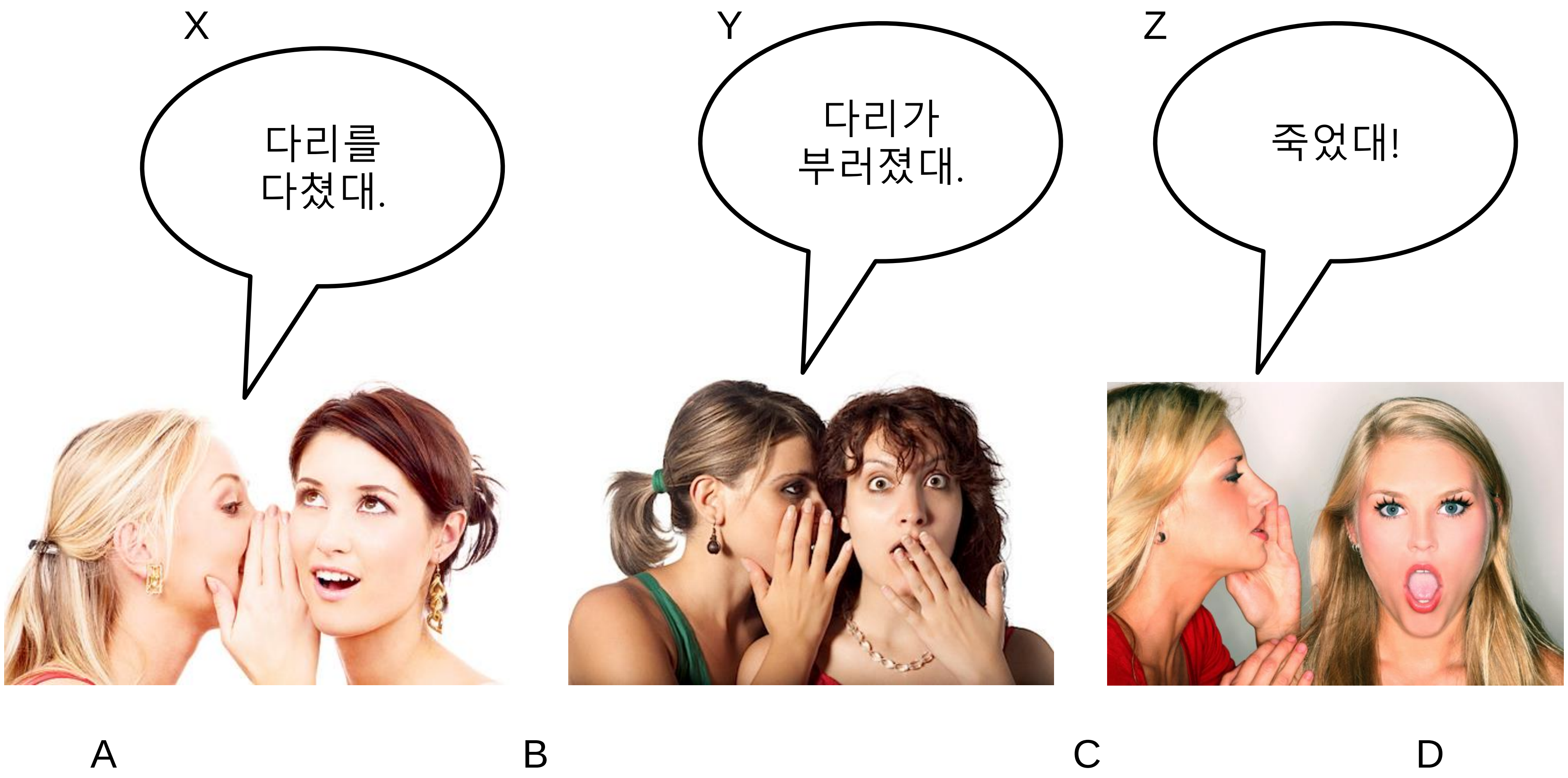
- Markov decision process (MDP): RL task satisfying the following Markov property

$$\Pr(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a, R_t = r_t, S_{t-1} = s_{t-1}, \ldots, S_0 = s_0, A_0 = a_0)$$
$$= \Pr(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a)$$

for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$, $s' \in \mathcal{S}$, $r \in \mathbb{R}$, $r_t \in \mathbb{R}$, $s_{t-1} \in \mathcal{S}, \ldots, s_0 \in \mathcal{S}, a_0 \in \mathcal{A}(s_0)$



Markov Chain Example



Markov Chain Example

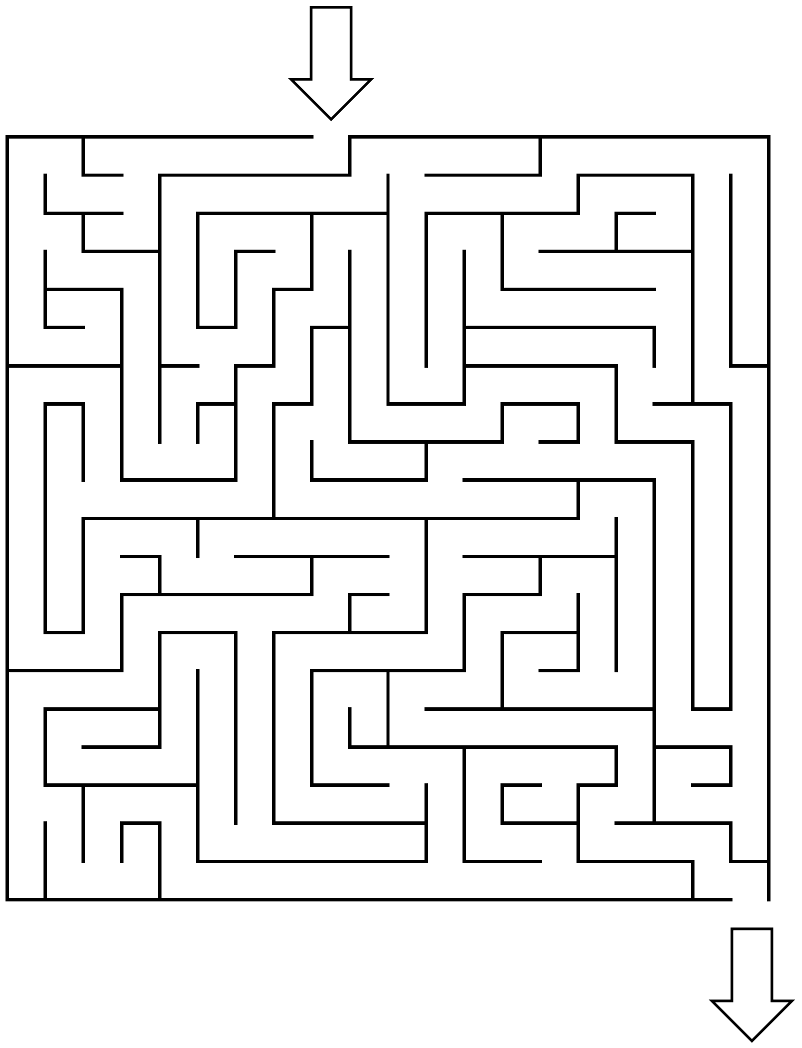
- In the example in the previous page, let
  - X: The sentence A tells B, i.e., “He hurt his legs.”
  - Y: The sentence B tells C, i.e., “He broke his legs.”
  - Z: The sentence C tells D, i.e., “He died.”
- If Y is randomly changed from X by person B and Z is randomly changed from Y by person C who does not have access to X, then $X - Y - Z$ form a Markov chain in that order.
- Def) $X - Y - Z$ form a Markov chain in that order if

$$\Pr(Z = z | X = x, Y = y) = \Pr(Z = z | Y = y) \quad \text{for all } x, y, z$$

Markov Decision Process

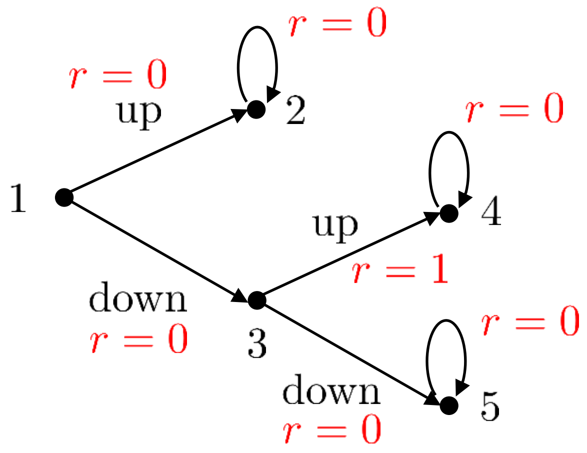
- Example of MDP
  - n-armed bandit problem: $\mathcal{S} = \emptyset$, $\mathcal{A} = \{1, 2, \ldots, n\}$, $\Pr(R_{t+1} = r | A_t = a) = p_a(r)$
  - $N \times N$ 2D maze game: $\mathcal{S} = \{1, \ldots, N\}^2$, $\mathcal{A} = \{\mathrm{L}, \mathrm{R}, \mathrm{U}, \mathrm{D}\}$
- Example of non-MDP
  - Go: if the opponent's next move is affected by past actions of itself

MDP Example – Maze Game



MDP Example

- Example)
  - $\mathcal{S} = \{1, 2, 3, 4, 5\}$
  - $\mathcal{A}(s) = \{\text{up}, \text{down}\}$ if $s = 1, 3$
  - State 1 is the starting state
  - States 2,4,5 are terminal (absorbing) states
  - Reward is 1 if $s = 3$ and $s' = 4$, otherwise reward is 0

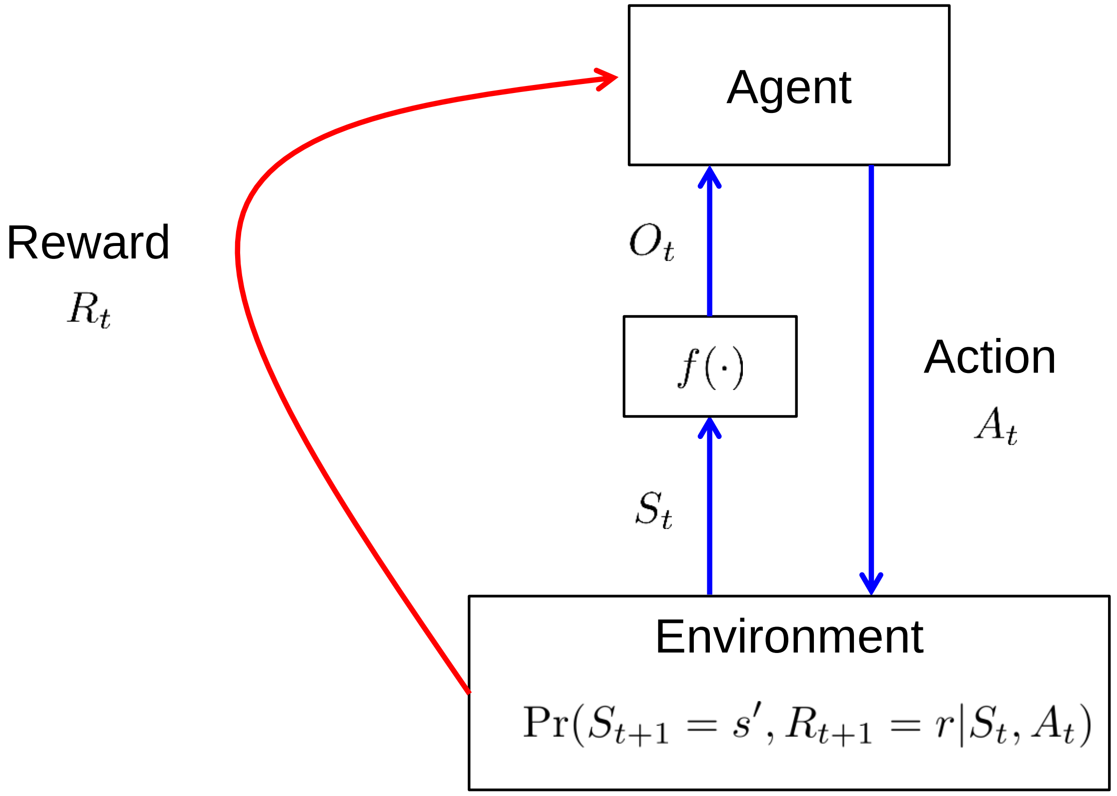


$$p(s', r|s, a) = \begin{cases} 1 & \text{if } s' = 2,\ r = 0,\ s = 1,\ a = \text{up} \\ 1 & \text{if } s' = 3,\ r = 0,\ s = 1,\ a = \text{down} \\ 1 & \text{if } s' = 4,\ r = 1,\ s = 3,\ a = \text{up} \\ 1 & \text{if } s' = 5,\ r = 0,\ s = 3,\ a = \text{down} \\ 1 & \text{if } r = 0 \text{ and } s' = s = 2, 4, \text{ or } 5 \\ 0 & \text{otherwise} \end{cases}$$

POMDP

- Partially Observable MDP (POMDP): state is only partially observable, e.g.

$$O_t = f(S_t),\ \Pr(O_t|S_t, A_{t-1}), \ldots$$

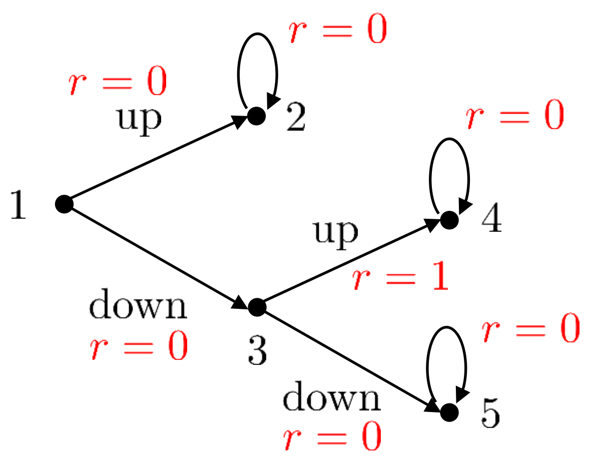


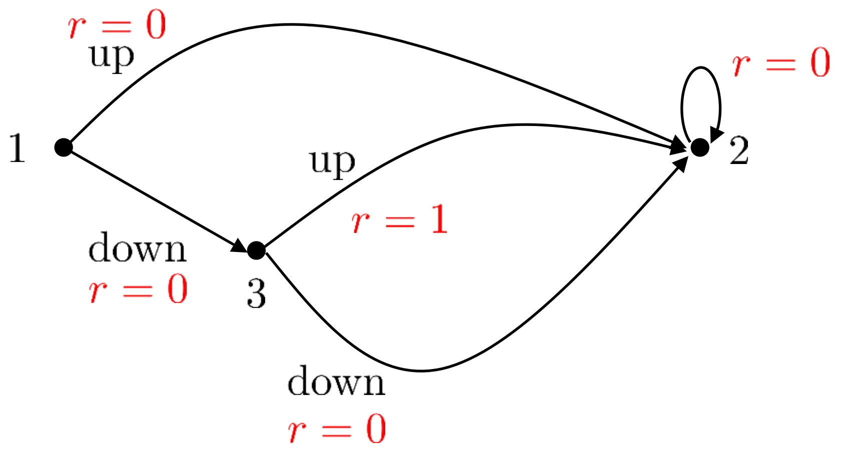
Types of Tasks

- Episodic task: Terminates in a special state called the terminal state
  - Example) Go, Maze game
- Continuing task: Does not terminate
  - Example) Robot performing tasks without terminating

Absorbing State

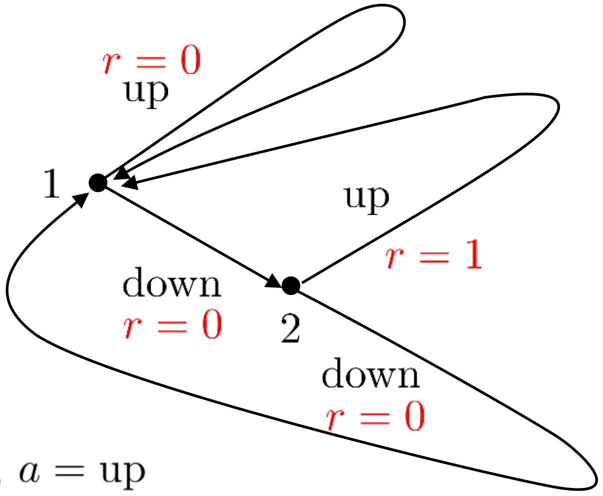
- Absorbing state is introduced for unified treatment of episodic and continuing tasks, i.e., episodic task can be treated as a continuing task
- Multiple or single absorbing states





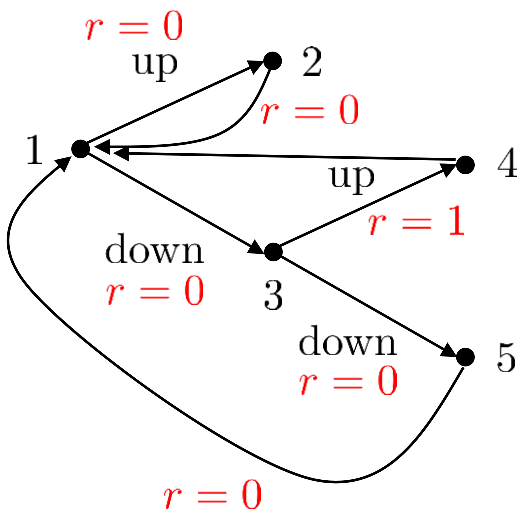
Repeated Tasks

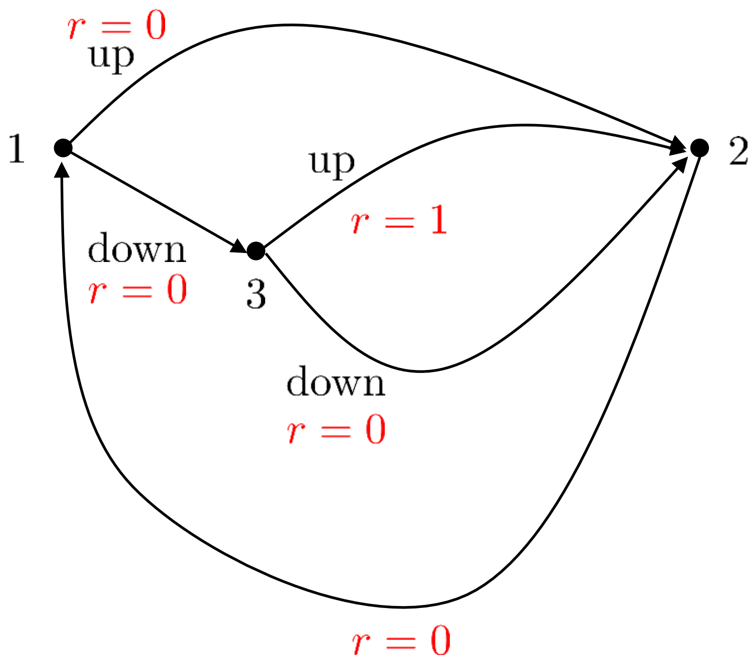
- Some tasks are repeated, i.e., we go back to the initial state when reached the end of task.
- Example)
  - $\mathcal{S} = \{1, 2\}$
  - $\mathcal{A} = \{\text{up}, \text{down}\}$
  - State 1 is the starting state
  - Reward is 1 if $s = 2$ and $s' = 1$, otherwise reward is 0



$$p(s', r|s, a) = \begin{cases} 1 & \text{if } s' = 1,\ r = 0,\ s = 1,\ a = \text{up} \\ 1 & \text{if } s' = 2,\ r = 0,\ s = 1,\ a = \text{down} \\ 1 & \text{if } s' = 1,\ r = 1,\ s = 2,\ a = \text{up} \\ 1 & \text{if } s' = 1,\ r = 0,\ s = 2,\ a = \text{down} \\ 0 & \text{otherwise} \end{cases}$$

Other Examples





Policy & Discounted Return

- Policy: function π that specifies probability of each action that the agent chooses given the current state

$$\pi_t(a|s) := \Pr(a_t = a|s_t = s)$$

- Goal of reinforcement learning: finding a policy maximizing expected future reward
- Which future reward?
- We often use exponentially discounted return

$$G_t := R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

  - $0 \le \gamma \le 1$: discount rate
  - $\gamma = 0$: agent is myopic, i.e., tries to maximize immediate reward only
  - $\gamma \to 1$: agent becomes farsighted, i.e., takes future rewards into account more strongly

Value Functions

- State-value function for policy π: value of state s under policy π

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

- Action-value function for policy π: value of taking action a at state s under policy π

$$q_\pi(s,a) := \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\middle| S_t = s, A_t = a\right]$$

- They exist if $\gamma < 1$ or termination is always reached eventually.
- Value functions need to be estimated from experience

Bellman Equation

- Bellman equation for v_π (consistency condition for v_π)

$$\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\middle| S_t = s\right] \\
&= \mathbb{E}_\pi\left[R_{t+1} + \sum_{k=1}^{\infty}\gamma^k R_{t+k+1}\middle| S_t = s\right] \\
&= \mathbb{E}_\pi\left[R_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k R_{t+k+2}\middle| S_t = s\right] \\
&= \sum_a \pi(a|s)\sum_{s',r} p(s',r|s,a)\left[r + \gamma\mathbb{E}_\pi\left(\sum_{k=0}^{\infty}\gamma^k R_{t+k+2}\middle| S_{t+1} = s'\right)\right] \\
&= \sum_a \pi(a|s)\sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right],
\end{aligned}$$

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

Optimal Value Functions

- Optimal policy π_*: $v_{\pi_*}(s) \geq v_\pi(s)$ for all $s \in \mathcal{S}$ and all π
- Optimal state-value function

$$v_*(s) := \max_\pi v_\pi(s)$$

- Optimal action-value function

$$\begin{aligned}
q_*(s,a) &:= \max_\pi q_\pi(s,a) \\
&= \max_\pi \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \\
&= \max_\pi \mathbb{E}_\pi\left[\sum_{k=0}^\infty \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right] \\
&= \max_\pi \mathbb{E}_\pi\left[R_{t+1} + \gamma \sum_{k=0}^\infty \gamma^k R_{t+k+2} \middle| S_t = s, A_t = a\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a\right]
\end{aligned}$$

Bellman Optimality Equation

- Bellman optimality equation for v_*

$$\begin{aligned}
v_*(s) &= \max_{a\in\mathcal{A}(s)} q_{\pi_*}(s,a) = \max_{a\in\mathcal{A}(s)} \mathbb{E}_{\pi_*}[G_t | S_t = s, A_t = a] \\
&= \max_{a\in\mathcal{A}(s)} \mathbb{E}_{\pi_*}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right] \\
&= \max_{a\in\mathcal{A}(s)} \mathbb{E}_{\pi_*}\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \middle| S_t = s, A_t = a\right] \\
&= \max_{a\in\mathcal{A}(s)} \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) \middle| S_t = s, A_t = a\right] \\
&= \max_{a\in\mathcal{A}(s)} \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_*(s')\right]
\end{aligned}$$

- Has a unique solution for finite MDP.
- Once we have $v_*(s)$ by solving the above, an optimal policy can be obtained by assigning any non-zero probabilities only to the actions that maximize $\sum_{s',r} p(s',r|s,a)\left[r + \gamma v_*(s')\right]$.
- But, obtaining an optimal policy this way becomes too expensive computationally as problem becomes more complex, e.g., it would be impossible to solve for the game of Go.

Assignments, etc.

- Reading assignment: Chapters 3 ~ 6 of RL book
  - "Reinforcement learning: An introduction" by R. Sutton and A. Barto
    - Draft available online at http://incompleteideas.net/sutton/book/the-book-2nd.html
    - Most recent version is dated June 19, 2017
- Project #1
  - Deep learning using TensorFlow
  - Due: November 8
- No class on November 29 due to KAIST entrance exam