Reinforcement Learning Problem
Week #3

Reinforcement Learning Loop

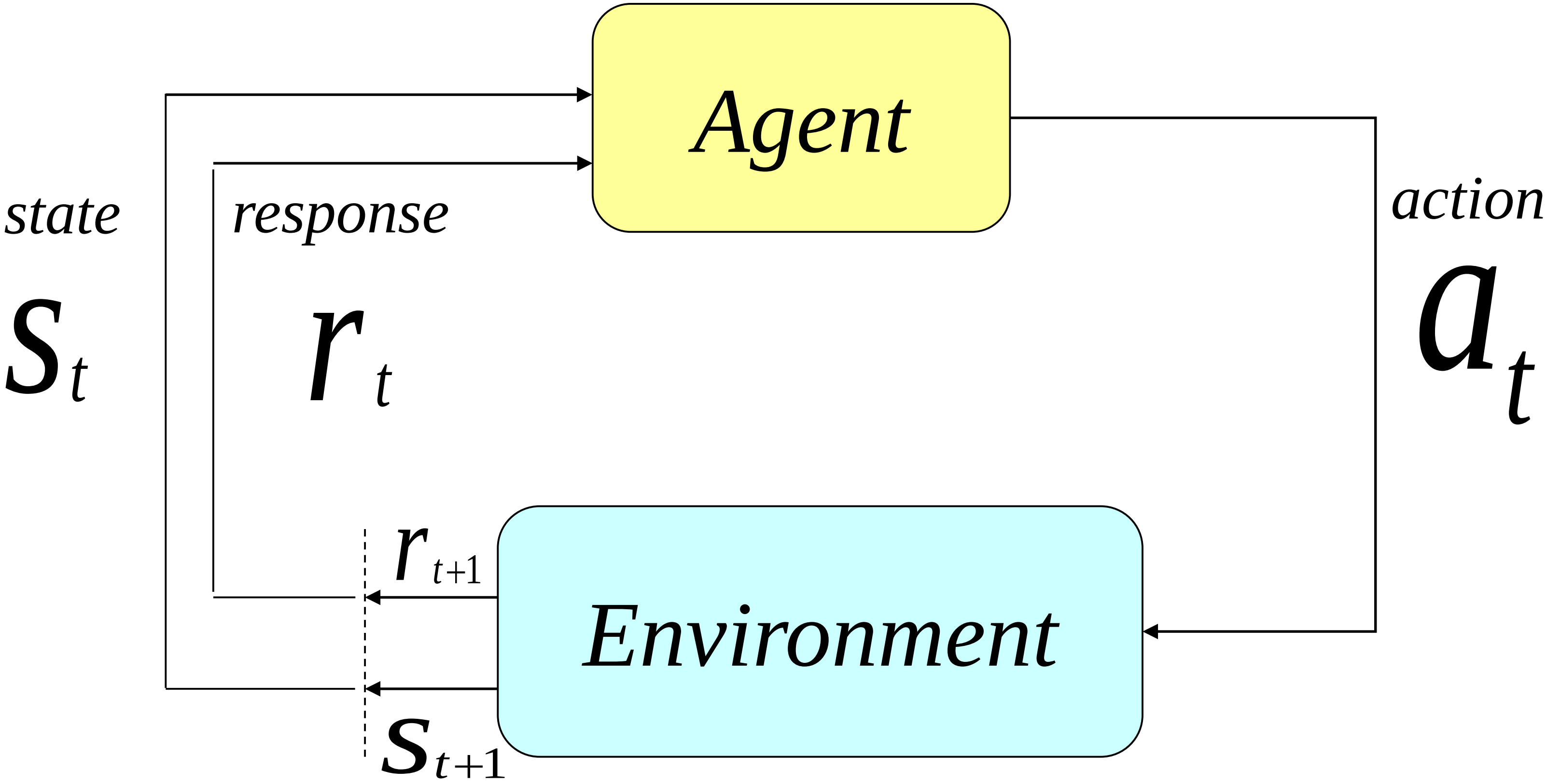


Figure reproduced from the figure
on page 52 in reference [1]

Agent – Environment Interaction

- The figure on slide 2 represents the *interaction between agent and environment*.
- Agent takes an action a_t at discrete time t.
- Perceiving the action, environment communicates two *signals* with agent at time $t+1$:
  - *reward*, $r_{t+1} \in R$
  - *some representation of environment's state*, s_{t+1};

Time Domain

- Time is *discrete*, may be *extended to continuous time* [2].
- Time steps over which reinforcement learning occurs may not need be *equally spaced*. Time steps may actually be *event-based*.
- That is, each time step is a *successive stage of some decision-making process*.

Reward: Response of Environment

- the *response of environment* to the *last action taken by agent.*
- a *numerical signal* r_t, represents the extent to which *environment* found agent's selection (i.e, current action) *favorably* or *unfavorably*.
- may represent positive or negative response either as a *crisp* (binary) measure $\{0,1\}$ or as a *finite set of values* within a certain interval $\{a,\ldots,b\}$ or as a *continuous function* (infinitely many values) with a certain range $[a,b]$.

Assignment of Reward

- *Reward* is used in RL problems to make agent *move directly toward its goal*.
- This is one of the *features that distinguishes RL from other learning schemes*.
- For agent to learn the *goal-directed behavior*, the *rewards* should be provided *only to actions that really do lead to the goal*.
- There may be *subgoals* that might help agent reach its goal, but, does not regularly result in primary goal.
- *Assigning rewards to these subgoals, that do not necessarily end up in the main goal, may cause agent to learn to maximize its long-run reward reaching these subgoals and not its main goal*.

Representation of Environment's State

- *A* *state* $s_t \in S$ is the situation/position the environment is at, at time *t*. Depending on the RL task, a *state* manifests the location of environment within the domain.
- Examples may range from
  - *low-level readings* such as the "angular position of the rotor in an electric motor" to
  - *high-level modes of environment* such as some "tendency of a society to feel pessimistic, optimistic, lost or happy..."

Representation of Environment's State

- Given the previous examples of state representation, a wide spectrum of methods exist to represent the states of environment: numerically in a single or multiple dimensional space, using strings, sets of tuples, etc.
- ***In general, any information that contributes to agent's decision-making can be considered to be a state.***
- Most of the time the state information is not clearly perceived by the agent. The state information is noisy or imperfect. Hence, instead of a clear state information, agent perceives some representation of state or an *observation implying some state*.

Action: Decision of Agent

- *Action* is an attempt that agent selects to better understand/control environment.
- By selecting an action, agent attempts to anticipate/predict the behavior of environment and fine-tune its decisions to maximize a cumulative criterion.
- "***An action is any decision that agent wants to learn how to make***" (pp. 53 [1]).

Separation of Environment from Agent

- The rule for distinguishing environment from agent: ***Anything under absolute control of agent is considered as part of agent***. Anything not under the absolute control of agent (i.e., ***cannot be arbitrarily changed by the agent***) is considered to be outside of it and hence ***part of the environment***.

RL Framework

- A considerable abstraction of the problem of goal-directed learning from interaction.
- It asserts that
  - Any problem of goal-directed behavior can be reduced to three signals communicated between an agent and its environment: the representation
    - of choices made by the agent (*actions*)
    - of the basis on which the choices are made (*states*)
    - that define the agent's goal (*rewards*)

Goals & Rewards

- *Goals* are what we want the agent to achieve.

- *Rewards* are used to define the goal or to direct the agent to achieve its goal in such a way that the *goal of the agent becomes to maximize its reward in the long run*.

Goals & Rewards ...(2)

- *Examples*
  - while having a mouse agent learn to find the cheese through a maze, it may be given a reward +1 when it finds the cheese and 0 until then.
  - to make it learn to find the cheese asap, for each unit of time it may be negatively rewarded (with -1 for instance).
  - Tic-tac-toe playing agent may be rewarded *only* when it wins.
  - Discussion: how to reward a chess playing agent?

Returns

- ***Return*** is the *long-term (accumulated) reward*.
- Agent seeks to maximize the *expected* return.
- The return may be simply defined as the sum of individual returns.

$$R_t = \sum_{k=0}^{T} r_{t+k+1}$$

- An alternative definition of return where the ***most recent reward has the highest effect*** on the expected return is the ***discounted return***:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \quad 0 \le \gamma < 1$$

Markov Property

- State is *any information of environment available to agent*.
- State provides *current and past information* on environment.
- State ***does not and should not be expected to provide every detail of knowledge on environment***.

Markov Property... (2)

- This implies that, due to the nature of the task, most of the time, some information may be unavailable (i.e., ***hidden***) to agent, which may or may not be valuable in solving the learning problem.
- Therefore, each environment state s_t may be represented at agent's side by whatever observation o_t is perceived by the agent.

Markov Property... (3)

Hence, responses to agent should be provided rather on the basis of whether or not *agent has forgotten some valuable information that it once knew* than whether or not agent knows it.

Markov Property

- The *state information* should ideally *summarize past inputs while preserving relevant information*.

- A *state* with that property is said to ***be Markov*** or to ***have Markov property***

Markov state

- A ***general environment*** whose policy is based upon all past state information has a state probability distribution as in the following:

$$\Pr\{s_{t+1} = s^{i}, r_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, \ldots, r_1, s_0, a_0\}$$

- For a *first degree* ***Markovian environment***, this conditional state probability distribution depends only upon the current state and action:

$$\Pr\{s_{t+1} = s^{i}, r_{t+1} = r \mid s_t, a_t\}$$

Markov Decision Processes (MDPs)

- An RL task that satisfies the Markovian property is defined as a *Markov decision process (MDP)*. An MDP with finite state and action sets is called a finite MDP.
- ***Transition probability:***

$$P^a_{s^i s^j} = \Pr\{s_{t+1} = s^j \mid s_t = s^i, a_t = a\}$$

- ***Expected next reward:***

$$R^a_{s^i s^j} = E\{r_{t+1} \big| s_t = s^i, a_t = a, s_{t+1} = s^j\}$$

- These two quantities completely specify the dynamics of a finite MDP.

Value Functions

- Value functions are functions of states ($V^\pi(s)$) or of state-action pairs ($Q^\pi(s,a)$) that define a basis for agent's decision about "how good" it is to be at a state *s* or to select an action *a* at a specific state *s*, respectively, for a specific policy $\boldsymbol{\pi}$.
- ***State-value function for policy π:***

$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s\right\}$$

- ***Action-value function for policy π :***

$$Q^{\pi}(s,a) = E_{\pi}\{R_t \mid s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a\right\}$$

Recursive Relationships of Value Functions

- Value functions fulfill for a specific state the following recursive property:

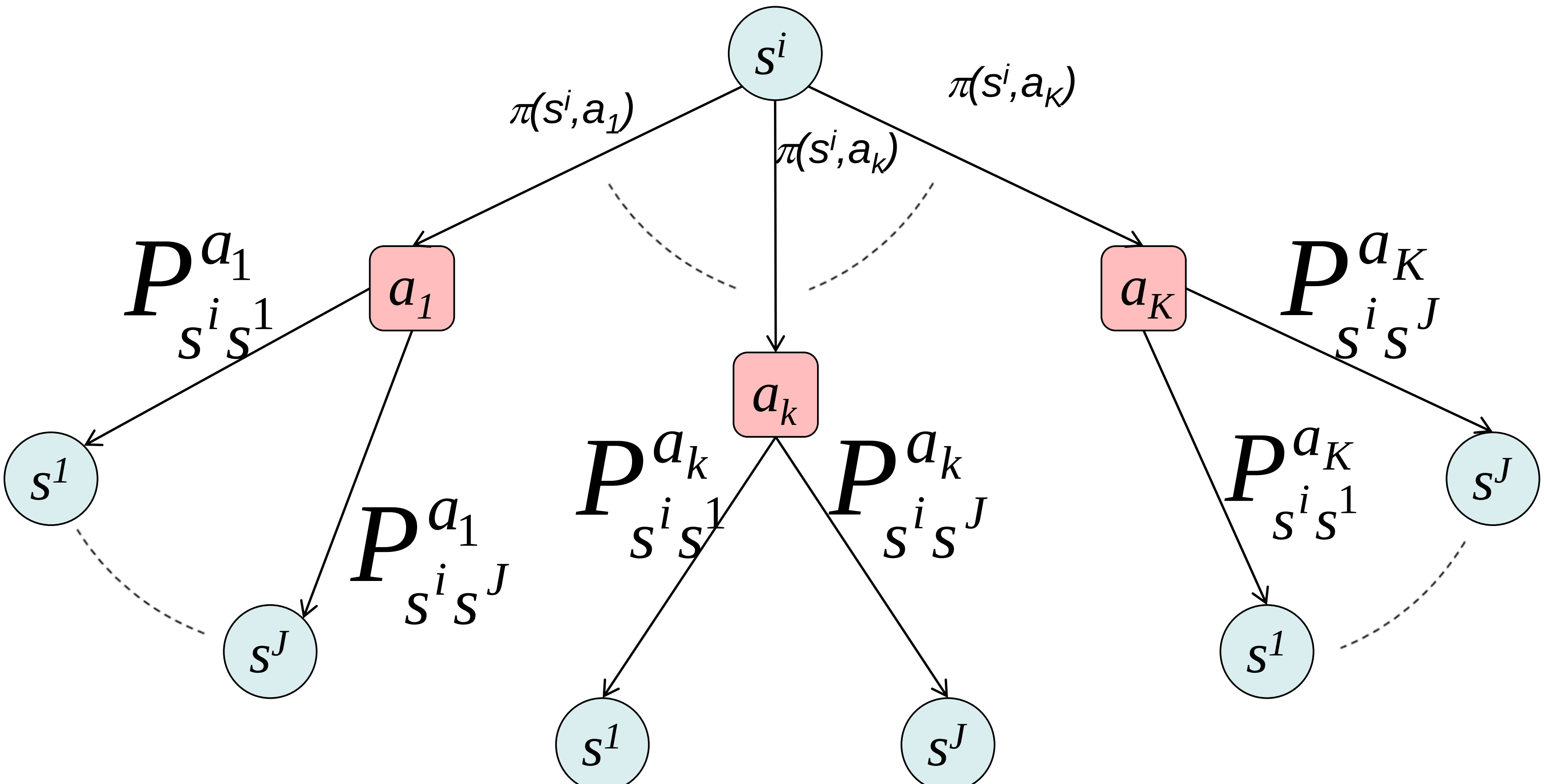
$$V^{\pi}(s^i) = E_{\pi}\{R_t \mid s_t = s^i\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s^i \right\} =$$

$$= E_{\pi}\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_t = s^i \right\} =$$

$$= \sum_a \pi(s^i, a) \sum_j P^a_{s^i s^j} \left[R^a_{s^i s^j} + \gamma E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_{t+1} = s^j \right\} \right] =$$

$$= \sum_a \pi(s^i, a) \sum_j P^a_{s^i s^j} \left[R^a_{s^i s^j} + \gamma V^{\pi}(s^j) \right].$$

Backup Diagram of the Recursive Value Functions



$$\pi(s) \leftarrow \arg\max_a \sum_j P^a_{s^i s^j} \left\{ R^a_{s^i s^j} + \gamma V^\pi(s^j) \right\}$$

Better and Optimal Value Functions

- A policy π_i is defined to be ***better*** than or equal to a policy π_j ($\pi_i \geq \pi_j$) if

$$V^{\pi_i}(s) \geq V^{\pi_j}(s) \quad for\, all\; s \in S.$$

- The policy π^* is the ***optimal policy*** if

$$V^*(s) = \max_{\pi} V^{\pi}(s) \quad for\, all\; s \in S.$$

or

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \quad for\, all\; s \in S \quad and \quad for\, all\; a \in A.$$

References

- [1] Sutton, R. S. and Barto A. G., "*Reinforcement Learning: An introduction*," MIT Press, 1998
- [2] Bertsekas, D. P. and Tsitsiklis, J. N., "*Neuro-Dynamic Programming*," Athena Scientific, 1996