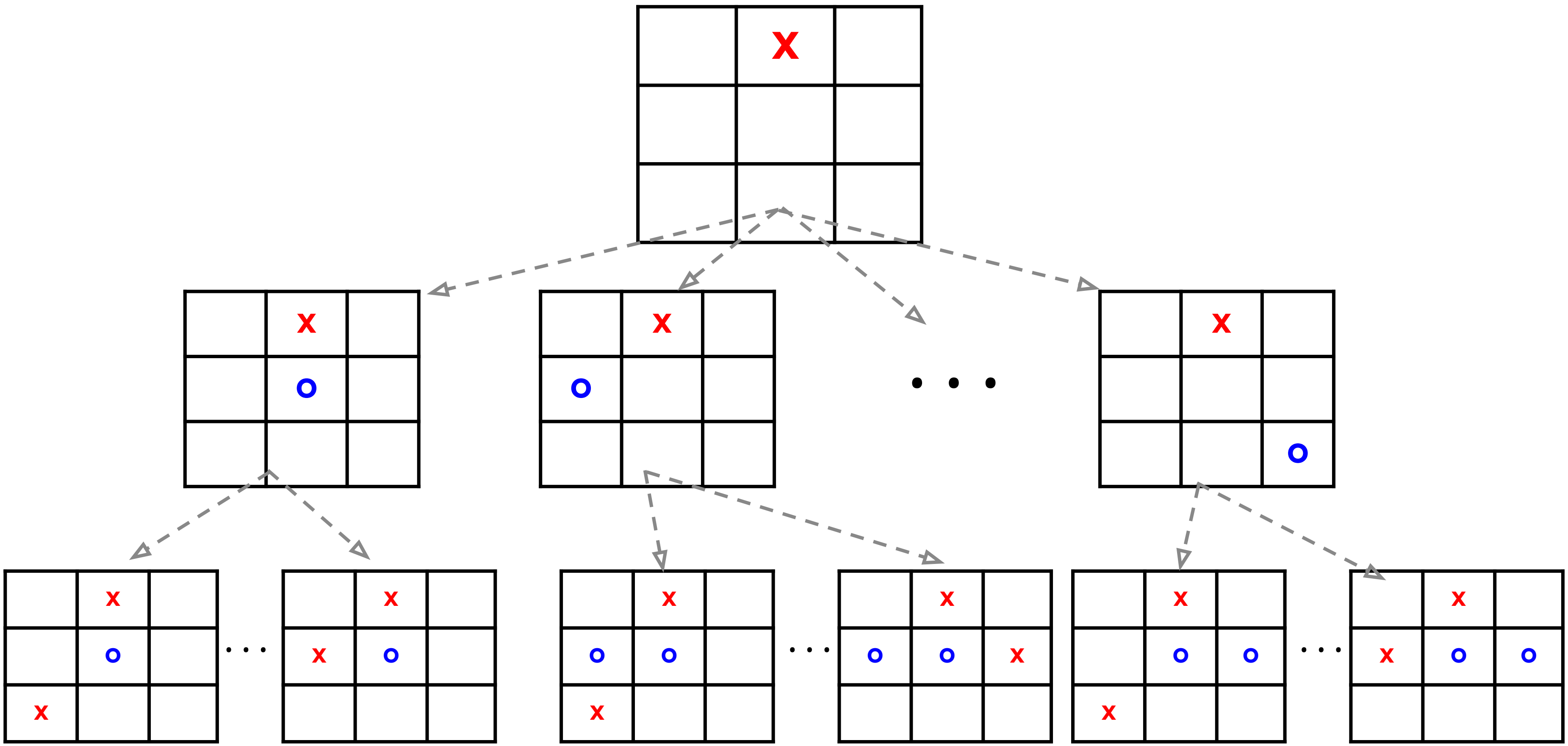
Lecture 3:
MDPs, Returns, Value functions, Q-function

B. Ravindran

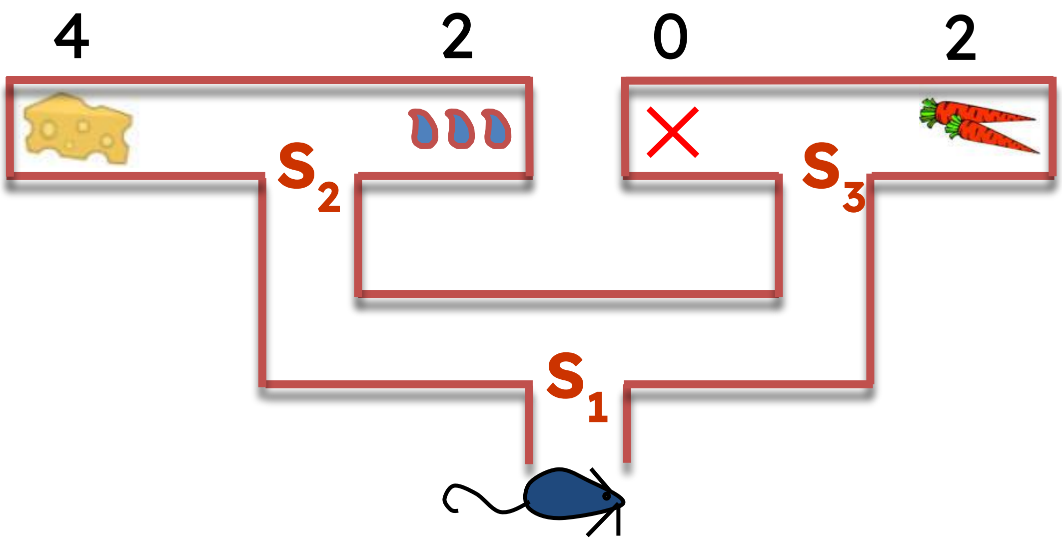
Immediate Reinforcement

- The payoff accrues immediately after an action is chosen
- One key question - the dilemma between exploration and exploitation
- Bandit problems encapsulates '*Explore vs Exploit*'

What about Tic-Tac-Toe?

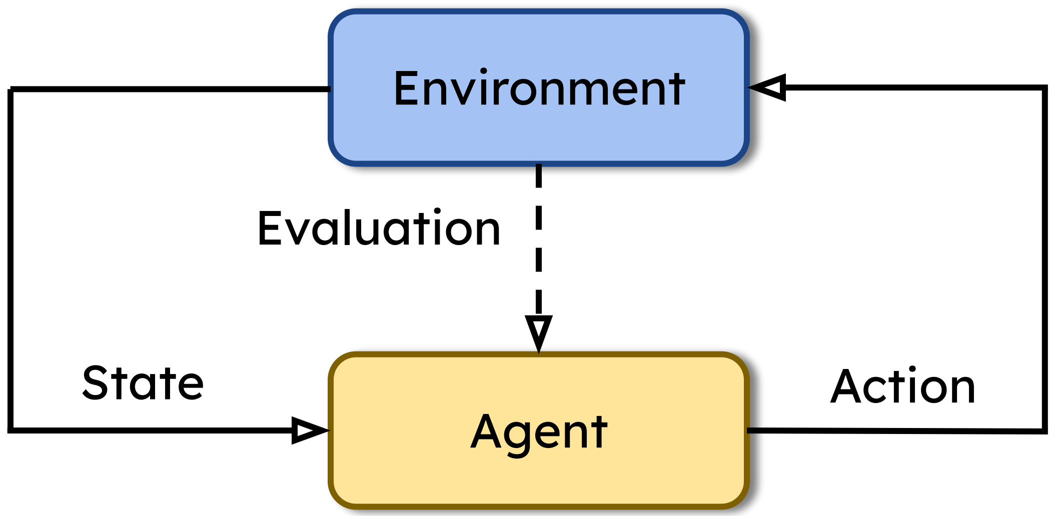


Action at a (Temporal) Distance



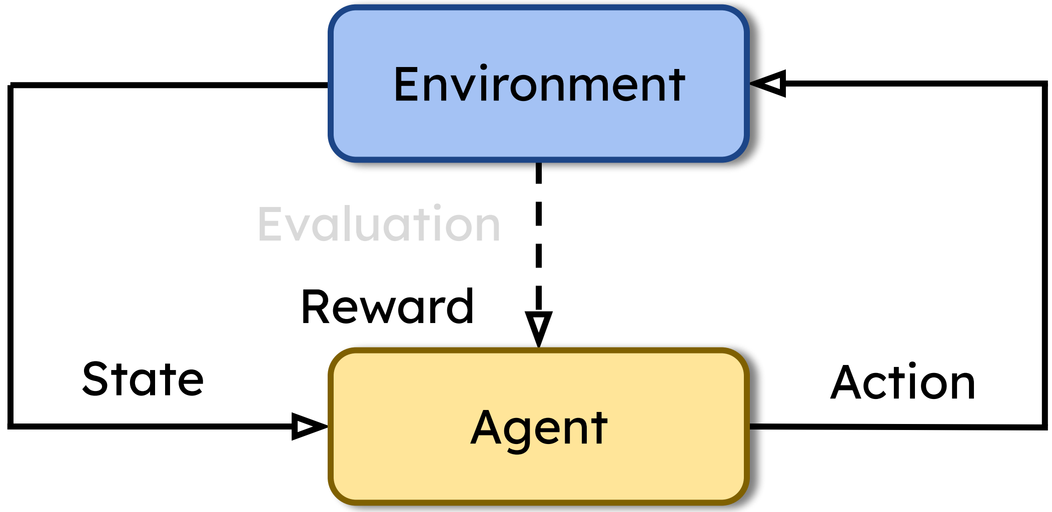
- Learning an appropriate action at S_1:
  - depends on the actions at S_2 and S_3
  - gains no immediate feedback

Full RL Framework



- Learn from close interaction
  - with a stochastic environment
  - having noisy <u>delayed</u> scalar evaluation
  - with a goal to maximize a measure of long term performance

Full RL Framework



- ❑ Learn from close interaction
  - ❑ with a stochastic environment
  - ❑ having noisy <u>delayed</u> scalar evaluation
  - ❑ with a goal to maximize a measure of long term performance

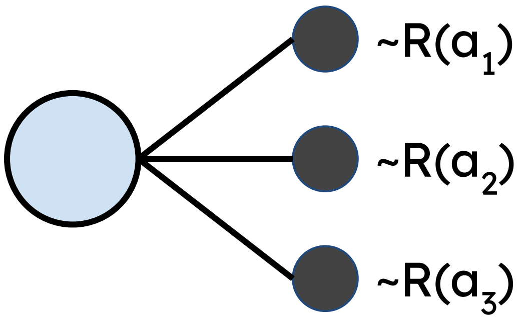
Designing an RL solution

- States
  - Enough information to take decisions
  - Raw inputs often not sufficient

- Actions
  - The control variables
  - Discrete – items to recommend, moves in a game
  - Continuous – torque to a motor, rate of mixing

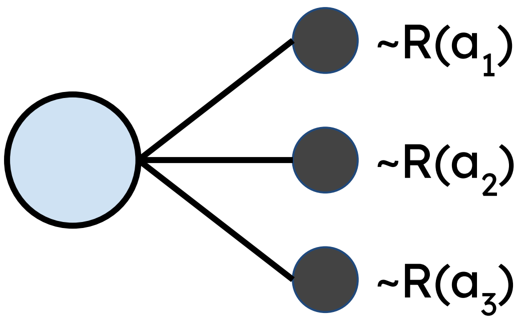
- Rewards
  - Define the *goal* of the problem

Full RL Problem



Bandit Problem

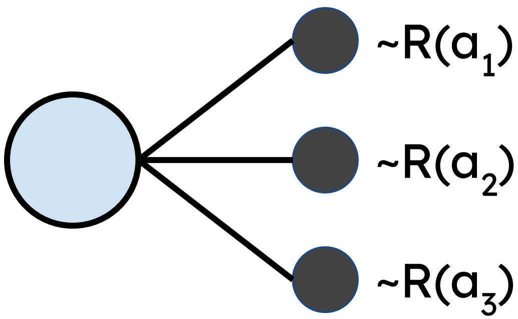
Full RL Problem



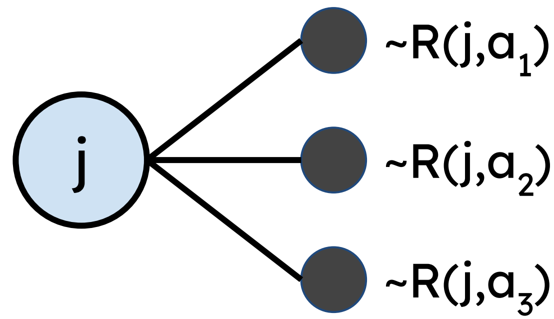
Bandit Problem

$$Q(a^*) = max_i \; \{Q(a_i)\}$$

Full RL Problem



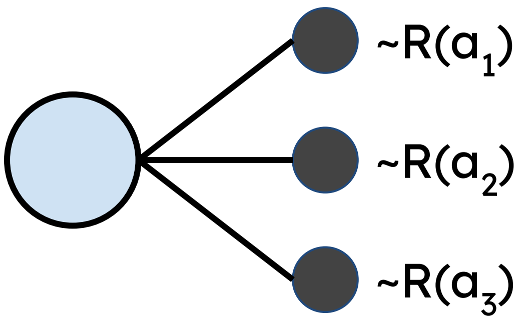
Bandit Problem



Contextual Bandit Problem

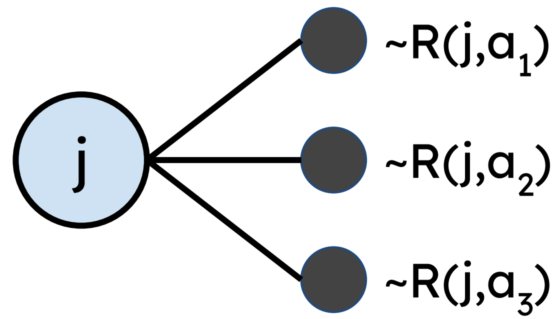
$$Q(a^*) = max_i \; \{Q(a_i)\}$$

Full RL Problem



Bandit Problem

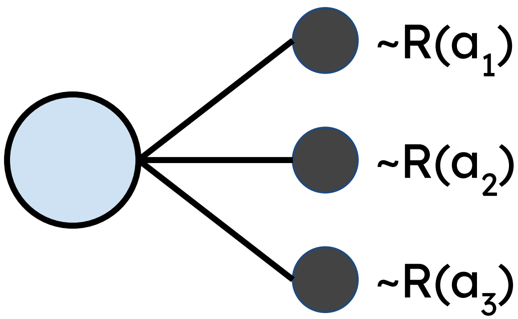
$$Q(a^*) = max_i \; \{Q(a_i)\}$$



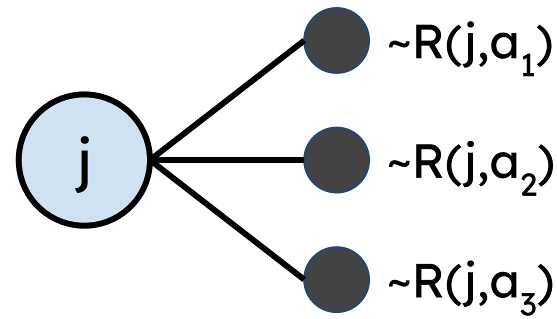
Contextual Bandit Problem

$$Q(j, a^*) = max_i \; \{Q(j, a_i)\}$$

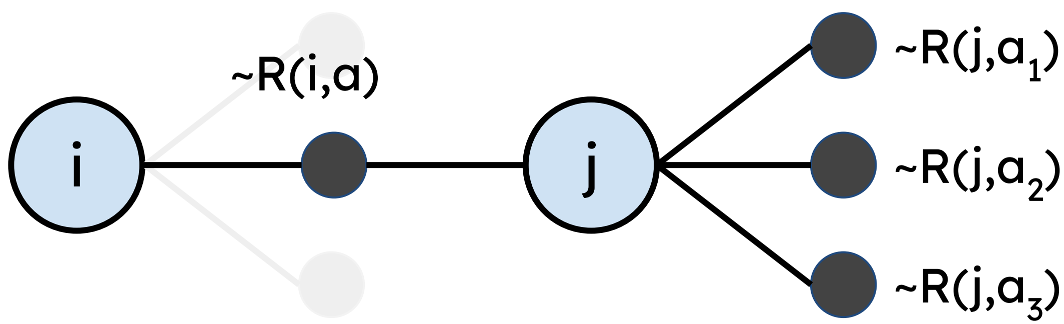
Full RL Problem



Bandit Problem

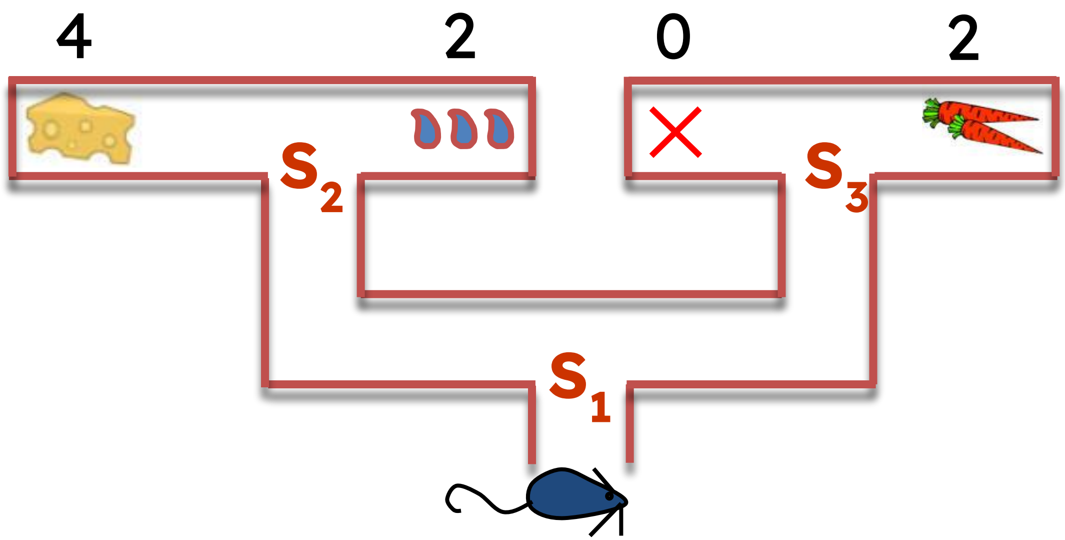


Contextual Bandit Problem



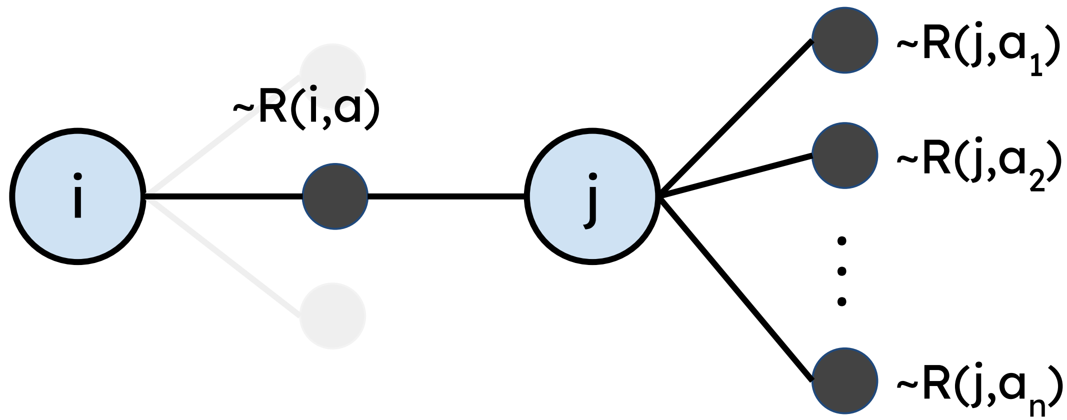
Full Reinforcement Learning Problem

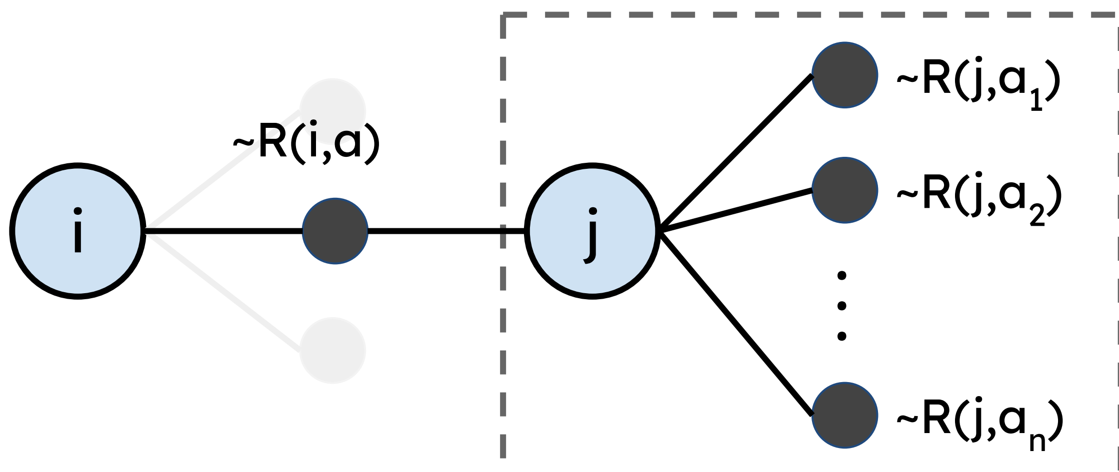
Action at a (Temporal) Distance



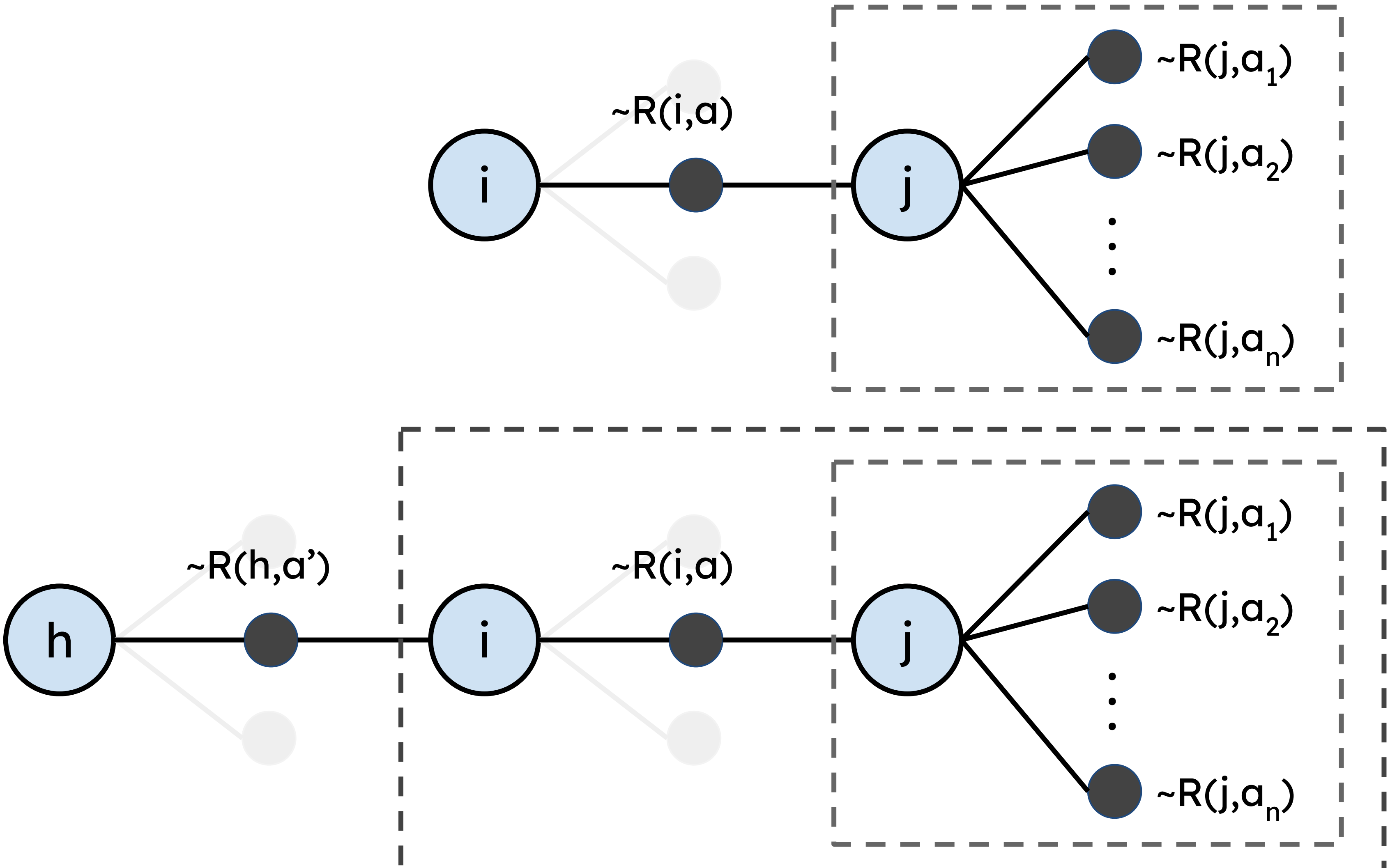
- learning an appropriate action at S_1:
  - depends on the actions at S_2 and S_3
  - gains no immediate feedback
- Idea: use prediction as surrogate feedback

Full RL Problem

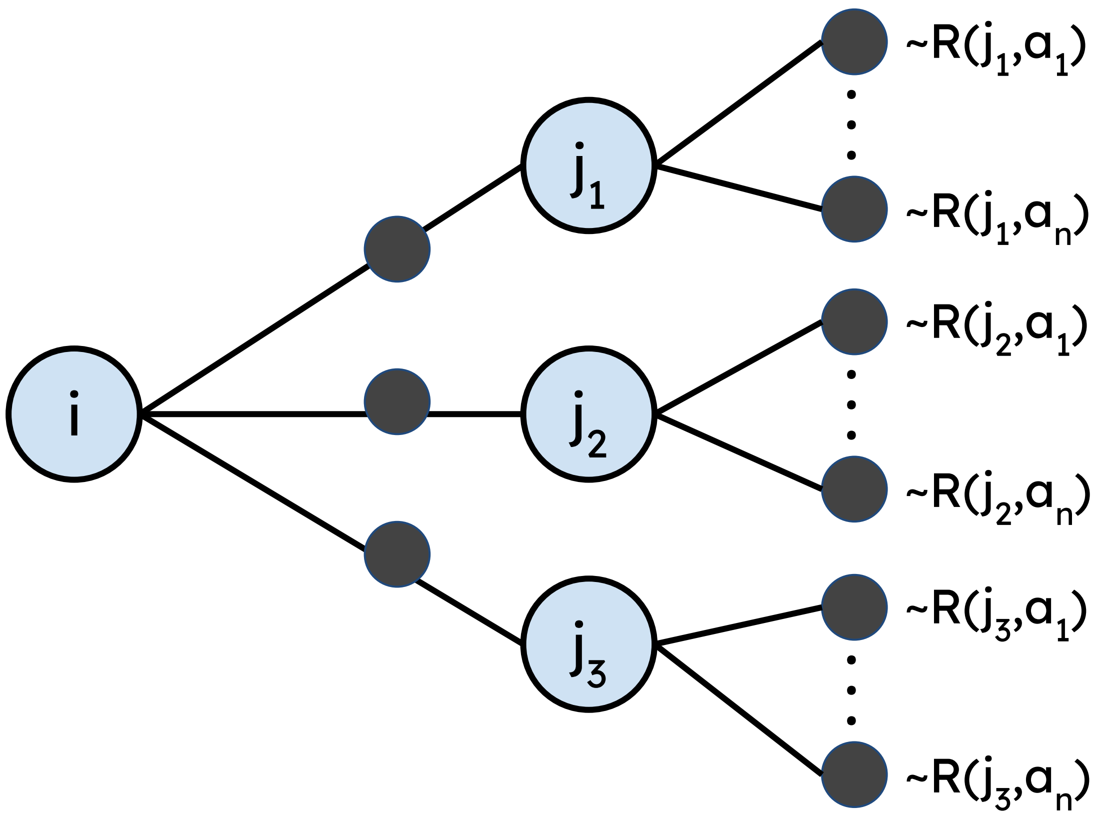




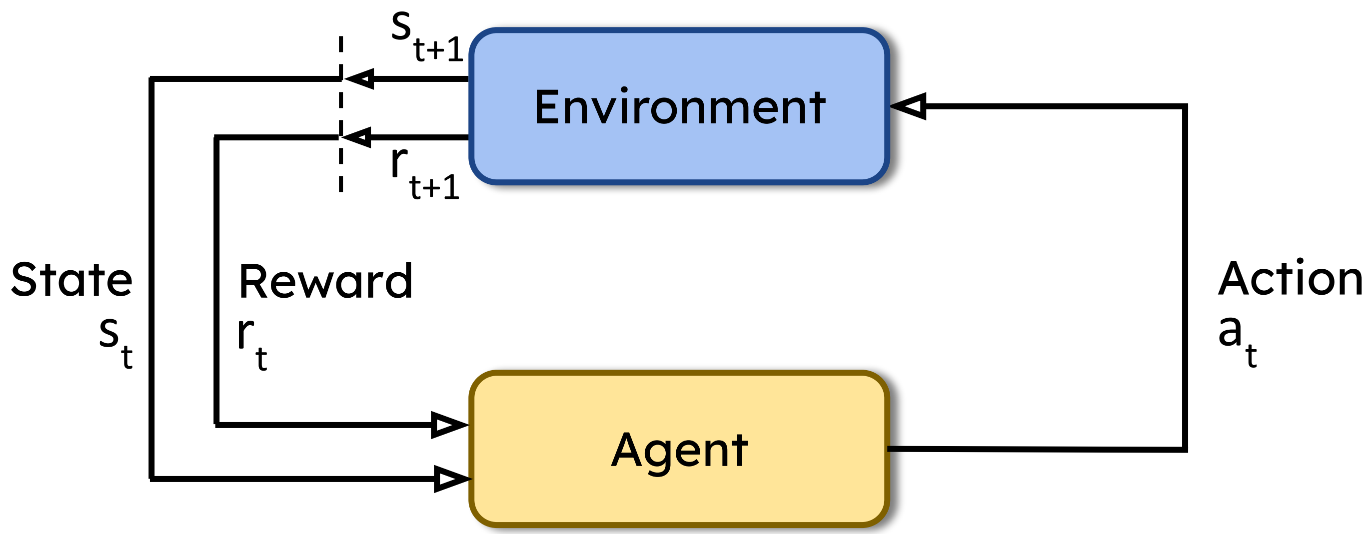
Full RL Problem



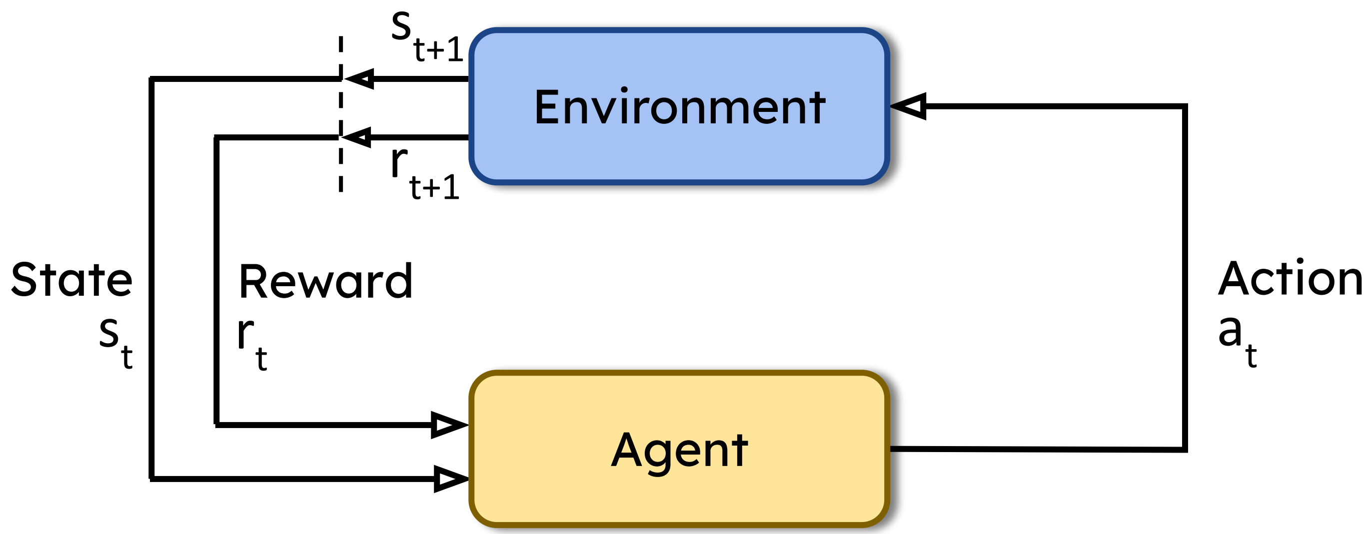
Full RL Problem



The Agent-Environment Interface



The Agent-Environment Interface



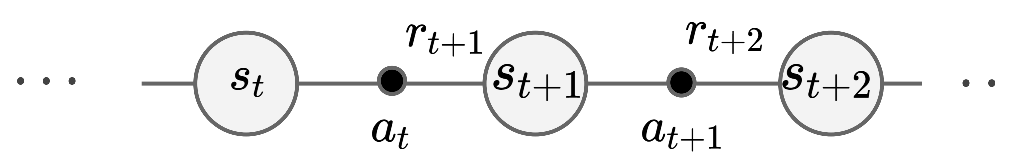
Agent and environment interact at discrete time steps: $t = 0, 1, 2, \ldots$

Agent observes state at step t: $s_t \in S$

produces action at step t : $a_t \in A(s_t)$

gets resulting reward: $r_{t+1} \in \Re$

and resulting next state: s_{t+1}



The Markov Property

- "the state" at step *t*, means whatever information is available to the agent at step *t* about its environment.
- The state can include immediate "sensations", highly processed sensations, and structures built up over time from sequences of sensations.
- Ideally, a state should summarize past sensations so as to retain all '*essential*' information, i.e., it should have the **Markov Property**:

$$\Pr\left\{s_{t+1} = s', r_{t+1} = r \,\middle|\, s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0\right\} = \Pr\left\{s_{t+1} = s', r_{t+1} = r \,\middle|\, s_t, a_t\right\}$$

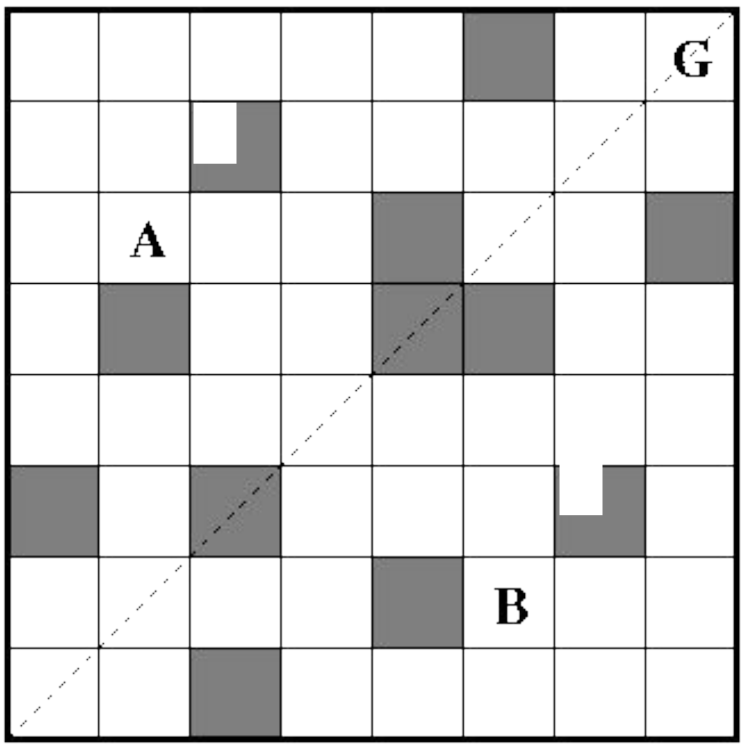
for all s', r, and histories $s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0$.

Markov Decision Processes

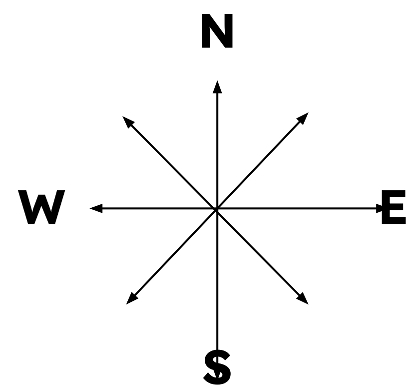
- MDP, *M*, is the tuple: $M = < S, A, r, p >$
  - S : set of states.
  - A : set of actions.
  - $p : S \times A \times S \rightarrow [0, 1]$: probability of transition
  - $r : S \times A \times S \rightarrow \mathbb{R}$: expected reward

- Policy: $\pi : S \times A \rightarrow \left[0,1\right]$ *(can be deterministic)*
- Maximize total expected reward
- Learn an *optimal* policy

Example : 2-D workspace



$$M = \langle S, A, p, r \rangle$$



Robot Control

- **Input** consists of the reading of the sonars, the bump sensors, the camera, the arm position, and wheel encoder
  - **State** is typically a short history of the sensor readings
- **Actions** are the torques to the motors
- **Positive rewards** on achieving the goal; **Negative rewards** for bumping into obstacles

MDP Example

A computer manufacturing company sells a computer that can either be hot-selling, or selling poorly.

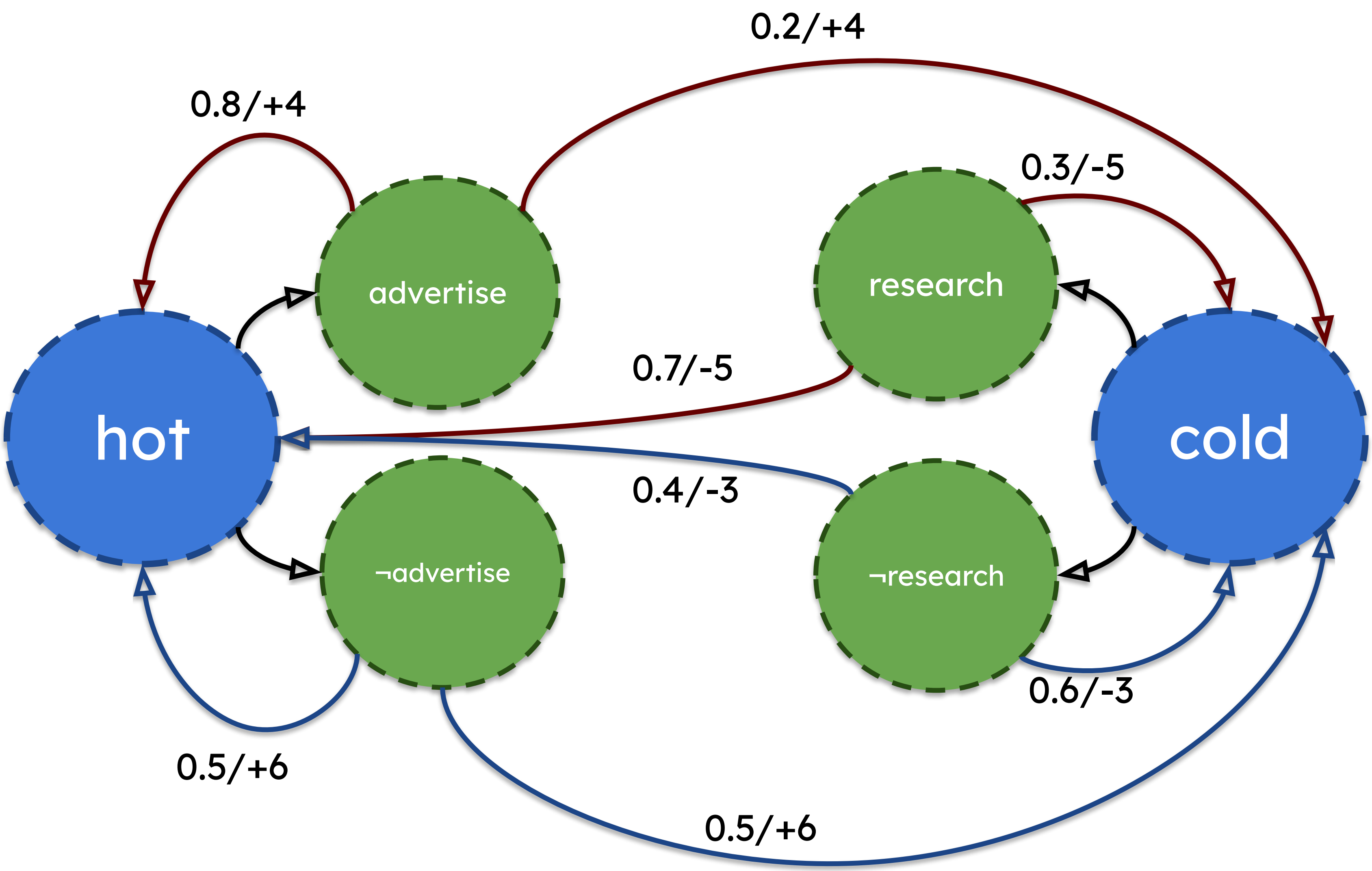
If the computer is hot-selling, the company can advertise it to get an immediate reward of 4 units. On doing so, with probability 0.8, the computer continues to sell well. However, with probability 0.2, the computer starts selling poorly. The company can alternately decide not to advertise a hot-selling computer yielding an immediate reward of 6 units with a 0.5 probability that the computer continues to remain hot-selling.

In case the computer is selling poorly, the company has the option to perform research to improve its computer. This results in an immediate reward of -5 units and a 0.7 probability of the research leading to the computer becoming a hot-selling one. Finally, if the company decides not to perform research for a poorly selling computer, it receives an immediate reward of -3, with the computer continuing to sell poorly with a probability of 0.6.

Formulating MDP

- What are the choices (actions) that can be made in this problem?
  - Actions: A = {advertise, ¬advertise, research, ¬research}
- What are the states in which these choices can be made?
  - advertise or ¬advertise for hot-selling product
  - research or ¬research for poorly-selling product
  - States: S = {hot, cold}
- Applicable actions in each state:
  - A(hot) = {advertise, ¬advertise}
  - A(cold) = {research, ¬research}
- Finally, consider the transition probabilities & rewards

Formulating MDP (Contd.)



The Agent Learns a Policy

Policy at step t, π_t :

a mapping from states to action probabilities

$$\pi_t(s, a) = \text{ probability that } a_t = a \text{ when } s_t = s$$

- Reinforcement Learning methods specify how the agent changes its policy as a result of experience.
- Roughly, the agent's goal is to get as much reward as it can over the long run.

Returns

Suppose the sequence of rewards after step *t* is:

$$r_{t+1}, r_{t+2}, r_{t+3}, \ldots$$

What do we want to maximize?

We want to maximize the **return** - G_t, for each step *t.*

Episodic tasks: interaction breaks naturally into episodes, *e.g., plays of a game, trips through a maze.*

$$G_t = r_{t+1} + r_{t+2} + \cdots + r_T,$$

where *T* is a final time step at which a **terminal state** is reached, ending an episode.

Returns for Continuing Tasks

- **Continuing tasks**: interaction does not have natural episodes.
- **Discounted return**:

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

where $\gamma, 0 \le \gamma \le 1,$ is the **discount rate**.

short-sighted $0 \leftarrow \gamma \rightarrow 1$ far-sighted

In general,

we want to maximize the **expected return**, $E\{G_t\}$, for each step t.

$v_\pi(s)$

Value Functions

- Expected future rewards starting from a state (or state-action pair) and following policy π

State - value function for policy π :

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

Action - value function for policy π :

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

Value Functions

- Expected future rewards starting from a state (or state-action pair) and following policy π

State - value function for policy π :

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

Action - value function for policy π :

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

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a)$$

Solving RL problems

- Learn an *optimal policy* – a mapping from states to actions such that no other policy has a higher long term reward

- Can learn such a policy directly
- Or through estimating an *optimal value function*
- Optimal Value function: The estimated long term reward that you would get starting from a state and behaving optimally