Evaluative vs. Instructive Learning

Learning from instruction:

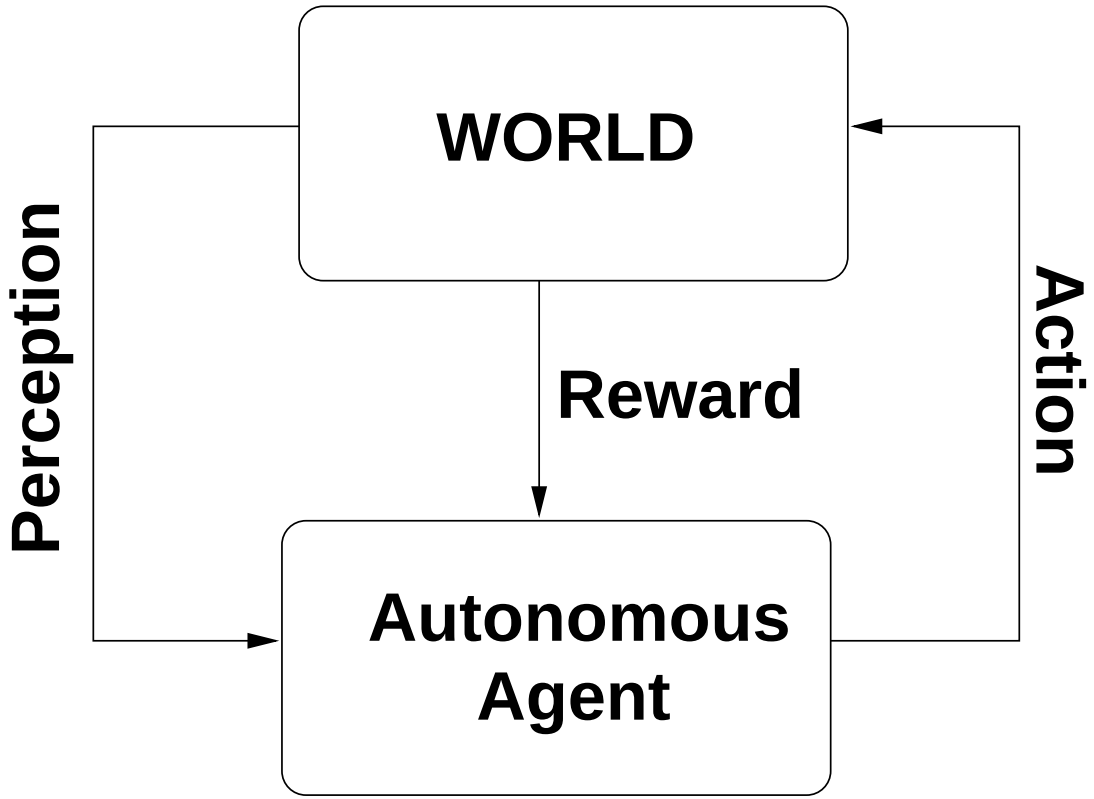
- Examples are labeled
- Teacher tells the learner how to achieve goal.
- Structured controlled mode of learning.

Learning from evaluation:

- Learner is only rewarded for achieving goal, but not told how.
- Learner generates training experience from self-exploration
- Autonomous unstructured mode of learning.

Reinforcement Learning

(Sutton and Barto, MIT Press, 1998)



Reinforcement learning is the study of how an agent interacts with an environment to learn to do a task.

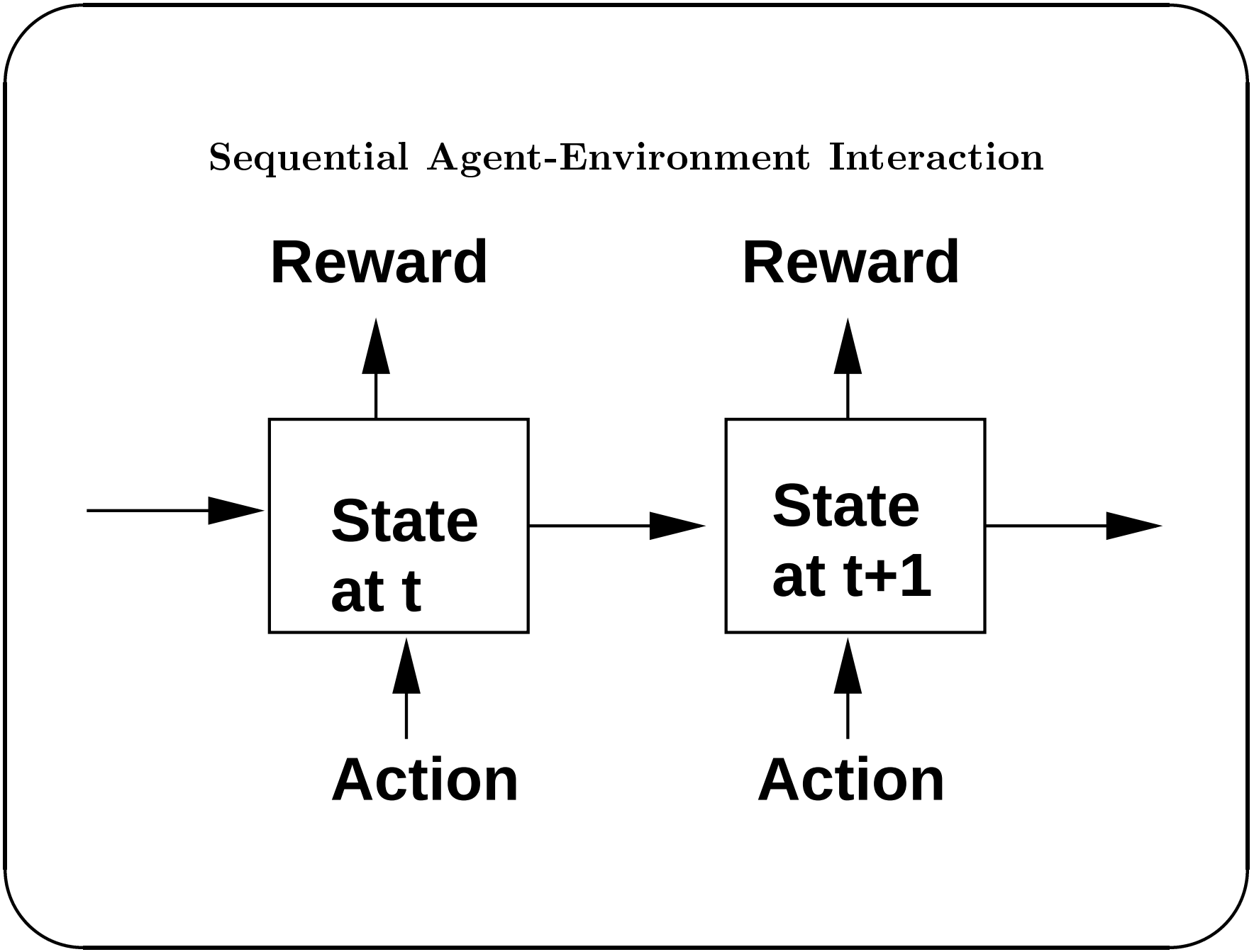
Probabilistic Models of Sequential Processes

Action/State	**Yes**	**No**
Observable	Markov Decision Process	Markov Chain
Hidden	Partially Observable MDP	Hidden Markov Models

Agent Environment Interface

At each time instant $t \geq 0$:

- The agent senses the state of the environment $s_t \in S$
- The agent consults its current *policy* $\pi_t : S \to A(s_t)$ and carries out an action $a_t \in A(s_t)$
- One time instant later, the agent receives a *scalar reward* r_{t+1} (which can be positive, negative, or 0).
- The time counter is incremented $t \leftarrow t + 1$.


Sequential Agent-Environment Interaction
Reward
Reward
State
at t
State
at t+1
Action
Action

Examples of RL Problems I

Robotics

- *States:* Raw sensor values, processed sensory representations, physical locations.
- *Actions:* Motion, manipulation, interaction
- *Reward:* reached goal, collisions, soda cans collected.
- *Examples:*
  - Learning new behaviors – pushing, docking (Asada, Mahadevan & Connell, Lin)
  - Learning behavior coordination – walking (Maes & Brooks)
  - Learning grasping (Salganicoff & Bacjsy)
  - Multi-robot coordination (Mataric)

Examples of RL Problems II

Industrial Optimization

- *States:* machine state/location, schedule, parts.
- *Actions:* Produce part, move machine, add a job.
- *Reward:* Number of parts sold, cost of schedule, waiting time, inventory level.
- *Examples:*
  - Scheduling for the space shuttle (Zhang & Dietterich)
  - Controlling elevators in a high-rise building (Crites & Barto)
  - Frequency allocation for cellphones (Singh & Bertsekas)
  - Flexible manufacturing (Mahadevan et. al)
  - Packaging (Moore et. al)

Examples of RL Problems III

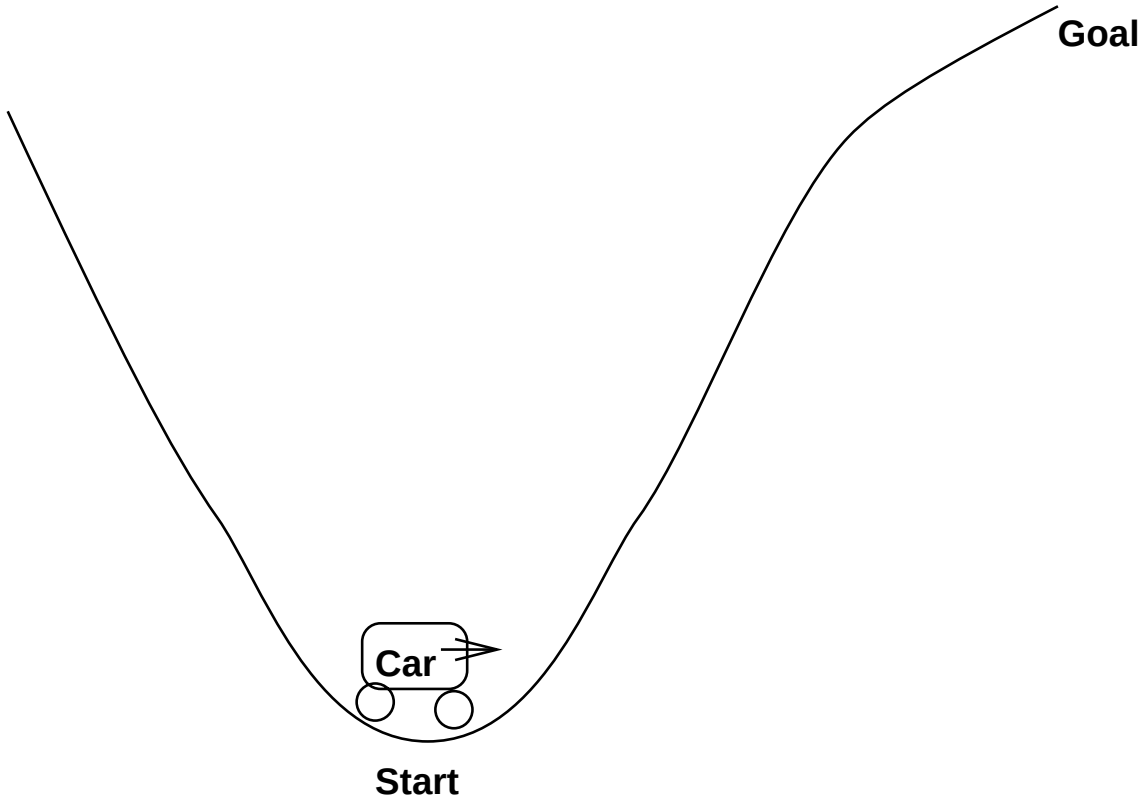
Game playing

- *States:* pieces on the board, location of the agent/monsters.
- *Actions:* move a piece, eat a monster.
- *Reward:* game won/lost, gold pieces collected, monsters eaten.
- *Examples:*
  - Backgammon (Tesauro)
  - Pacman (Lin)
  - Amazon (Chapman & Kaelbling)

Specifying Rewards and Goals

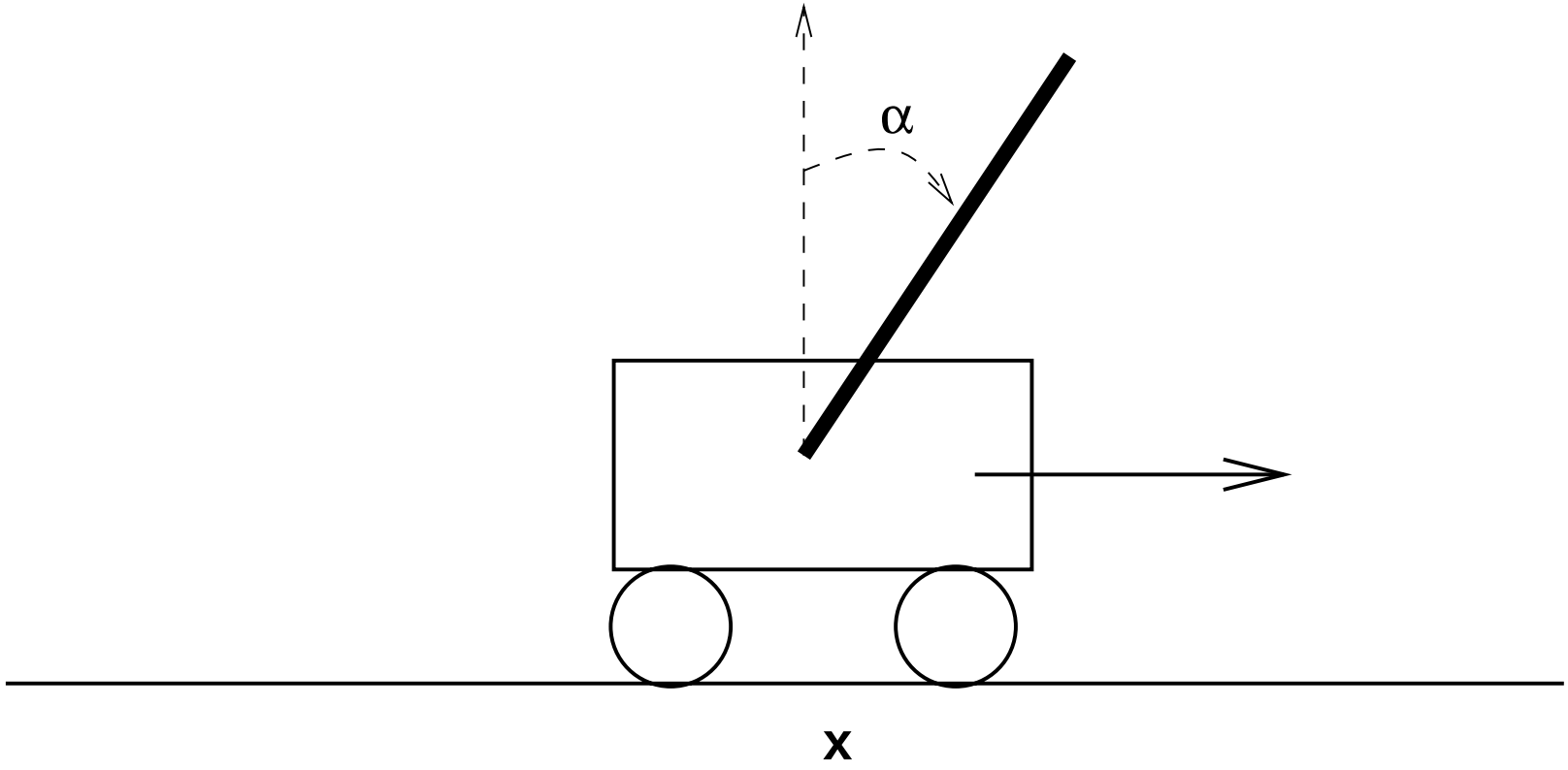
- General principle: reward desired outcome, not a particular solution.
- Shortest path to goal: agent is punished by -1 at every non-goal state.
- Avoid termination: agent is rewarded by $+1$ for every non-terminal state.
- Shaping: reward intermediate subgoals (could lead to sub-optimal solution)

Example: Mountain Car



Goal of car is to get to the top of the hill ASAP, but car is underpowered. States are position on hill and car velocity. Actions are coast, thrust forward, thrust backward.

Example: Pole Balancing



Goal is to keep pole balanced for as long as possible. States are position of cart, angle of pole, and velocity and angular velocity of cart and pole. Actions are apply force to the left, and to the right.

Maximizing Return

Let r_t be the reward received by the agent at time t. What should the goal of the agent be?

- Maximize total reward defined as the *return* R_t^n

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

- What if the task is not *episodic* (no natural termination state)? (e.g. pole balancing, obstacle avoidance)

Infinite-horizon tasks

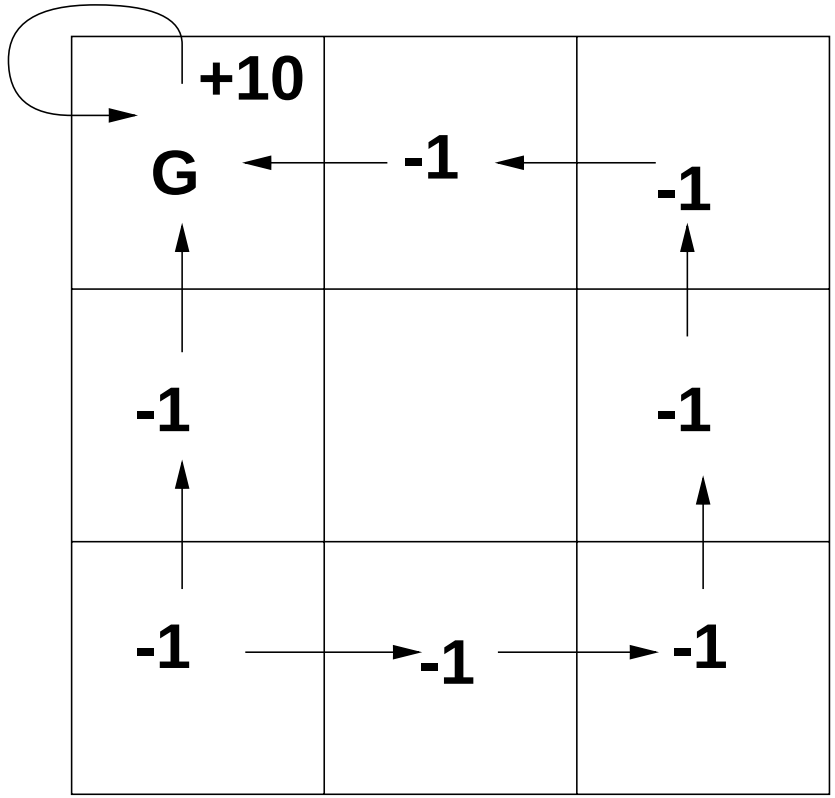
- Non-episodic (continual) tasks generate *infinite-horizon* sequential decision tasks.
- Maximize *discounted* return R_t^γ $(0 \leq \gamma \leq 1)$

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

- Maximize *average-reward* ρ_t

$$\rho_t = \lim_{n \to \infty} \frac{R_t^n}{n}$$

Mapping Episodic Tasks into Continual Tasks



The agent can move N,E,W,S in any state (except when such a move will lead it outside the grid). Agent is rewarded for reaching top-left square.

Markovian Environments

An environment is *Markov* if the future is independent of the past history, given the currrent state. Formally,

$$Pr(s_{t+1} = s, r_{t+1} = r \mid s_0, a_0, s_1, a_1, \ldots, s_t, a_t)$$
$$= Pr(s_{t+1} = s, r_{t+1} = r \mid s_t, a_t)$$

Examples of Markovian environments:

- Backgammon, chess, Pacman.
- Manufacturing, scheduling

What is an example of a non-Markovian environment?

Can any non-Markovian task be made Markovian?

Markov Decision Process

A Markov Decision Process (MDP) is specified using the following components:

- A finite set of states S
- A finite set of actions A (where $A(s)$ is the set of actions possible in state s)
- A transition matrix specifying the one-step dynamics of the environment $P^a_{ss'} = Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$
- A reward function $R^a_{ss'} = E(r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s')$

Value Function

The *value function* represents the long-term utility of a state (or state-action pair). It is associated with a particular policy. Formally, the value function for a policy π is

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

The *action value* function for a policy π is

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

How do we compute the (action) value function of a policy π?

Bellman Equation

Assume policy π is stationary and deterministic. The value function can be expressed as a set of linear equations, one for each state s (or one for each state-action pair).

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\left(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots\right) \\ &= E_{\pi}\left(r_{t+1} + \gamma \sum_{k=0}^{k=\infty} \gamma^k r_{t+k+2} \mid s_t = s\right) \\ &= \sum_{s' \in S} P_{ss'}^{\pi(s)}\left(R_{ss'}^{\pi(s)} + \gamma V^{\pi}(s')\right) \end{aligned}$$

Optimal Value Functions

We can define a partial ordering on policies, using their associated value functions. A policy $\pi \geq \pi'$ if over all states $s \in S$, $V^{\pi} \geq V^{\pi'}$.

Fundamental result: For any MDP, there is always an *optimal policy* π^* such that $V^{\pi^*} = V^* \geq V^{\pi}$ for any other policy π. In particular, over all $s \in S$,

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

There may be many optimal policies, but they all define the same optimal value function. The optimal *action-value* function also exists and is unique.

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$$

Optimal Value Function

Note that we can express the optimal action value function in terms of the optimal value function (and vice versa).

$$Q^*(s,a) = E\left(r_{t+1} + \gamma V^*(s_{t+1} \mid s_t = s, a_t = a)\right)$$

Also,

$$V^*(s) = \max_{a \in A(s)} Q^*(s,a)$$

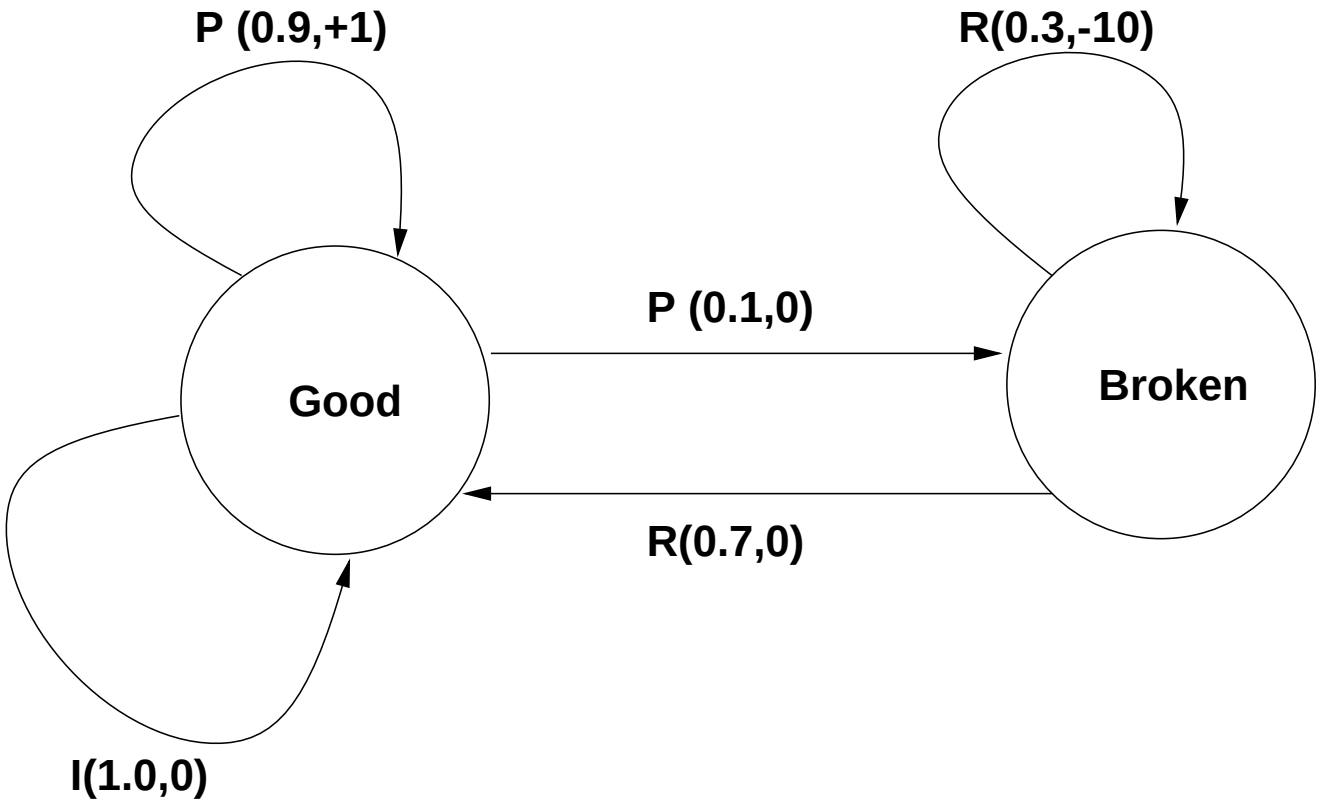
How to compute the optimal value function?

Bellman Optimality Equation

Note that

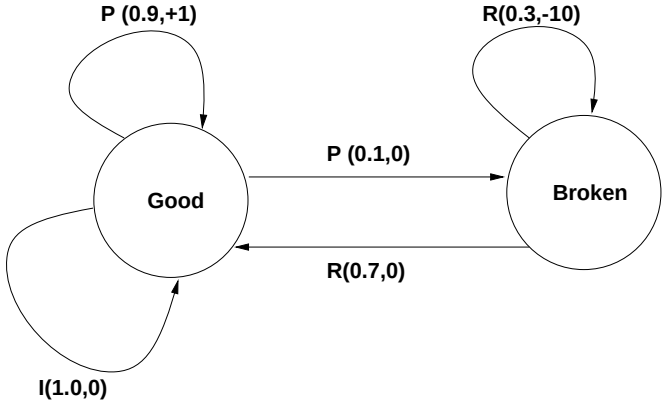
$$
\begin{aligned}
V^*(s) &= \max_{a \in A(s)} Q^*(s,a) \\
&= \max_a E_{\pi^*}\left[R_t \mid s_t = s, a_t = a\right] \\
&= \max_a E_{\pi^*}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right) \\
&= \max_a E_{\pi^*}\left(r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s, a_t = a\right) \\
&= \max_a E\left(r_{t+1} + \gamma V^*(s_{t+1} \mid s_t = s, a_t = a)\right) \\
&= \max_{a \in A(s)} \sum_{s' \in S} P^a_{ss'}\left(R^a_{ss'} + \gamma V^*(s')\right)
\end{aligned}
$$

Example MDP from Manufacturing



Machine can be in a good state or broken. Producing a part gives a reward, but ages the machine. If the machine fails, it costs some money to fix it.

Value Function for Manufacturing MDP



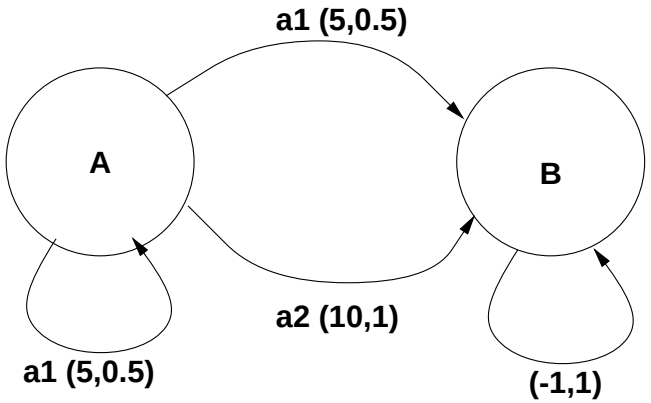
What is the value function of the policy π that chooses the "produce" action when the machine is in "good" state? Assume $\gamma = 0.5$.

$$\begin{aligned} V^\pi(g) &= (0.9)(1) + (0.1)(0) + \gamma\left((0.9)V^\pi(g) + (0.1)V^\pi(b)\right) \\ V^\pi(b) &= (0.3)(-10) + (0.7)(0) + \gamma\left((0.3)V^\pi(b) + (0.7)V^\pi(g)\right) \end{aligned}$$

Solving this gives us $V^\pi(g) = 1.36$, and $V^\pi(b) = -2.97$.

Exercise: calculate the value function of the "idle" policy.

Example of Computing Optimal Value Function



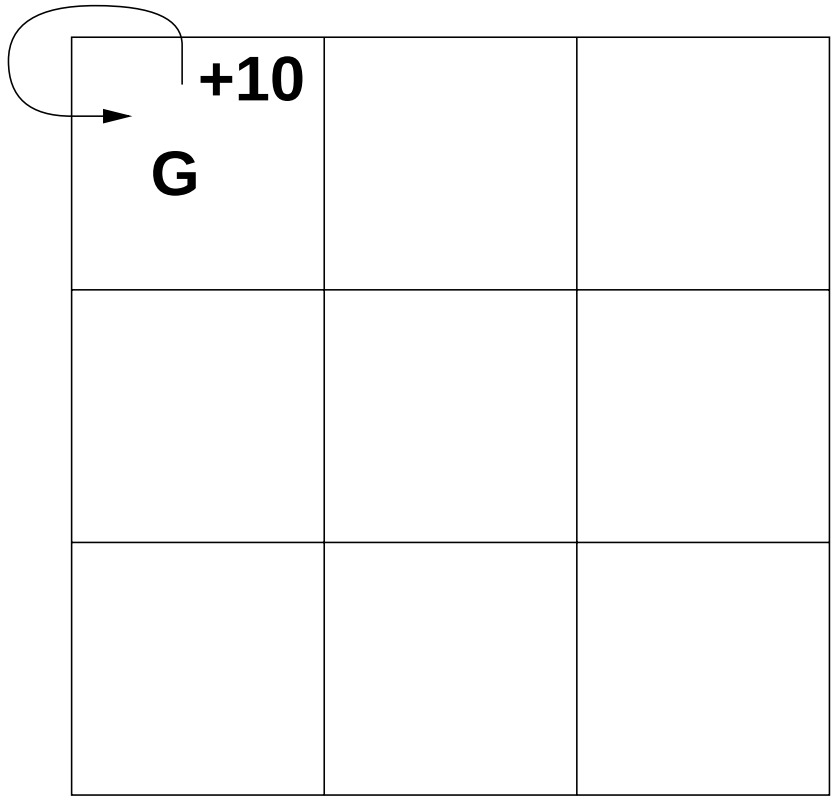
Using the Bellman equation, we get

$$\begin{aligned} V^*(A) &= \max\left(5 + \gamma(0.5)(V^*(A) + V^*(B)), 10 + \gamma V^*(B)\right) \\ V^*(B) &= -1 + \gamma V^*(B) \end{aligned}$$

Solving these equations for $\gamma = 0.5$, we get

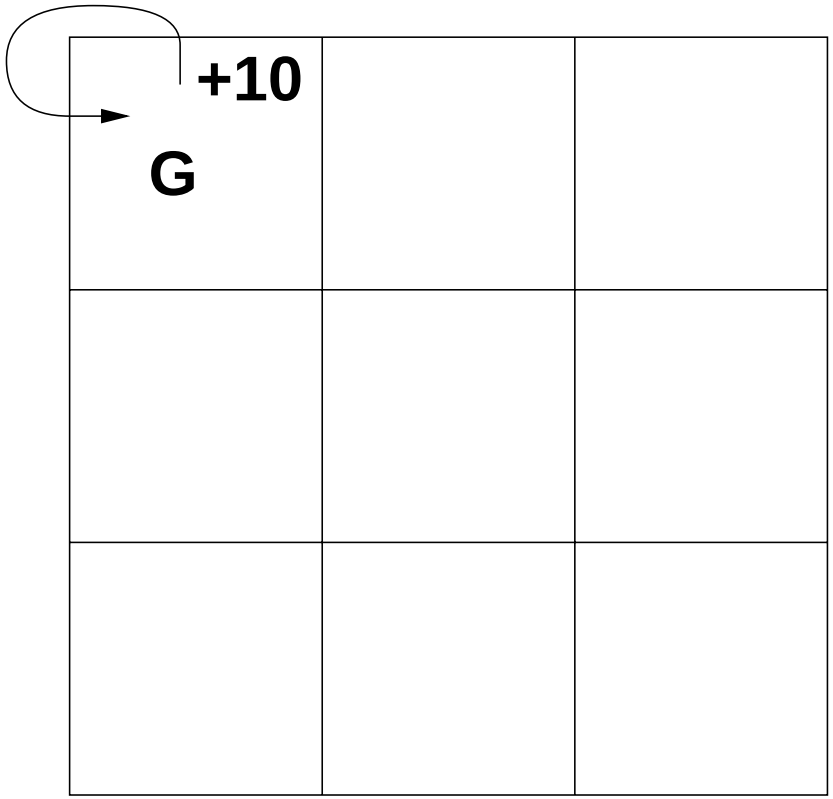
$$\begin{aligned} V^*(B) &= \frac{-1}{1-\gamma} = -2 \\ V^*(A) &= \max\left(\frac{5 + (0.5)(0.5)(-2)}{1 - (0.5)(0.5)}, 10 + (0.5)(-2)\right) = 9 \end{aligned}$$

Computing Optimal Value Function for Grid World



Every non-goal transition is punished by -1. What is the optimal value function (assuming a discount factor of 0.9)?

Computing Optimal Value Function for Grid World



Every non-goal transition is punished by -1. Actions are deterministic. What is the optimal value function (assuming a discount factor of 0.5)?

Summary of Fundamental Equations

Optimal value function:

$$V^*(s) = \max_{a \in A(s)} \sum_{s' \in S} P^a_{ss'} \left(R^a_{ss'} + \gamma V^*(s') \right)$$

Optimal action value function:

$$Q^*(s,a) = \sum_{s' \in S} P^a_{ss'} \left(R^a_{ss'} + \gamma \max_{a'} Q^*(s',a') \right)$$

Questions:

- What are efficient algorithms for computing (learning) these value functions?
- How to represent the value function on a very large (infinite) state space?
- What if the one-step dynamics and rewards are unknown?

Dynamic Programming

(Bellman, Howard)

- Assume one-step dynamics $P^a_{ss'}$ and expected rewards $R^a_{ss'}$ are known.
- Value iteration
- Policy iteration
- At each step, DP carries out a sweep over the whole state space (updating the value of each state)

Reinforcement learning algorithms can be viewed as approximate DP algorithms.

Value Iteration

Evaluating a fixed policy:

Repeat over all states $s \in S$

$$V^{k+1}(s) = \sum_{s' \in S} P_{ss'}^{\pi(s)} \left(R_{ss'}^{\pi(s)} + \gamma V^k(s') \right)$$

Until $\max_{s \in S} \mid V^{k+1}(s) - V^k(s) \mid \leq \epsilon$

Finding the optimal value function:

Repeat over all states $s \in S$

$$V^{k+1}(s) = \max_{a \in A(s)} \sum_{s' \in S} P_{ss'}^{a} \left(R_{ss'}^{a} + \gamma V^k(s') \right)$$

Until $\max_{s \in S} \mid V^{k+1}(s) - V^k(s) \mid \leq \epsilon$