Machine Learning I
MATH60629

Sequential Decision Making I
Summary
— Week #12

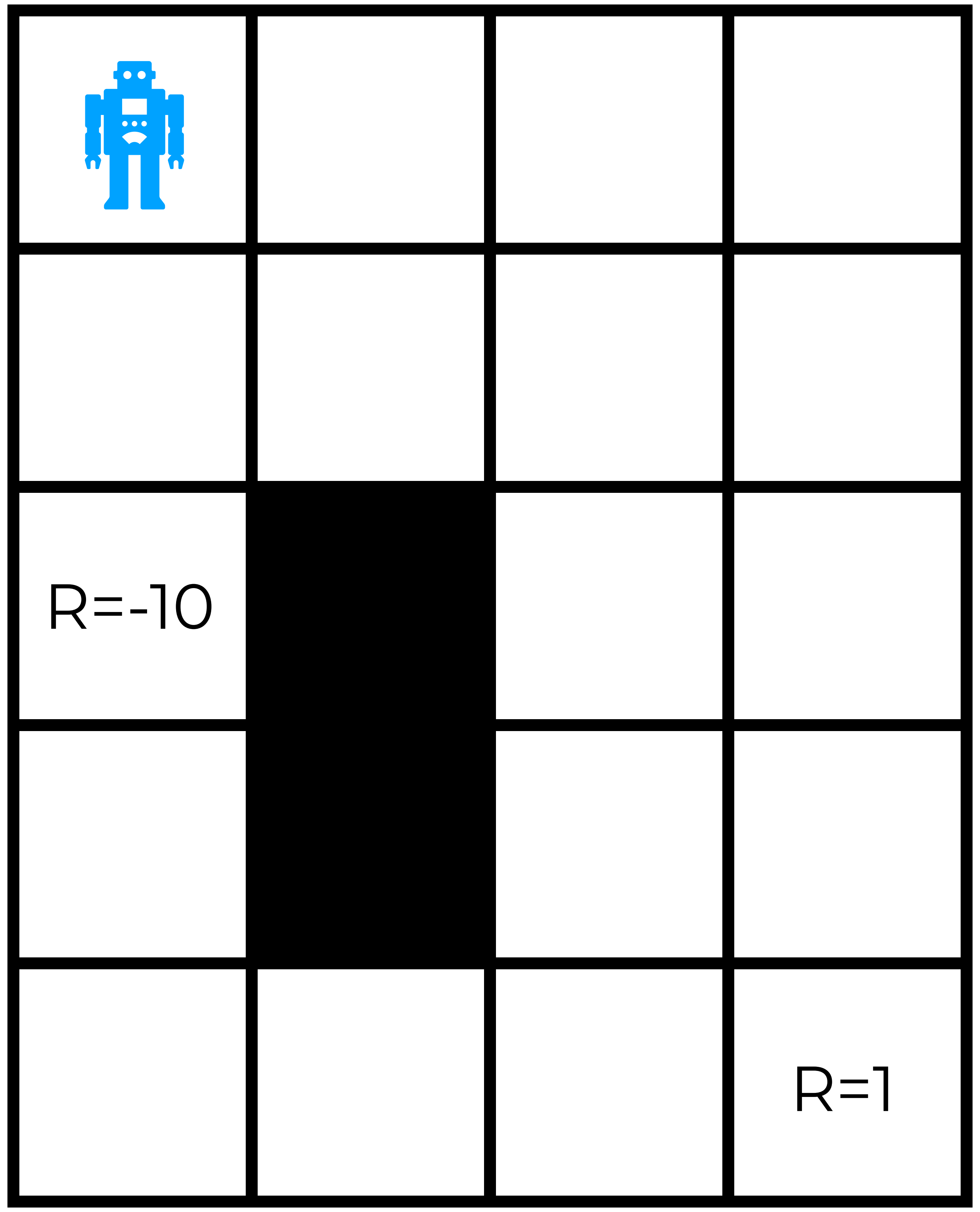
Recall

Three main components

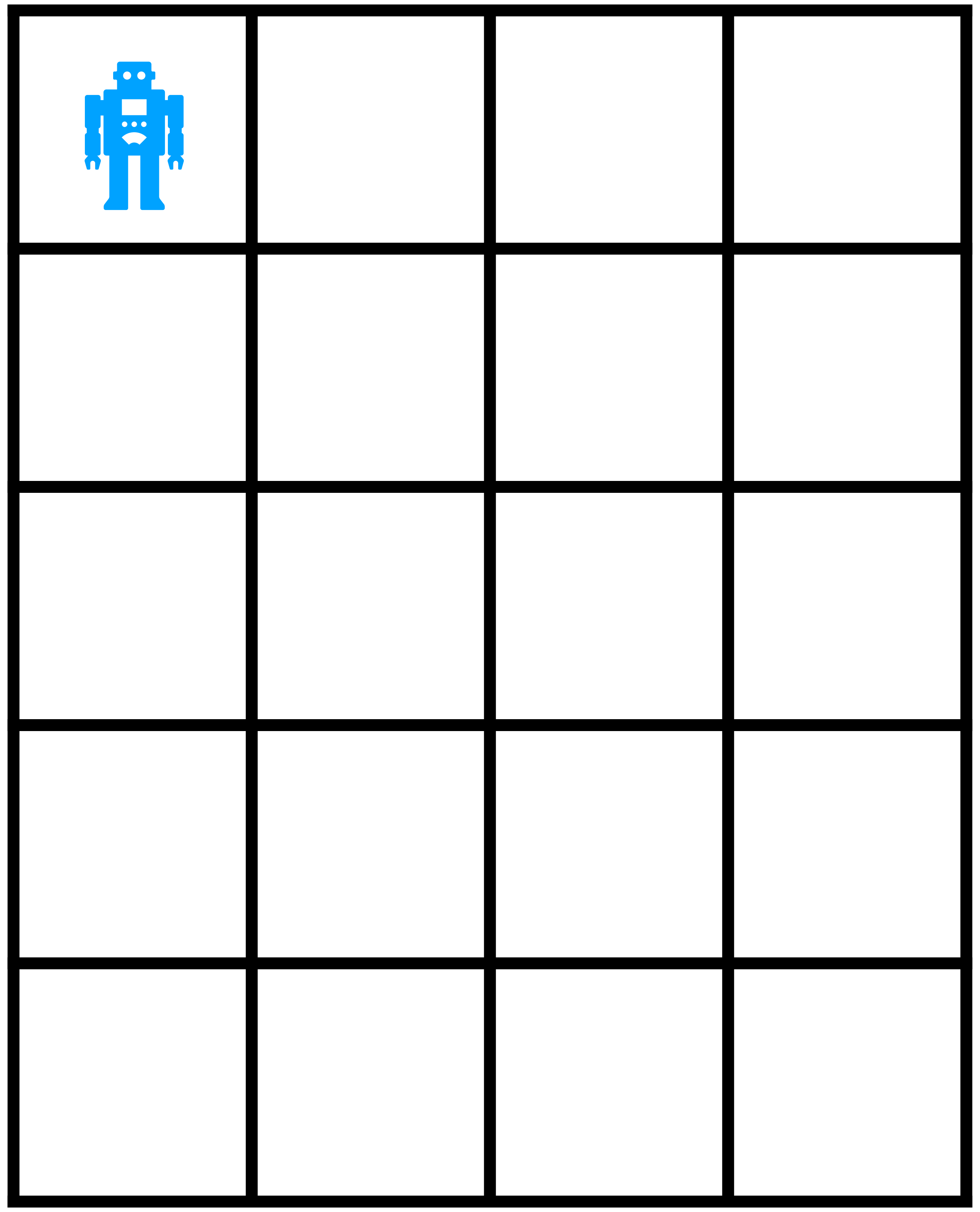
- Task (T)
- Performance measure (P)
- Experience (E)

Supervised learning

- Experience a fixed data set
  - Fit a model using this data
  - Use the model to make predictions about unseen data
  - Eventually: Predictions may be used downstream

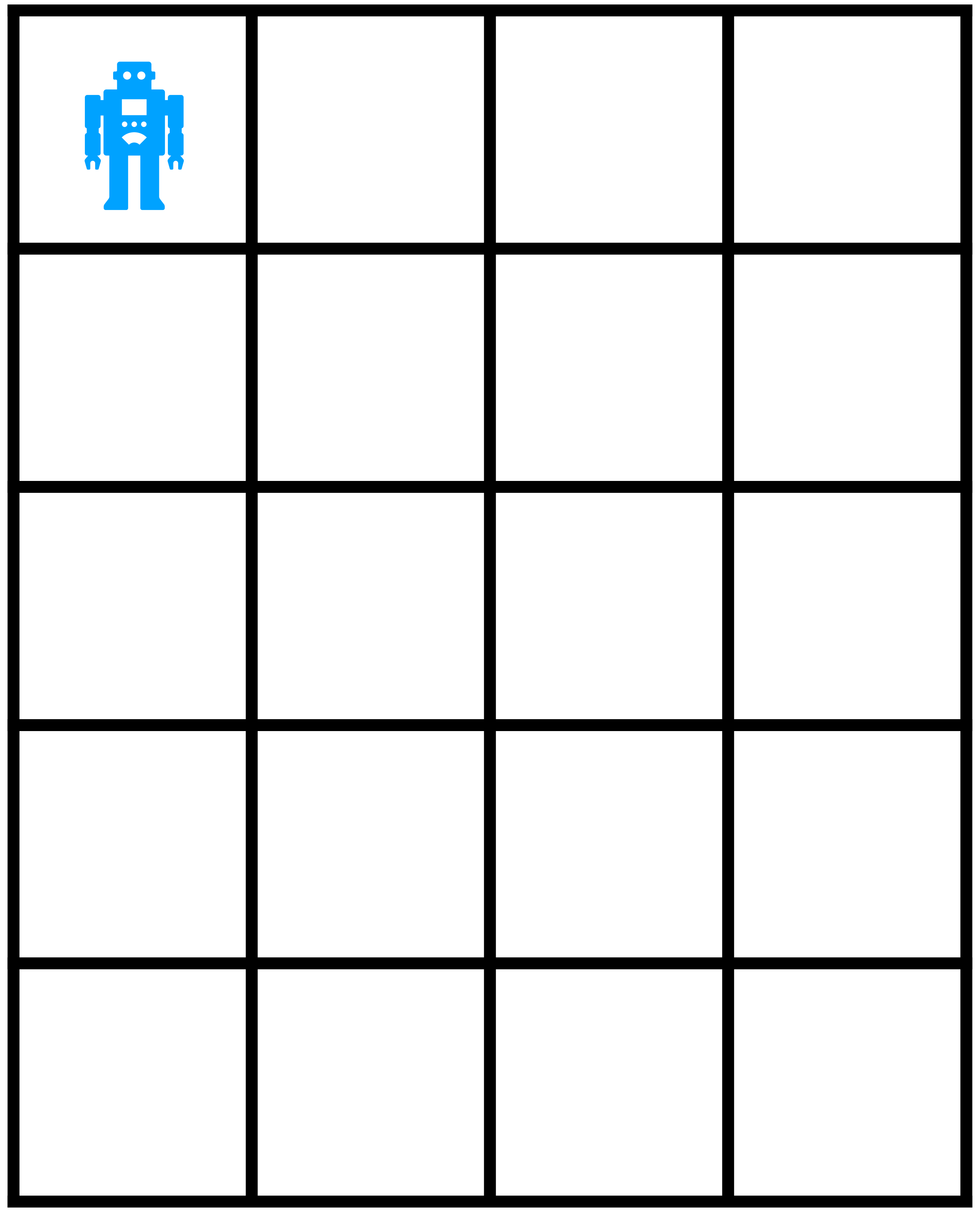

R=-10
R=1

Initial example with grid world



- Each cell is a state (S)
- Actions indicate which movements are possible: $A := \{L, R, U, D\}$
- Rewards encode the task: $R(s)$
- Transition probabilities encode the outcome of an action: $P(s' \mid s, a)$

Initial example with grid world



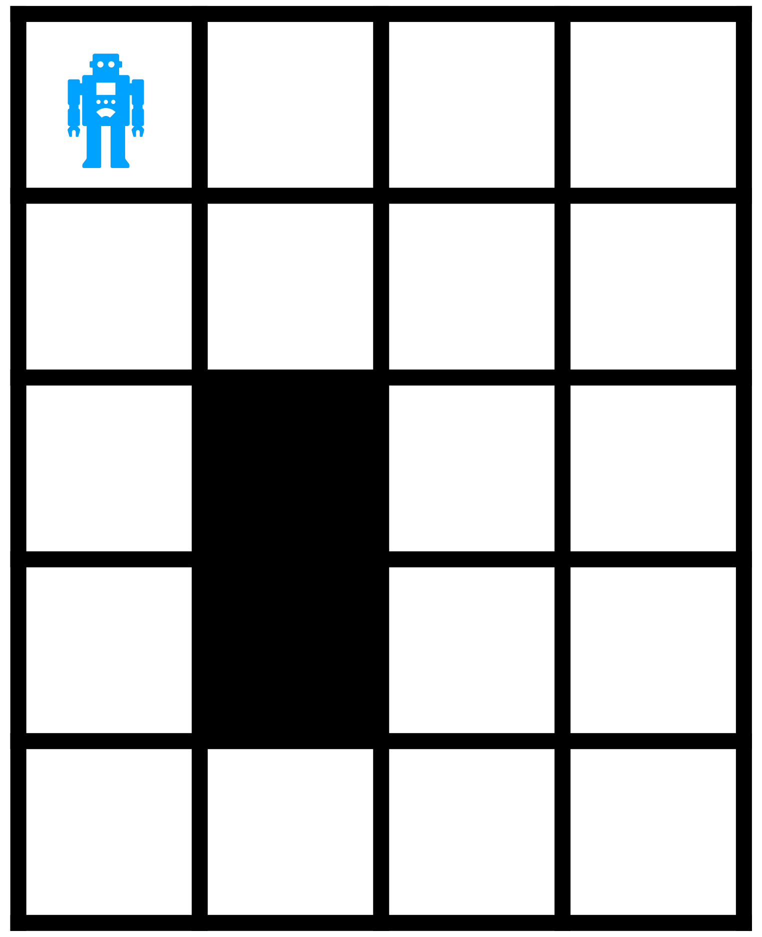
- Each cell is a state (S)
- Actions indicate which movements are possible: $A := \{L, R, U, D\}$
- Rewards encode the task: $R(s)$
- Transition probabilities encode the outcome of an action: $P(s' \mid s, a)$

Planning

This week we discuss a version of RL where these are observed

Markov Decision Process (MDP)

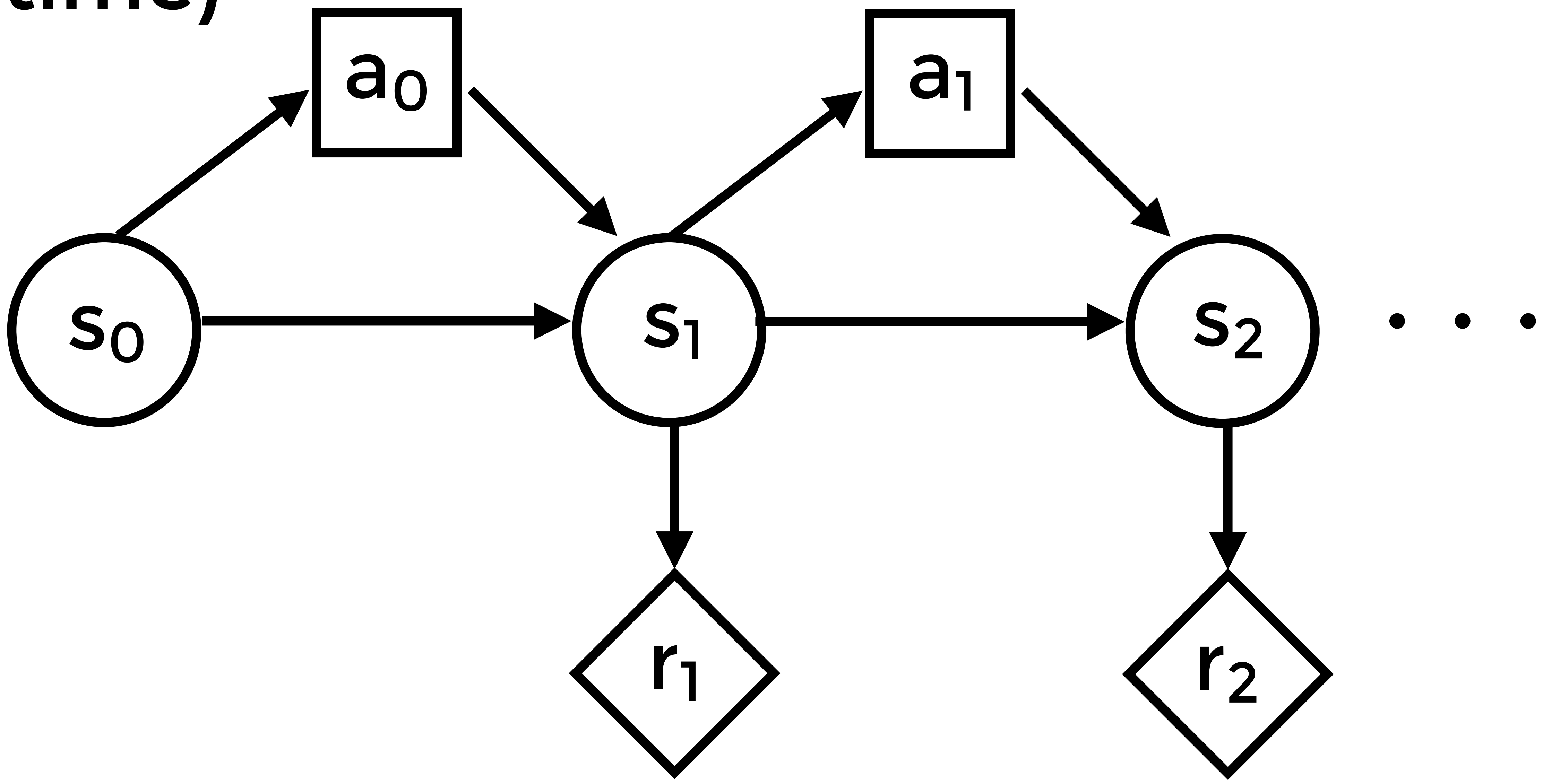
- Provide a framework for decision-making under uncertainty
  - Markov process with decisions and utilities
  - Assumes stationarity (i.e., transitions are fixed across time)

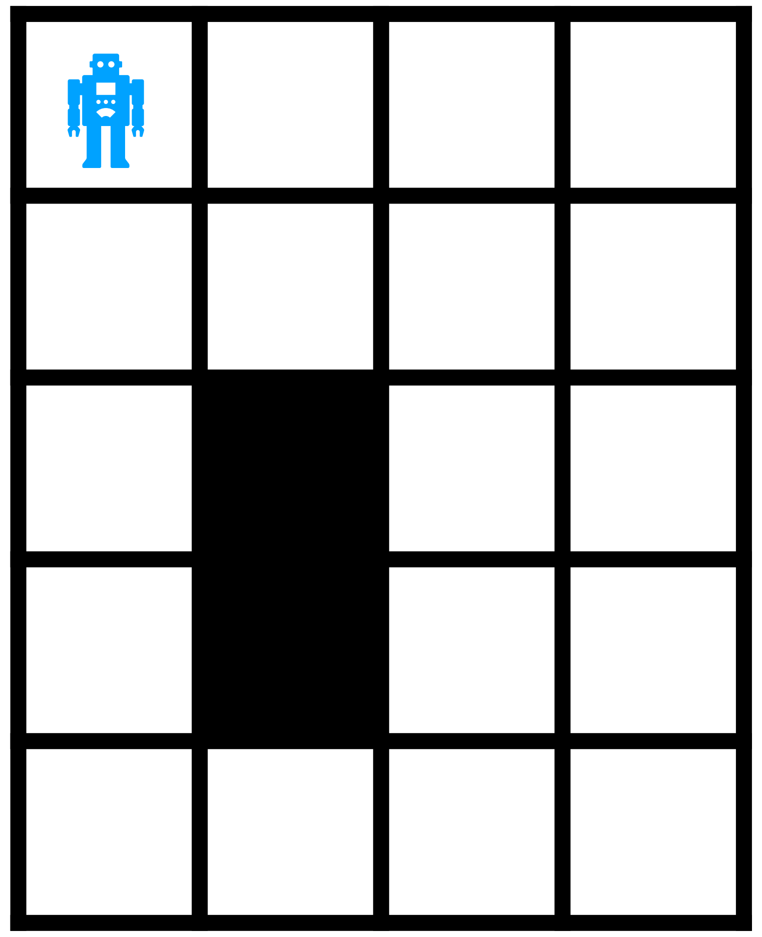


Markov Decision Process (MDP)

- Provide a framework for decision-making under uncertainty
  - Markov process with decisions and utilities
  - Assumes stationarity (i.e., transitions are fixed across time)

- Square nodes: decisions
- Circle nodes: States
- Diamond nodes: utility

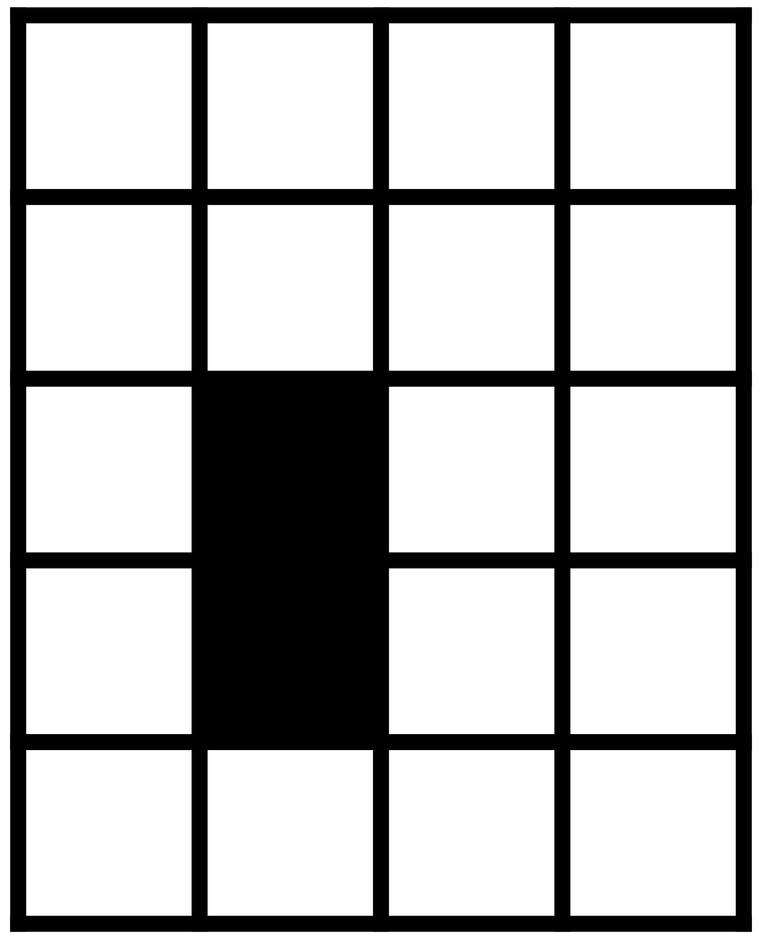




Markov Decision Process (MDP)

$\langle A, S, P, R, \gamma \rangle$

- A: set of actions
- P(S' | S,A): transition probabilities
- R(S): reward function
- γ : discount factor $\in [0, 1]$

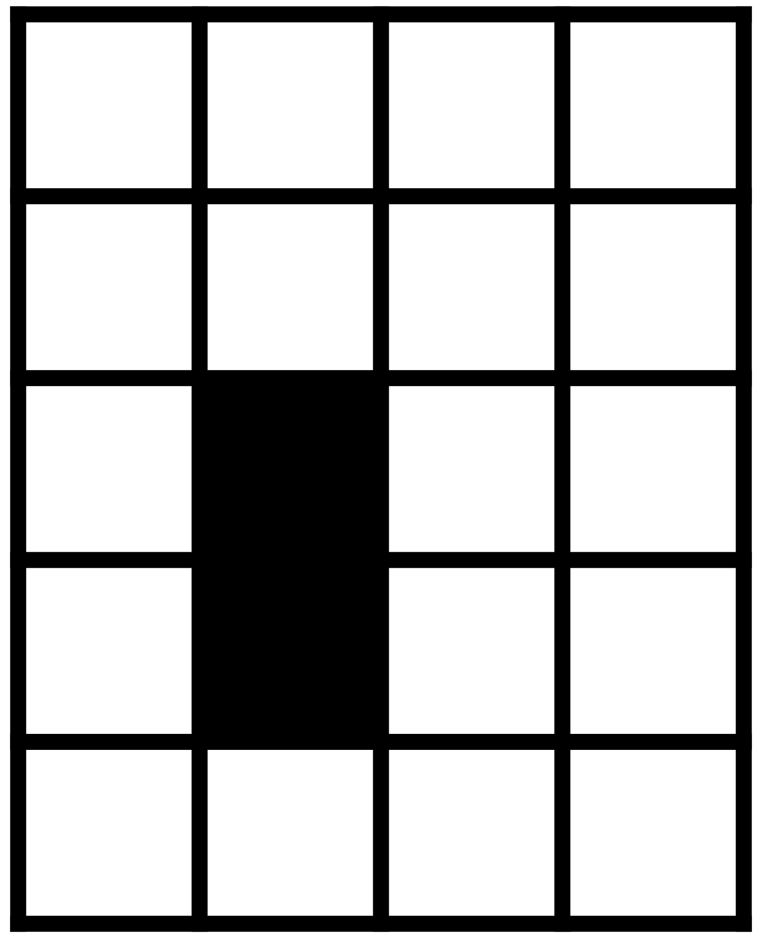


Markov Decision Process (MDP)

$\langle A, S, P, R, \gamma \rangle$

- A: set of actions
- P(S' | S,A): transition probabilities
- R(S): reward function
- γ : discount factor $\in [0, 1]$

- A policy: $\pi : S \rightarrow A$

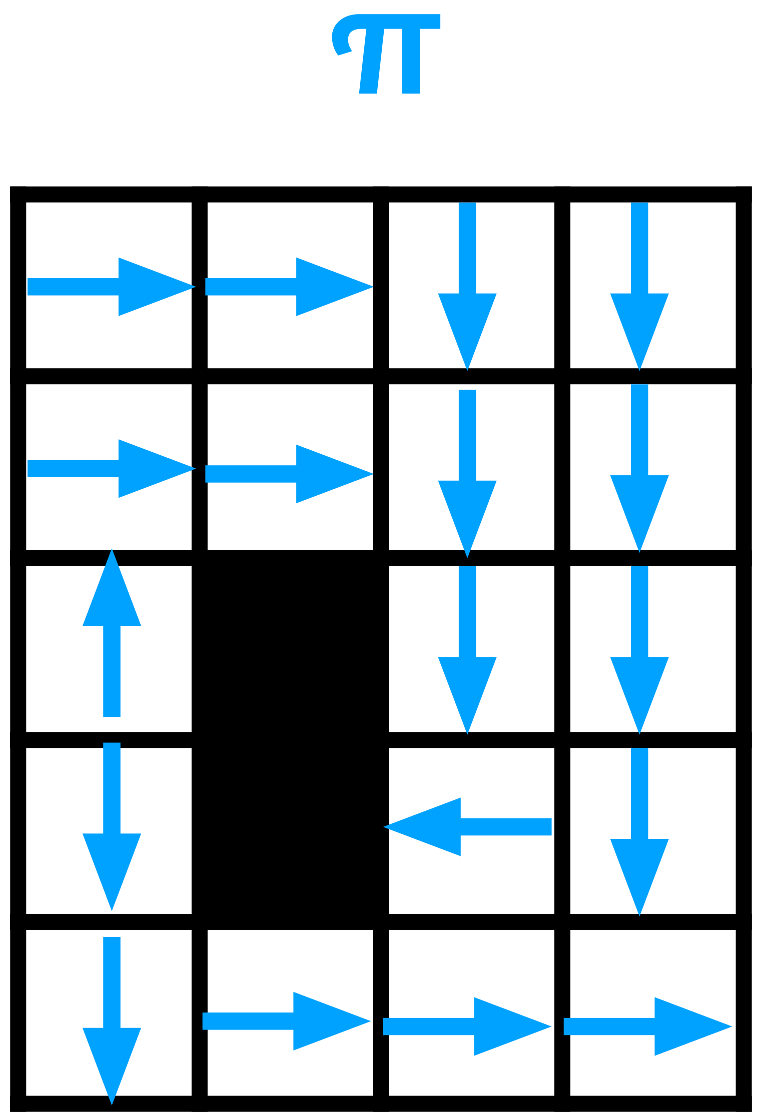


Markov Decision Process (MDP)

$\langle \text{A}, \text{S}, \text{P}, \text{R}, \gamma \rangle$

- A: set of actions
- P(S' | S,A): transition probabilities
- R(S): reward function
- γ : discount factor $\in [0, 1]$

- A policy: $\pi : \text{S} \to \text{A}$

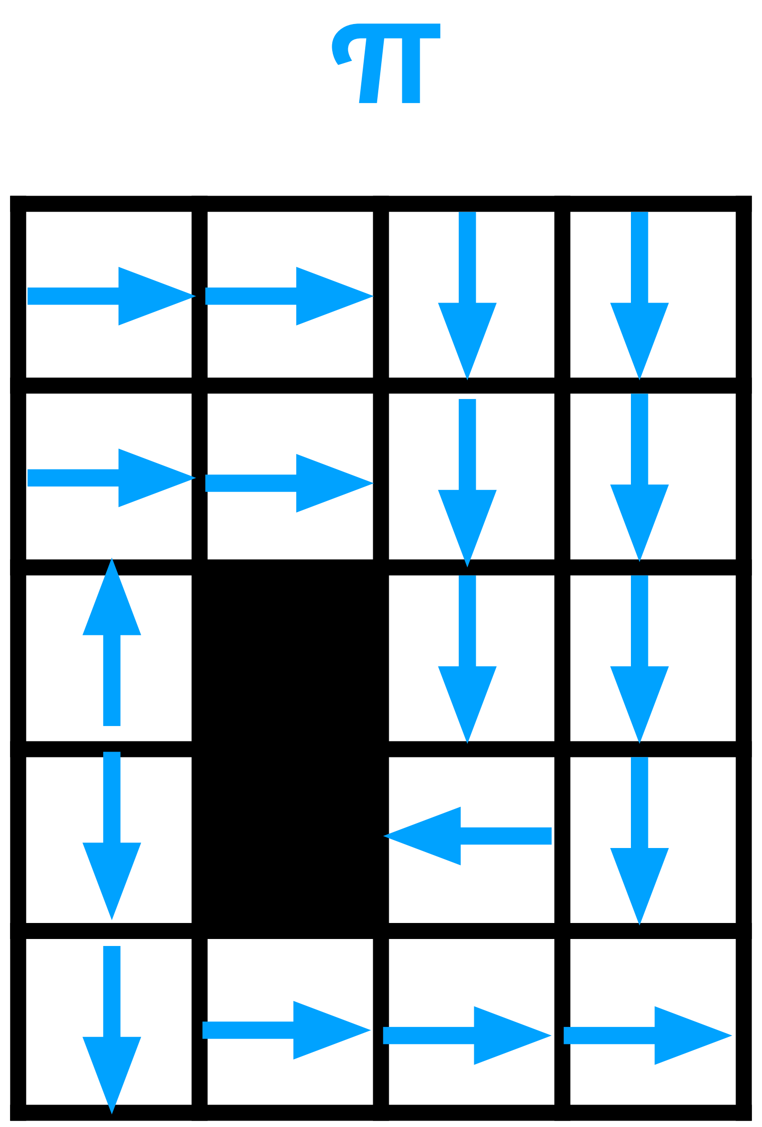


Markov Decision Process (MDP)

$\langle$A, S, P, R, $\gamma\rangle$

- A: set of actions
- P(S' | S,A): transition probabilities
- R(S): reward function
- γ : discount factor $\in [0, 1]$

- A policy: π : S $\rightarrow$ A
- **Goal:** find the optimal policy



Solving an MDP

- **Find the optimal policy of an MDP**

$$\boldsymbol{\pi}^*(\mathbf{s}) \quad \forall \mathbf{s}$$

Solving an MDP

- **Find the optimal policy of an MDP**

$$\pi^*(s) \;\; \forall s$$

- **Policies are evaluated using their expected utility:**

$$EU(\pi) = \sum_{t=0}^{\infty} \gamma^t \sum_{s_{t+1}} P(s_{t+1} \mid s_t, \pi(s_t)) R(s_{t+1})$$

Solving an MDP

- Find the optimal policy of an MDP

$$\pi^*(s) \quad \forall s$$

- Policies are evaluated using their expected utility:

$$EU(\pi) = \sum_{t=0}^{\infty} \gamma^t \sum_{s_{t+1}} P(s_{t+1} \mid s_t, \pi(s_t)) R(s_{t+1})$$

Solving an MDP

- Find the optimal policy of an MDP

$$\pi^*(s) \ \ \forall s$$

- Policies are evaluated using their expected utility:

$$EU(\pi) = \boxed{\sum_{t=0}^{\infty} \gamma^t} \sum_{s_{t+1}} P(s_{t+1} \mid s_t, \pi(s_t)) \boxed{R(s_{t+1})}$$

- The optimal policy is the one with highest expected utility: $EU(\pi^*) \geq EU(\pi) \ \ \forall \pi$

Solving an MDP

- Three well-known techniques:
  1. Value iteration
  2. Policy Iteration
  3. Linear Programming

Value Function

- $V(s_t)$: The value of being in state s at time t

Value Function

- $V(s_t)$: The value of being in state s at time t

$$V(s_t) := \text{expected sum of rewards of being in s}$$

Finite horizon

- Assume that the process has T steps

Finite horizon

- Assume that the process has T steps
- The value at step T is

Finite horizon

- Assume that the process has T steps
- The value at step T is $V(s_T) = R(s_T)$

Finite horizon

- Assume that the process has T steps
- The value at step T is $V(s_T) = R(s_T)$
- The value at step T-1 is

$$V(s_{T-1}) = \max_{a_{T-1}} \left\{ R(s_{T-1}) + \gamma \sum_{s_T} P(s_T \mid s_{T-1}, a_{T-1}) R(s_T) \right\}$$

Finite horizon

- Assume that the process has T steps
- The value at step T is $V(s_T) = R(s_T)$
- The value at step T-1 is

$$V(s_{T-1}) = \max_{a_{T-1}} \left\{ R(s_{T-1}) + \gamma \sum_{s_T} P(s_T \mid s_{T-1}, a_{T-1}) R(s_T) \right\}$$

- The value at step t is$(0 \leq t \leq T)$

$$V(s_t) = \max_{a_t} \left\{ R(s_t) + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t) V(s_{t+1}) \right\}$$

Finite horizon

- Assume that the process has T steps
- The value at step T is $V(s_T) = R(s_T)$
- The value at step T-1 is

$$V(s_{T-1}) = \max_{a_{T-1}} \left\{ R(s_{T-1}) + \gamma \sum_{s_T} P(s_T \mid s_{T-1}, a_{T-1}) R(s_T) \right\}$$

- The value at step t is$(0 \leq t \leq T)$

$$V(s_t) = \max_{a_t} \left\{ R(s_t) + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t) V(s_{t+1}) \right\}$$

Value iteration (VI)

- Iteratively update V(s) for each state until convergence

Value iteration (VI)

- Iteratively update V(s) for each state until convergence
- (Initialize V(s) for every state)

Value iteration (VI)

- Iteratively update V(s) for each state until convergence
- (Initialize V(s) for every state)
- For i=1,2,3,...
  - For s=1,...,S

$$V(s) = \max_{a} \left\{ R(s) + \gamma \sum_{s'} P(s' \mid s, a) V(s') \right\}$$

Value iteration (VI)

- Iteratively update V(s) for each state until convergence
- (Initialize V(s) for every state)
- For i=1,2,3,...
  - For s=1,...,S

$$\mathbf{V}(\mathbf{s}) = \max_{\mathbf{a}} \left\{ \mathbf{R}(\boldsymbol{s}) + \gamma \sum_{\mathbf{s}'} \mathbf{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) \mathbf{V}(\mathbf{s}') \right\}$$

- The policy is implicit
  - Once converged: $\boldsymbol{\pi}^*(\mathbf{s}) = \arg\max_{\mathbf{a}} \left\{ \mathbf{R}(\mathbf{s}) + \gamma \sum_{\mathbf{s}'} \mathbf{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) \mathbf{V}^*(\mathbf{s}') \right\} \; \forall \mathbf{s}$

Policy Iteration (PI)

- Improve policy explicitly.

Policy Iteration (PI)

- Improve policy explicitly.

Start with any (e.g., random) policy π

Policy Iteration (PI)

- Improve policy explicitly.

Start with any (e.g., random) policy π

Iterate until convergence:

1. Given current policy get the value of each state

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) V^{\pi}(s') \qquad \forall s$$

Policy Iteration (PI)

- Improve policy explicitly.

Start with any (e.g., random) policy π

Iterate until convergence:

1. Given current policy get the value of each state

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) V^{\pi}(s') \qquad \forall s$$

2. Update the current policy

$$\pi'(s) = \arg\max_{a} \left\{ R(s) + \gamma \sum_{s'} P(s' \mid s, a) V^{\pi}(s') \right\} \quad \forall s$$

Policy Iteration (PI)

- Improve policy explicitly.

Start with any (e.g., random) policy π

Iterate until convergence:

Policy Evaluation

1. Given current policy get the value of each state

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) V^{\pi}(s') \qquad \forall s$$

Policy Update

2. Update the current policy

$$\pi'(s) = \arg\max_{a} \left\{ R(s) + \gamma \sum_{s'} P(s' \mid s, a) V^{\pi}(s') \right\} \quad \forall s$$

- Demo of the PI algorithm in a deterministic environment

http://www.cs.toronto.edu/~lcharlin/courses/60629/reinforcejs/gridworld_dp.html