Agents as Processes

Agents carry out actions:

- forever infinite horizon
- until some stopping criteria is met indefinite horizon
- finite and fixed number of steps finite horizon

Decision-theoretic Planning

What should an agent do when

- it gets rewards (including punishments) and tries to maximize its rewards received

Decision-theoretic Planning

What should an agent do when

- it gets rewards (including punishments) and tries to maximize its rewards received
- actions can be stochastic; the outcome of an action can't be fully predicted

Decision-theoretic Planning

What should an agent do when

- it gets rewards (including punishments) and tries to maximize its rewards received
- actions can be stochastic; the outcome of an action can't be fully predicted
- there is a model that specifies the (probabilistic) outcome of actions and the rewards

Decision-theoretic Planning

What should an agent do when

- it gets rewards (including punishments) and tries to maximize its rewards received
- actions can be stochastic; the outcome of an action can't be fully predicted
- there is a model that specifies the (probabilistic) outcome of actions and the rewards
- the world is fully observable?

Initial Assumptions

- flat or modular or hierarchical
- explicit states or features or individuals and relations
- static or finite stage or indefinite stage or infinite stage
- fully observable or partially observable
- deterministic or stochastic dynamics
- goals or complex preferences
- single agent or multiple agents
- knowledge is given or knowledge is learned
- perfect rationality or bounded rationality

Utility and time

- Would you prefer \$1000 today or \$1000 next year?
- What price would you pay now to have an eternity of happiness?
- How can you trade off pleasures today with pleasures in the future?

Utility and time

- How would you compare the following sequences of rewards (per week):

  A: \$1000000, \$0, \$0, \$0, \$0, \$0,...
  B: \$1000, \$1000, \$1000, \$1000, \$1000,...
  C: \$1000, \$0, \$0, \$0, \$0,...
  D: \$1, \$1, \$1, \$1, \$1,...
  E: \$1, \$2, \$3, \$4, \$5,...

Rewards and Values

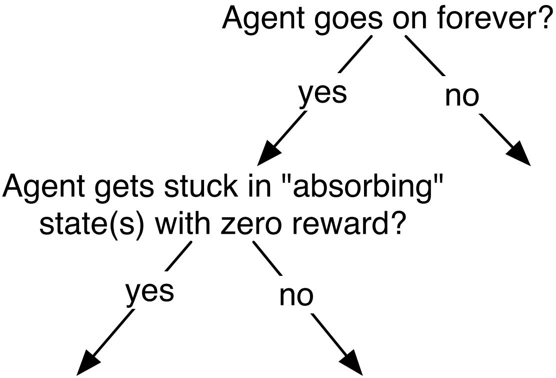
Suppose the agent receives a sequence of rewards $r_1, r_2, r_3, r_4, \ldots$ in time. What utility should be assigned? “Return” or “value”

Rewards and Values

Suppose the agent receives a sequence of rewards $r_1, r_2, r_3, r_4, \ldots$ in time. What utility should be assigned? “Return” or “value”

- total reward $V = \sum_{i=1}^{\infty} r_i$
- average reward $V = \lim_{n \to \infty} (r_1 + \cdots + r_n)/n$

Average vs Accumulated Rewards



Rewards and Values

Suppose the agent receives a sequence of rewards $r_1, r_2, r_3, r_4, \ldots$ in time.

- discounted return $V = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \cdots$
  γ is the discount factor $0 \leq \gamma \leq 1$.

Properties of the Discounted Rewards

- The discounted return for rewards $r_1, r_2, r_3, r_4, \ldots$ is

$$\begin{aligned} V &= r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \cdots \\ &= \end{aligned}$$

Properties of the Discounted Rewards

- The discounted return for rewards $r_1, r_2, r_3, r_4, \ldots$ is

$$\begin{aligned} V &= r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \cdots \\ &= r_1 + \gamma(r_2 + \gamma(r_3 + \gamma(r_4 + \ldots))) \end{aligned}$$

- If V_t is the value obtained from time step t

$$V_t =$$

Properties of the Discounted Rewards

- The discounted return for rewards $r_1, r_2, r_3, r_4, \ldots$ is
$$\begin{aligned} V &= r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \cdots \\ &= r_1 + \gamma(r_2 + \gamma(r_3 + \gamma(r_4 + \ldots))) \end{aligned}$$
- If V_t is the value obtained from time step t
$$V_t = r_t + \gamma V_{t+1}$$

Properties of the Discounted Rewards

- The discounted return for rewards $r_1, r_2, r_3, r_4, \ldots$ is

$$\begin{aligned} V &= r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \cdots \\ &= r_1 + \gamma(r_2 + \gamma(r_3 + \gamma(r_4 + \ldots))) \end{aligned}$$

- If V_t is the value obtained from time step t

$$V_t = r_t + \gamma V_{t+1}$$

- How is the infinite future valued compared to immediate rewards?

Properties of the Discounted Rewards

- The discounted return for rewards $r_1, r_2, r_3, r_4, \ldots$ is

$$\begin{aligned} V &= r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \cdots \\ &= r_1 + \gamma(r_2 + \gamma(r_3 + \gamma(r_4 + \ldots))) \end{aligned}$$

- If V_t is the value obtained from time step t

$$V_t = r_t + \gamma V_{t+1}$$

- How is the infinite future valued compared to immediate rewards?
  $1 + \gamma + \gamma^2 + \gamma^3 + \cdots = 1/(1 - \gamma)$
  Therefore $\dfrac{\text{minimum reward}}{1-\gamma} \leq V_t \leq \dfrac{\text{maximum reward}}{1-\gamma}$
- We can approximate V with the first k terms, with error:

$$V - (r_1 + \gamma r_2 + \cdots + \gamma^{k-1} r_k) = \gamma^k V_{k+1}$$

World State

- The world state is the information such that if the agent knew the world state, no information about the past is relevant to the future. Markovian assumption.
- S_i is state at time i, and A_i is the action at time i:

$$P(S_{t+1} \mid S_0, A_0, \ldots, S_t, A_t) =$$

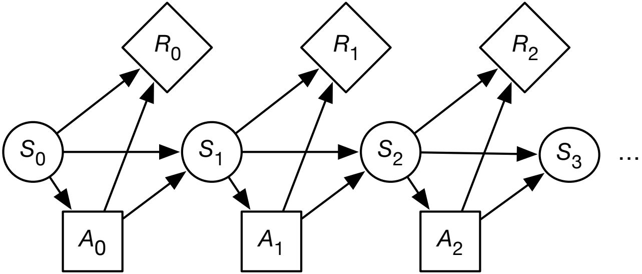
World State

- The world state is the information such that if the agent knew the world state, no information about the past is relevant to the future. Markovian assumption.
- S_i is state at time i, and A_i is the action at time i:

$$P(S_{t+1} \mid S_0, A_0, \ldots, S_t, A_t) = P(S_{t+1} \mid S_t, A_t)$$

  $P(s' \mid s, a)$ is the probability that the agent will be in state s' immediately after doing action a in state s.
- The dynamics is stationary if the distribution is the same for each time point.

- A Markov decision process augments a Markov chain with actions and rewards:



Markov Decision Processes

An MDP consists of:

- set S of states.
- set A of actions.

Markov Decision Processes

An MDP consists of:

- set S of states.
- set A of actions.
- $P(S_{t+1} \mid S_t, A_t)$ specifies the dynamics.

Markov Decision Processes

An MDP consists of:

- set S of states.
- set A of actions.
- $P(S_{t+1} \mid S_t, A_t)$ specifies the dynamics.
- $R(S_t, A_t)$ specifies the expected reward at time t.
  $R(s, a)$ is the expected reward of doing a in state s

Markov Decision Processes

An MDP consists of:

- set S of states.
- set A of actions.
- $P(S_{t+1} \mid S_t, A_t)$ specifies the dynamics.
- $R(S_t, A_t)$ specifies the expected reward at time t.
  $R(s, a)$ is the expected reward of doing a in state s
- γ is discount factor.

Example: to party or relax?

Each week *Sam* has to decide whether to party or relax:

- States: $\{healthy, sick\}$
- Actions: $\{relax, party\}$
- Dynamics:

Example: to party or relax?

Each week *Sam* has to decide whether to party or relax:

- States: $\{healthy, sick\}$
- Actions: $\{relax, party\}$
- Dynamics:

State	Action	$P(healthy \mid State, Action)$
healthy	relax	0.95
healthy	party	0.7
sick	relax	0.5
sick	party	0.1

Example: to party or relax?

Each week *Sam* has to decide whether to party or relax:

- States: $\{healthy, sick\}$
- Actions: $\{relax, party\}$
- Dynamics:

State	Action	$P(healthy \mid State, Action)$
healthy	relax	0.95
healthy	party	0.7
sick	relax	0.5
sick	party	0.1

- Reward:

Example: to party or relax?

Each week *Sam* has to decide whether to party or relax:

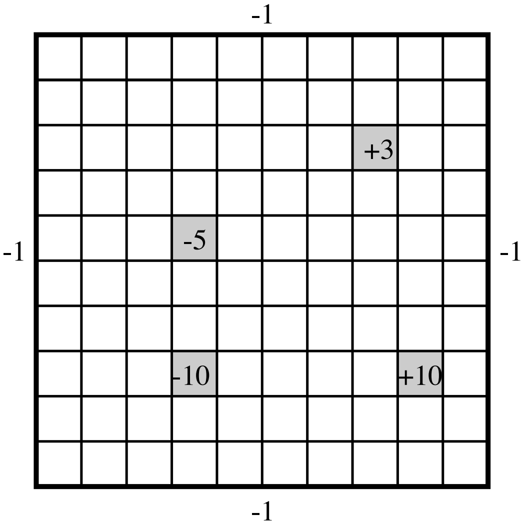
- States: $\{healthy, sick\}$
- Actions: $\{relax, party\}$
- Dynamics:

State	Action	$P(healthy \mid State, Action)$
healthy	relax	0.95
healthy	party	0.7
sick	relax	0.5
sick	party	0.1

- Reward:

State	Action	Reward
healthy	relax	7
healthy	party	10
sick	relax	0
sick	party	2

Example: Simple Grid World



Grid World Model

- Actions: up, down, left, right.
- 100 states corresponding to the positions of the robot.
- Robot goes in the commanded direction with probability 0.7, and one of the other directions with probability 0.1.
- If it crashes into an outside wall, it remains in its current position and has a reward of -1.
- Four special rewarding states; the agent gets the reward when leaving.

Planning Horizons

The planning horizon is how far ahead the planner looks to make a decision.

- The robot gets flung to one of the corners at random after leaving a positive ($+10$ or $+3$) reward state.
  - the process never halts
  - infinite horizon

Planning Horizons

The planning horizon is how far ahead the planner looks to make a decision.

- The robot gets flung to one of the corners at random after leaving a positive (+10 or +3) reward state.
  - the process never halts
  - infinite horizon
- The robot gets +10 or +3 in the state, then it stays there getting no reward. These are absorbing states.
  - The robot will eventually reach an absorbing state.
  - indefinite horizon

Information Availability

What information is available when the agent decides what to do?

- fully-observable MDP (FOMDP) the agent gets to observe S_t when deciding on action A_t.

Information Availability

What information is available when the agent decides what to do?

- fully-observable MDP (FOMDP) the agent gets to observe S_t when deciding on action A_t.
- partially-observable MDP (POMDP) the agent has some noisy sensor of the state. It is a mix of a hidden Markov model and MDP. It needs to remember (some function of) its sensing and acting history.

Information Availability

What information is available when the agent decides what to do?

- fully-observable MDP (FOMDP) the agent gets to observe S_t when deciding on action A_t.
- partially-observable MDP (POMDP) the agent has some noisy sensor of the state. It is a mix of a hidden Markov model and MDP. It needs to remember (some function of) its sensing and acting history.

[This lecture only considers FOMDPs.
POMDPS are much harder to solve.]

Policies

- A stationary policy is a function:

  $$\pi : S \to A$$

  Given a state s, $\pi(s)$ specifies what action the agent who is following π will do.

Policies

- A stationary policy is a function:

  $$\pi : S \rightarrow A$$

  Given a state s, $\pi(s)$ specifies what action the agent who is following π will do.
- An optimal policy is one with maximum expected discounted reward.

Policies

- A stationary policy is a function:

  $$\pi : S \rightarrow A$$

  Given a state s, $\pi(s)$ specifies what action the agent who is following π will do.
- An optimal policy is one with maximum expected discounted reward.
- An MDP with stationary dynamics and rewards with infinite or indefinite horizon, always has an stationary policy that is optimal.

Policies

- A stationary policy is a function:

  $$\pi : S \to A$$

  Given a state s, $\pi(s)$ specifies what action the agent who is following π will do.
- An optimal policy is one with maximum expected discounted reward.
- An MDP with stationary dynamics and rewards with infinite or indefinite horizon, always has an stationary policy that is optimal.
  (A randomized policy or a non-stationary policy is never better than a stationary policy.)

Example: to party or relax?

Each week *Sam* has to decide whether to party or relax each weekend:

- States: $\{healthy, sick\}$
- Actions: $\{relax, party\}$

How many stationary policies are there?

Example: to party or relax?

Each week *Sam* has to decide whether to party or relax each weekend:

- States: $\{healthy, sick\}$
- Actions: $\{relax, party\}$

How many stationary policies are there?
What are they?

Example: to party or relax?

Each week *Sam* has to decide whether to party or relax each weekend:

- States: $\{healthy, sick\}$
- Actions: $\{relax, party\}$

How many stationary policies are there?
What are they?

For the grid world with 100 states and 4 actions,
how many stationary policies are there?

Value of a Policy

Given a policy π:

- $Q^{\pi}(s, a)$, where a is an action and s is a state, is the expected value of doing a in state s, then following policy π.

Value of a Policy

Given a policy π:

- $Q^{\pi}(s, a)$, where a is an action and s is a state, is the expected value of doing a in state s, then following policy π.
- $V^{\pi}(s)$, where s is a state, is the expected value of following policy π in state s.

Value of a Policy

Given a policy π:

- $Q^\pi(s, a)$, where a is an action and s is a state, is the expected value of doing a in state s, then following policy π.
- $V^\pi(s)$, where s is a state, is the expected value of following policy π in state s.
- Q^π and V^π can be defined mutually recursively:

$$\begin{aligned} V^\pi(s) &= \\ Q^\pi(s, a) &= \end{aligned}$$

Value of a Policy

Given a policy π:

- $Q^\pi(s, a)$, where a is an action and s is a state, is the expected value of doing a in state s, then following policy π.
- $V^\pi(s)$, where s is a state, is the expected value of following policy π in state s.
- Q^π and V^π can be defined mutually recursively:

$$\begin{aligned} V^\pi(s) &= Q^\pi(s, \pi(s)) \\ Q^\pi(s, a) &= \end{aligned}$$

Value of a Policy

Given a policy π:

- $Q^\pi(s,a)$, where a is an action and s is a state, is the expected value of doing a in state s, then following policy π.
- $V^\pi(s)$, where s is a state, is the expected value of following policy π in state s.
- Q^π and V^π can be defined mutually recursively:

$$\begin{aligned} V^\pi(s) &= Q^\pi(s,\pi(s)) \\ Q^\pi(s,a) &= R(s,a) + \gamma \sum_{s'} P(s' \mid a,s) V^\pi(s') \end{aligned}$$

Value of the Optimal Policy

- $Q^*(s, a)$, where a is an action and s is a state, is the expected value of doing a in state s, then following the optimal policy.

Value of the Optimal Policy

- $Q^*(s, a)$, where a is an action and s is a state, is the expected value of doing a in state s, then following the optimal policy.
- $V^*(s)$, where s is a state, is the expected value of following the optimal policy in state s.

Value of the Optimal Policy

- $Q^*(s, a)$, where a is an action and s is a state, is the expected value of doing a in state s, then following the optimal policy.
- $V^*(s)$, where s is a state, is the expected value of following the optimal policy in state s.
- Q^* and V^* can be defined mutually recursively:

$$Q^*(s, a) =$$

$$V^*(s) =$$

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

Value of the Optimal Policy

- $Q^*(s, a)$, where a is an action and s is a state, is the expected value of doing a in state s, then following the optimal policy.
- $V^*(s)$, where s is a state, is the expected value of following the optimal policy in state s.
- Q^* and V^* can be defined mutually recursively:

$$
\begin{aligned}
Q^*(s, a) &= R(s, a) + \gamma \sum_{s'} P(s' \mid a, s) V^*(s') \\
V^*(s) &= \\
\pi^*(s) &=
\end{aligned}
$$

Value of the Optimal Policy

- $Q^*(s,a)$, where a is an action and s is a state, is the expected value of doing a in state s, then following the optimal policy.
- $V^*(s)$, where s is a state, is the expected value of following the optimal policy in state s.
- Q^* and V^* can be defined mutually recursively:

$$
\begin{aligned}
Q^*(s,a) &= R(s,a) + \gamma \sum_{s'} P(s' \mid a,s) V^*(s') \\
V^*(s) &= \max_a Q^*(s,a) \\
\pi^*(s) &=
\end{aligned}
$$

Value of the Optimal Policy

- $Q^*(s,a)$, where a is an action and s is a state, is the expected value of doing a in state s, then following the optimal policy.
- $V^*(s)$, where s is a state, is the expected value of following the optimal policy in state s.
- Q^* and V^* can be defined mutually recursively:

$$\begin{aligned} Q^*(s,a) &= R(s,a) + \gamma \sum_{s'} P(s' \mid a,s) V^*(s') \\ V^*(s) &= \max_a Q^*(s,a) \\ \pi^*(s) &= \arg\max_a Q^*(s,a) \end{aligned}$$

Value Iteration

- Let V_k and Q_k be k-step lookahead value and Q functions.

Value Iteration

- Let V_k and Q_k be k-step lookahead value and Q functions.
- Idea: Given an estimate of the k-step lookahead value function, determine the $k+1$ step lookahead value function.

Value Iteration

- Let V_k and Q_k be k-step lookahead value and Q functions.
- Idea: Given an estimate of the k-step lookahead value function, determine the $k+1$ step lookahead value function.
- Set V_0 arbitrarily.

Value Iteration

- Let V_k and Q_k be k-step lookahead value and Q functions.
- Idea: Given an estimate of the k-step lookahead value function, determine the $k+1$ step lookahead value function.
- Set V_0 arbitrarily.
- Compute Q_{i+1}, V_{i+1} from V_i.

Value Iteration

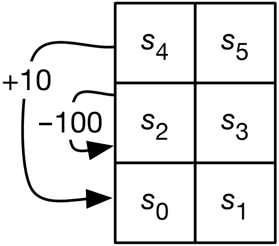
- Let V_k and Q_k be k-step lookahead value and Q functions.
- Idea: Given an estimate of the k-step lookahead value function, determine the $k+1$ step lookahead value function.
- Set V_0 arbitrarily.
- Compute Q_{i+1}, V_{i+1} from V_i.
- This converges exponentially fast (in k) to the optimal value function.

Value Iteration

- Let V_k and Q_k be k-step lookahead value and Q functions.
- Idea: Given an estimate of the k-step lookahead value function, determine the $k+1$ step lookahead value function.
- Set V_0 arbitrarily.
- Compute Q_{i+1}, V_{i+1} from V_i.
- This converges exponentially fast (in k) to the optimal value function.

  The error reduces proportionally to $\dfrac{\gamma^k}{1-\gamma}$

Tiny MDP Example: 6 states, 4 actions



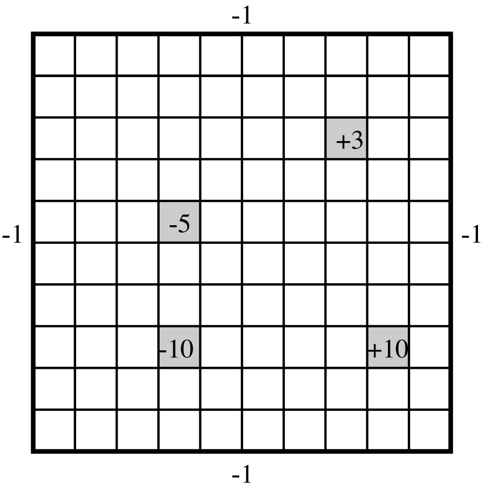
Actions (if crash, stay still with a reward of -1)

- ➡ move right
- ⬅ move left, except as above.
- ⬆ up risky: $P(up)$=0.8, $P(left)$=0.1, $P(right)$=0.1
- ⬇ up carefully: go up with extra reward of -1

AIPython: `python -i mdpExamples.py`

```
MDPtiny(discount=0.9).show()
```

Example: Simple Grid World



AIPython: `python -i mdpExamples.py`

```
grid(discount=0.9).show()
```

Asynchronous Value Iteration

- The agent doesn't need to sweep through all the states, but can update the value functions for each state individually.

Asynchronous Value Iteration

- The agent doesn't need to sweep through all the states, but can update the value functions for each state individually.
- This converges to the optimal value functions, if

Asynchronous Value Iteration

- The agent doesn't need to sweep through all the states, but can update the value functions for each state individually.
- This converges to the optimal value functions, if each state and action is visited infinitely often in the limit.

Asynchronous Value Iteration

- The agent doesn't need to sweep through all the states, but can update the value functions for each state individually.
- This converges to the optimal value functions, if each state and action is visited infinitely often in the limit.
- It can either store $V[s]$ or $Q[s, a]$.

Asynchronous VI: storing $V[s]$

- Repeat forever:
  - Select state s
  - $V[s] \leftarrow$

Asynchronous VI: storing $V[s]$

- Repeat forever:
  - Select state s
  - $V[s] \leftarrow \max_a \left(R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V[s'] \right)$

Asynchronous VI: storing $Q[s, a]$

- Repeat forever:
  - Select state s, action a
  - $Q[s, a] \leftarrow$

Asynchronous VI: storing $Q[s, a]$

- Repeat forever:
  - Select state s, action a
  - $Q[s, a] \leftarrow R(s, a) + \gamma \sum_{s'} P(s' \mid s, a) \left(\max_{a'} Q[s', a'] \right)$

Policy Iteration

- Set π_0 arbitrarily, let $i = 0$
- Repeat:
  - evaluate $Q^{\pi_i}(s, a)$
  - let $\pi_{i+1}(s) = argmax_a Q^{\pi_i}(s, a)$
  - set $i = i + 1$
- until $\pi_i(s) = \pi_{i-1}(s)$

Policy Iteration

- Set π_0 arbitrarily, let $i = 0$
- Repeat:
  - evaluate $Q^{\pi_i}(s, a)$
  - let $\pi_{i+1}(s) = argmax_a Q^{\pi_i}(s, a)$
  - set $i = i + 1$
- until $\pi_i(s) = \pi_{i-1}(s)$

Evaluating $Q^{\pi_i}(s, a)$ means finding a solution to a set of $|S| \times |A|$ linear equations with $|S| \times |A|$ unknowns.

It can also be approximated iteratively.

Modified Policy Iteration

Set $\pi[s]$ arbitrarily
Set $Q[s,a]$ arbitrarily
Repeat forever:

- Repeat for a while:
  - Select state s, action a
  - $Q[s,a] \leftarrow R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) Q[s', \pi[s']]$
- $\pi[s] \leftarrow argmax_a Q[s,a]$

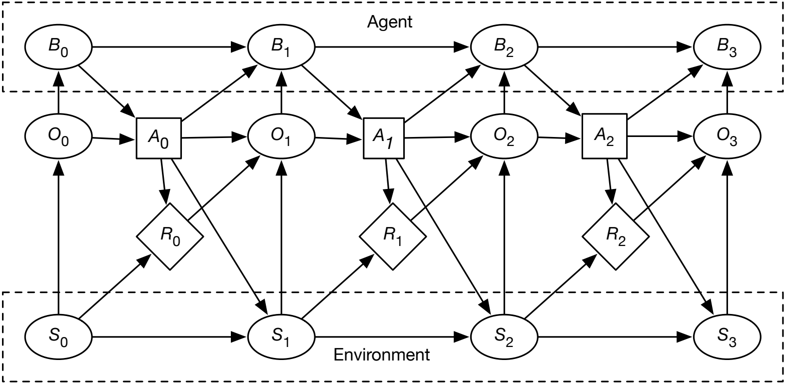
Q, V, π, R

$$Q^*(s,a) = \sum_{s'} P(s' \mid a, s) \left(R(s,a,s') + \gamma V^*(s')\right)$$
$$= R(s,a) + \gamma \sum_{s'} P(s' \mid a, s) V^*(s')$$
$$V^*(s) = \max_a Q^*(s,a)$$
$$\pi^*(s) = argmax_a Q^*(s,a)$$

where

$$R(s,a) = \sum_{s'} P(s' \mid a, s) R(s,a,s')$$

Partially Observable MDP (POMDP)



B_i agent's belief state at time i. A_i agent's action. O_i is what the agent observes. R_i is the reward. S_i is the world state.