

# 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 (and 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?

- 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

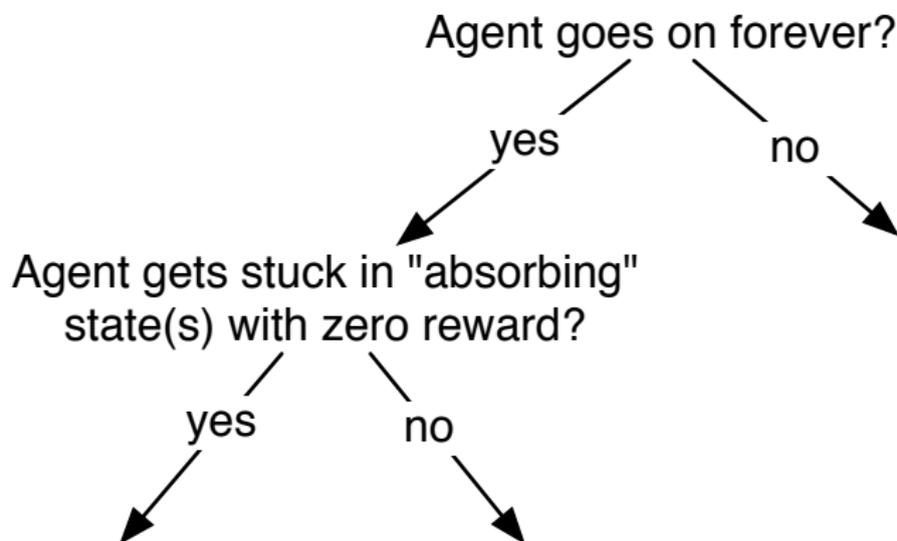
Suppose the agent receives a sequence of rewards  $r_1, r_2, r_3, r_4, \dots$  in time. What utility should be assigned?  
“Return” or “value”

# Rewards and Values

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

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

# Average vs Accumulated Rewards



# Rewards and Values

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

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

# Properties of the Discounted Rewards

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

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

# Properties of the Discounted Rewards

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

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

- If  $V_t$  is the value obtained from time step  $t$

$$V_t =$$

# Properties of the Discounted Rewards

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

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

- 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  $r_1, r_2, r_3, r_4, \dots$  is

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

- 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  $r_1, r_2, r_3, r_4, \dots$  is

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

- 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?

$$1 + \gamma + \gamma^2 + \gamma^3 + \dots = 1/(1 - \gamma)$$

$$\text{Therefore } \frac{\text{minimum reward}}{1 - \gamma} \leq V_t \leq \frac{\text{maximum reward}}{1 - \gamma}$$

- We can approximate  $V$  with the first  $k$  terms, with error:

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

# 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  $i$ :

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

# 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  $i$ :

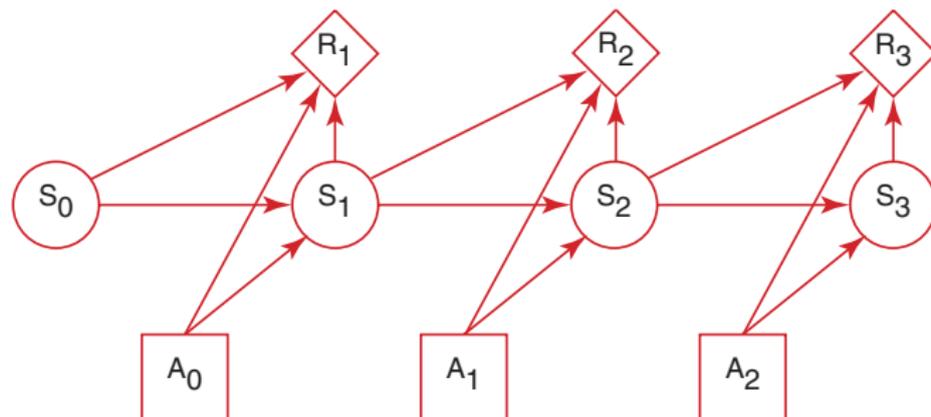
$$P(S_{t+1} \mid S_0, A_0, \dots, S_t, A_t) = P(S_{t+1} \mid S_t, A_t)$$

$P(s' \mid s, a)$  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.

# Decision Processes

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



# 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.
- $R(S_t, A_t, S_{t+1})$  specifies the reward at time  $t$ .  
 $R(s, a, s')$  is the expected reward received when the agent is in state  $s$ , does action  $a$  and ends up 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.
- $R(S_t, A_t, S_{t+1})$  specifies the reward at time  $t$ .  
 $R(s, a, s')$  is the expected reward received when the agent is in state  $s$ , does action  $a$  and ends up in state  $s'$ .  
Sometimes we use  $R(s, a) = \sum_{s'} P(s' \mid s, a)R(s, a, s')$ .

# 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.
- $R(S_t, A_t, S_{t+1})$  specifies the reward at time  $t$ .  
 $R(s, a, s')$  is the expected reward received when the agent is in state  $s$ , does action  $a$  and ends up in state  $s'$ .  
Sometimes we use  $R(s, a) = \sum_{s'} P(s' \mid s, a)R(s, a, s')$ .
- $\gamma$  is discount factor.

# Example: to exercise or not?

Each week *Sam* has to decide whether to exercise or not:

- States:  $\{fit, unfit\}$
- Actions:  $\{exercise, relax\}$
- Dynamics:

# Example: to exercise or not?

Each week *Sam* has to decide whether to exercise or not:

- States:  $\{fit, unfit\}$
- Actions:  $\{exercise, relax\}$
- Dynamics:

State	Action	$P(fit \mid State, Action)$
fit	exercise	0.99
fit	relax	0.7
unfit	exercise	0.2
unfit	relax	0.0

# Example: to exercise or not?

Each week *Sam* has to decide whether to exercise or not:

- States:  $\{fit, unfit\}$
- Actions:  $\{exercise, relax\}$
- Dynamics:

State	Action	$P(fit \mid State, Action)$
fit	exercise	0.99
fit	relax	0.7
unfit	exercise	0.2
unfit	relax	0.0

- Reward

# Example: to exercise or not?

Each week *Sam* has to decide whether to exercise or not:

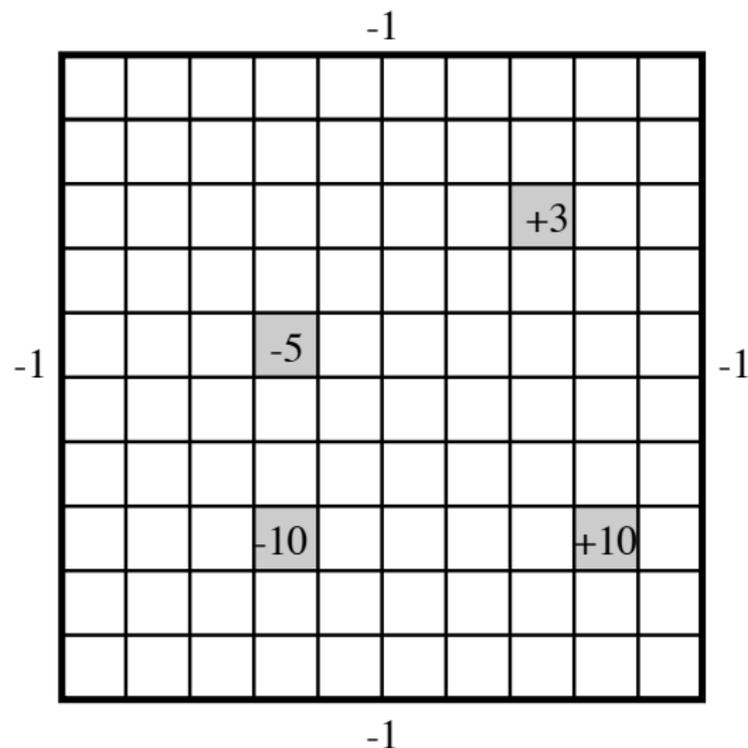
- States:  $\{fit, unfit\}$
- Actions:  $\{exercise, relax\}$
- Dynamics:

State	Action	$P(fit \mid State, Action)$
fit	exercise	0.99
fit	relax	0.7
unfit	exercise	0.2
unfit	relax	0.0

- Reward (does not depend on resulting state):

State	Action	Reward
fit	exercise	8
fit	relax	10
unfit	exercise	0
unfit	relax	5

# 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** the agent gets to observe  $S_t$  when deciding on action  $A_t$ .

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

- **fully-observable MDP** 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.

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

- **fully-observable MDP** 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]

- A **stationary policy** is a function:

$$\pi : S \rightarrow A$$

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

- A **stationary policy** is a function:

$$\pi : S \rightarrow A$$

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

- An **optimal policy** is one with maximum expected discounted reward.

- A **stationary policy** is a function:

$$\pi : S \rightarrow A$$

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

- An **optimal policy** is one with maximum expected discounted reward.
- For a fully-observable MDP with stationary dynamics and rewards with infinite or indefinite horizon, there is always an optimal stationary policy.

# Example: to exercise or not?

Each week *Sam* has to decide whether to exercise or not:

- States:  $\{fit, unfit\}$
- Actions:  $\{exercise, relax\}$

How many stationary policies are there?

# Example: to exercise or not?

Each week *Sam* has to decide whether to exercise or not:

- States:  $\{fit, unfit\}$
- Actions:  $\{exercise, relax\}$

How many stationary policies are there?

What are they?

# Example: to exercise or not?

Each week *Sam* has to decide whether to exercise or not:

- States:  $\{fit, unfit\}$
- Actions:  $\{exercise, relax\}$

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  $\pi$ :

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

# Value of a Policy

Given a policy  $\pi$ :

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

# Value of a Policy

Given a policy  $\pi$ :

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

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

# 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 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  $\frac{\gamma^k}{1 - \gamma}$

# 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  $s$

- ▶  $V[s] \leftarrow \max_a \sum_{s'} P(s' | s, a) (R(s, a, s') + \gamma V[s'])$

# 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  $a$

- ▶  $Q[s, a] \leftarrow \sum_{s'} P(s' | s, a) \left( R(s, a, s') + \gamma \max_{a'} Q[s', a'] \right)$

# Policy Iteration

- Set  $\pi_0$  arbitrarily, let  $i = 0$
- Repeat:
  - ▶ evaluate  $Q^{\pi_i}(s, a)$
  - ▶ let  $\pi_{i+1}(s) = \operatorname{argmax}_a Q^{\pi_i}(s, a)$
  - ▶ set  $i = i + 1$
- until  $\pi_i(s) = \pi_{i-1}(s)$

# Policy Iteration

- Set  $\pi_0$  arbitrarily, let  $i = 0$
- Repeat:
  - ▶ evaluate  $Q^{\pi_i}(s, a)$
  - ▶ let  $\pi_{i+1}(s) = \operatorname{argmax}_a Q^{\pi_i}(s, a)$
  - ▶ 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  $a$
  - ▶  $Q[s, a] \leftarrow \sum_{s'} P(s' | s, a) (R(s, a, s') + \gamma Q[s', \pi[s']])$
- $\pi[s] \leftarrow \operatorname{argmax}_a Q[s, a]$

$$Q^*(s, a) = \sum_{s'} P(s' | a, s) (R(s, a, s') + \gamma V^*(s'))$$

$$V^*(s) = \max_a Q(s, a)$$

$$\pi^*(s) = \operatorname{argmax}_a Q(s, a)$$

Let

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

Then:

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

$$Q^*(s, a) = \sum_{s'} P(s' | a, s) (R(s, a, s') + \gamma V^*(s'))$$

$$V^*(s) = \max_a Q(s, a)$$

$$\pi^*(s) = \operatorname{argmax}_a Q(s, a)$$

Let

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

Then:

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s'} P(s' | a, s) V^*(s')$$