CS 540 Introduction to Artificial Intelligence

Reinforcement Learning II

University of Wisconsin-Madison

Spring 2023

Announcements

Assignments:

- Homework 10 due Thursday May 4
- Complete course evaluations by Friday May 5

Class roadmap:

Thursday, April 27	Reinforcement Learning I
Tuesday, May 2	Advanced Search
Thursday, May 4	Ethics and Trust in AI

Final Exam: May 12 5:05 - 7:05 pm

Outline

Outline

- Review of reinforcement learning setting.

Outline

- Review of reinforcement learning setting.
  - MDPs, value functions, Q-learning

Outline

- Review of reinforcement learning setting.
  - MDPs, value functions, Q-learning
- Bellman equations and dynamic programming

Outline

- Review of reinforcement learning setting.
  - MDPs, value functions, Q-learning
- Bellman equations and dynamic programming
- From dynamic programming to Q-learning

Key Ideas in Reinforcement Learning

Define RL Problem

States, Actions, Transitions, Rewards, Markov property, discounting

Value Functions

Bellman Equation

Writing the value of one state in terms of successor states.

Using values to choose optimal actions.

Q-learning

Value Iteration

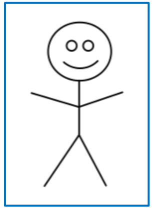
Exploration vs. Exploitation

Back to Our General Model

We have an **agent** **interacting** with the **world**

Back to Our General Model

We have an **agent** **interacting** with the **world**



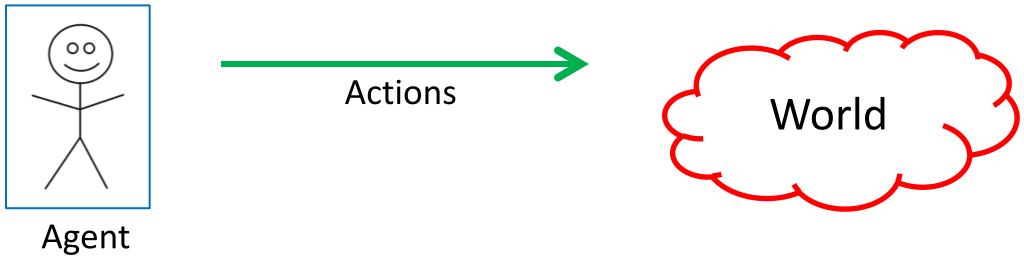
Agent

Back to Our General Model

We have an **agent** **interacting** with the **world**

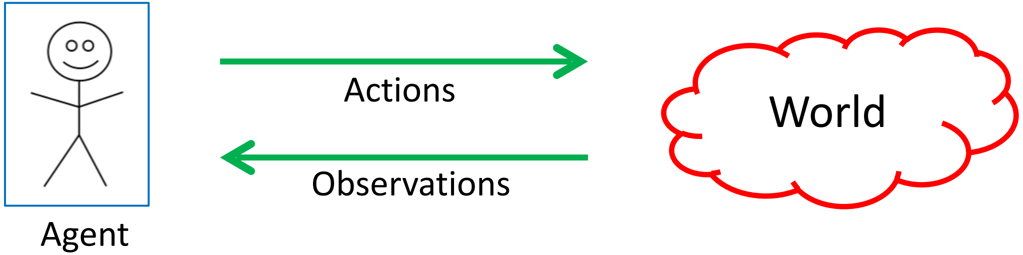
Back to Our General Model

We have an **agent** **interacting** with the **world**



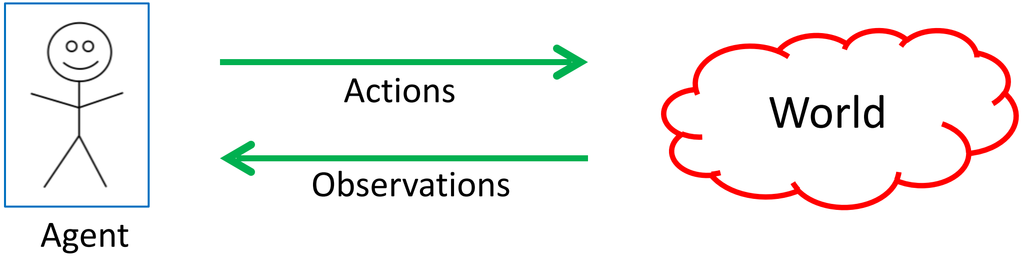
Back to Our General Model

We have an **agent** **interacting** with the **world**



Back to Our General Model

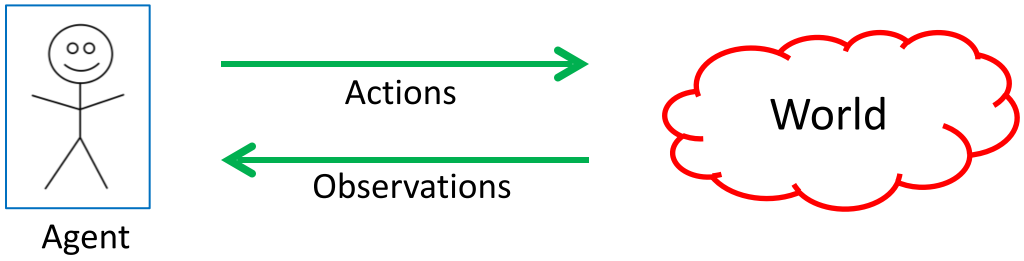
We have an **agent** **interacting** with the **world**



- Agent receives a reward based on state of the world

Back to Our General Model

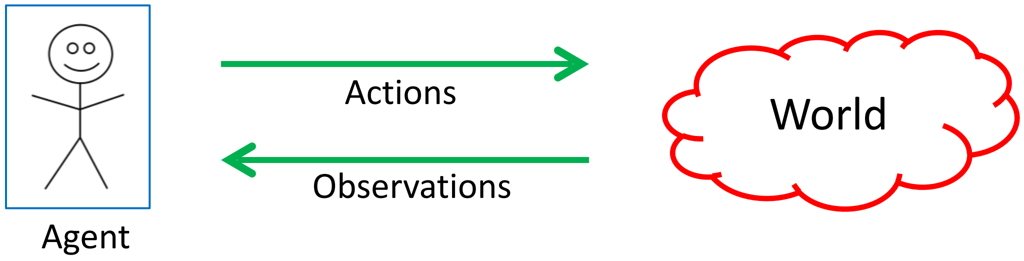
We have an **agent** **interacting** with the **world**



- Agent receives a reward based on state of the world
  - **Goal**: maximize reward / utility

Back to Our General Model

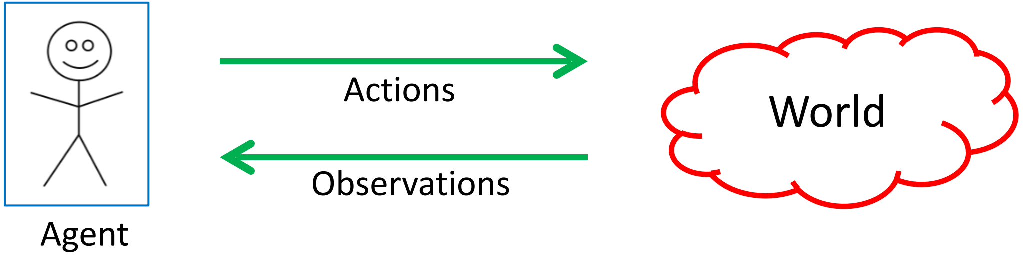
We have an **agent interacting** with the **world**



- Agent receives a reward based on state of the world
  - **Goal**: maximize reward / utility **($$$)**

Back to Our General Model

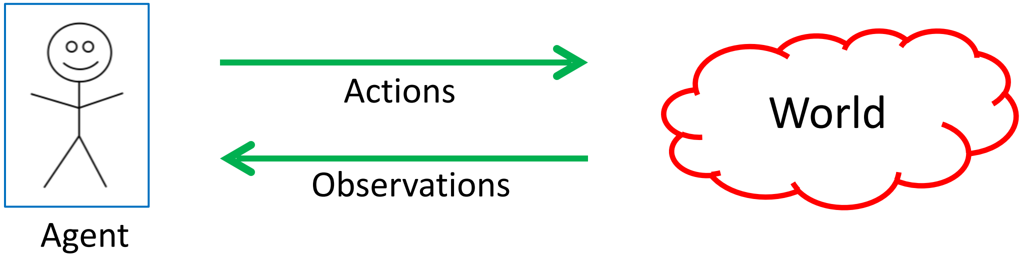
We have an **agent** **interacting** with the **world**



- Agent receives a reward based on state of the world
  - **Goal**: maximize reward / utility **($$$)**
  - Note: **data** consists of actions & observations

Back to Our General Model

We have an **agent** **interacting** with the **world**



- Agent receives a reward based on state of the world
  - **Goal**: maximize reward / utility **($$$)**
  - Note: **data** consists of actions & observations
    - Compare to unsupervised learning and supervised learning

Markov Decision Process (MDP)

Markov Decision Process (MDP)

The formal mathematical model:

Markov Decision Process (MDP)

The formal mathematical model:

- **State set** S. Initial state $s_{0.}$ **Action set** A

Markov Decision Process (MDP)

The formal mathematical model:

- **State set** S. Initial state $s_{0.}$ **Action set** A
- **State transition model:** $P(s_{t+1}|s_t, a_t)$

Markov Decision Process (MDP)

The formal mathematical model:

- **State set** S. Initial state $s_{0.}$ **Action set** A
- **State transition model:** $P(s_{t+1} | s_t, a_t)$
  - Markov assumption: transition probability only depends on s_t and a_t, and not previous actions or states.

Markov Decision Process (MDP)

The formal mathematical model:

- **State set** S. Initial state s_0. **Action set** A
- **State transition model:** $P(s_{t+1}|s_t, a_t)$
  - Markov assumption: transition probability only depends on s_t and a_t, and not previous actions or states.
- **Reward function:** $r(s_t)$

Markov Decision Process (MDP)

The formal mathematical model:

- **State set** S. Initial state s_0. **Action set** A
- **State transition model:** $P(s_{t+1}|s_t, a_t)$
  - Markov assumption: transition probability only depends on s_t and a_t, and not previous actions or states.
- **Reward function:** $r(s_t)$
- **Policy**: $\pi(s) : S \rightarrow A$, action to take at a particular state.

Markov Decision Process (MDP)

The formal mathematical model:

- **State set** S. Initial state s_0. **Action set** A
- **State transition model:** $P(s_{t+1} | s_t, a_t)$
  - Markov assumption: transition probability only depends on s_t and a_t, and not previous actions or states.
- **Reward function:** $r(s_t)$
- **Policy**: $\pi(s) : S \to A$, action to take at a particular state.

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \dots$$

Defining the Optimal Policy

For policy π, **expected utility** over all possible state sequences from s_0 produced by following that policy:

$$V^{\pi}(s_0) = \sum_{\substack{\text{sequences} \\ \text{starting from } s_0}} P(\text{sequence})U(\text{sequence})$$

Called the **value function** (for π, s_0)

Discounting Rewards

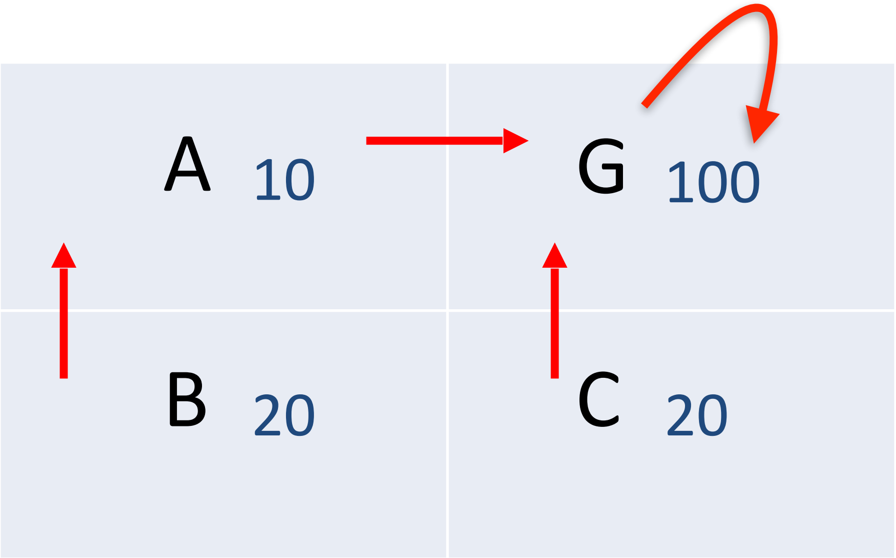
One issue: these are infinite series. **Convergence**?

- Solution

$$U(s_0, s_1 \ldots) = r(s_0) + \gamma r(s_1) + \gamma^2 r(s_2) + \ldots = \sum_{t \geq 0} \gamma^t r(s_t)$$

- Discount factor γ between 0 and 1
  - Set according to how important **present** is VS **future**
  - Note: has to be less than 1 for convergence

Example



Deterministic transitions; $\gamma = 0.8$; policy shown with red arrows.

Values and Policies

- Now that $V^{\pi}(s_0)$ is defined what a should we take?
  - First, set V*(s) to be expected utility for **optimal** policy from s
  - What's the expected utility of an action?
    - Specifically, action a in state s?

Values and Policies

- Now that $V^{\pi}(s_0)$ is defined what a should we take?
  - First, set V*(s) to be expected utility for **optimal** policy from s
  - What's the expected utility of an action?
    - Specifically, action a in state s?

$$\sum_{s'} P(s'|s, a)V^*(s')$$

Values and Policies

- Now that $V^{\pi}(s_0)$ is defined what a should we take?
  - First, set V*(s) to be expected utility for **optimal** policy from s
  - What's the expected utility of an action?
    - Specifically, action a in state s?

$$\sum_{s'} P(s'|s,a)V^*(s')$$

All the states we could go to

Values and Policies

- Now that $V^{\pi}(s_0)$ is defined what a should we take?
  - First, set V*(s) to be expected utility for **optimal** policy from s
  - What's the expected utility of an action?
    - Specifically, action a in state s?

$$\sum_{s'} P(s'|s,a)V^*(s')$$

All the states we could go to

Transition probability

Values and Policies

- Now that $V^{\pi}(s_0)$ is defined what a should we take?
  - First, set V*(s) to be expected utility for **optimal** policy from s
  - What's the expected utility of an action?
    – Specifically, action a in state s?

$$\sum_{s'} P(s'|s,a)V^*(s')$$

All the states we could go to

Transition probability

Expected rewards

Obtaining the Optimal Policy

Assume, we know the expected utility of an action.

- So, to get the optimal policy, compute

Obtaining the Optimal Policy

Assume, we know the expected utility of an action.

- So, to get the optimal policy, compute

$$\pi^*(s) = \text{argmax}_a \sum_{s'} P(s'|s, a)V^*(s')$$

All the states we could go to

Transition probability

Expected rewards

Obtaining the Optimal Policy

Assume, we know the expected utility of an action.

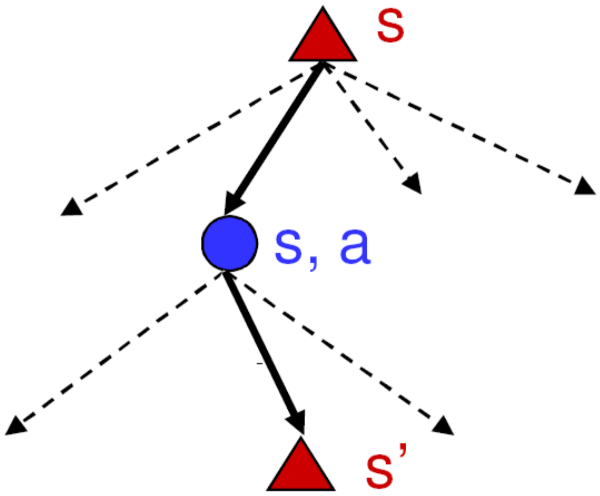
- So, to get the optimal policy, compute

$$\pi^*(s) = \text{argmax}_a \sum_{s'} P(s'|s,a)V^*(s')$$

All the states we could go to

Transition probability

Expected rewards



Credit L. Lazbenik

Bellman Equations

Let’s walk over one step for the value function:

Bellman Equations

Let's walk over one step for the value function:

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

Bellman Equations

Let's walk over one step for the value function:

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

Current state reward

Bellman Equations

Let's walk over one step for the value function:

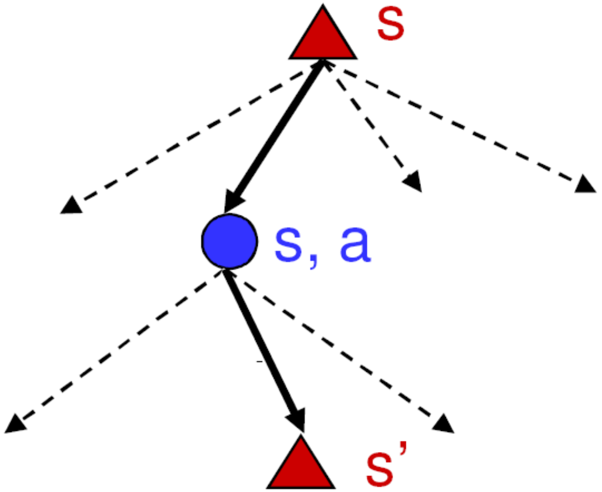
$$V^*(s) = \underbrace{r(s)}_{\text{Current state reward}} + \underbrace{\gamma \max_a \sum_{s'} P(s'|s,a)V^*(s')}_{\text{Discounted expected future rewards}}$$

Current state reward

Discounted expected future **rewards**

Bellman Equations

Let’s walk over one step for the value function:



Credit L. Lazbenik

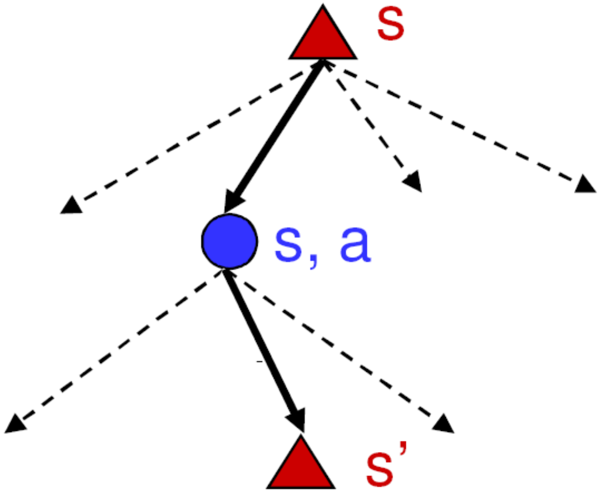
$$V^*(s) = r(s) + \gamma \max_a \underbrace{\sum_{s'} P(s'|s,a) V^*(s')}_{\text{Discounted expected future rewards}}$$

Current state reward

Discounted expected future **rewards**

Bellman Equations

Let’s walk over one step for the value function:



Credit L. Lazbenik

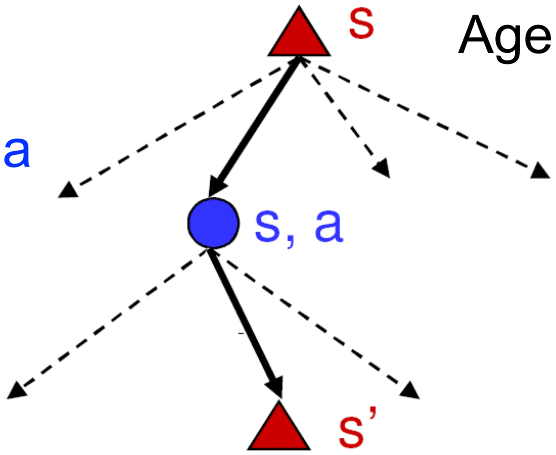
$$V^*(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s,a)V^*(s')$$

Current state reward

Discounted expected future **rewards**

Richard Bellman: Inventor of dynamic programming.

The Bellman equation



Agent receives reward r(s)

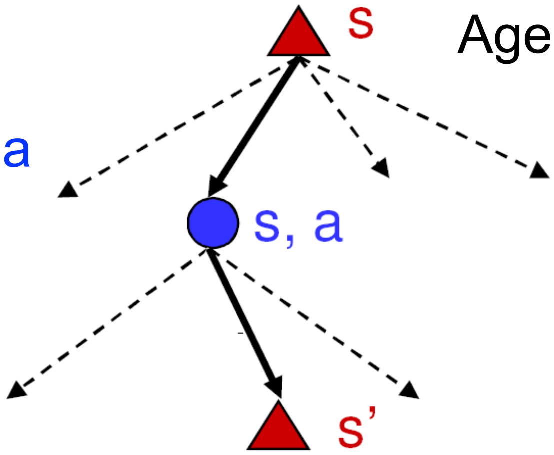
Agent chooses action a

Environment returns
$s' \sim P(\cdot \mid s, a)$

Define state value $V^*(s)$ as the expected sum of discounted rewards if the agent follows an *optimal* policy starting in state s.

Image source: L. Lazbenik

The Bellman equation



Agent receives reward r(s)

Agent chooses action a

Environment returns

$s' \sim P(\cdot \mid s, a)$

- What is the expected utility of taking action **a** in state **s**?

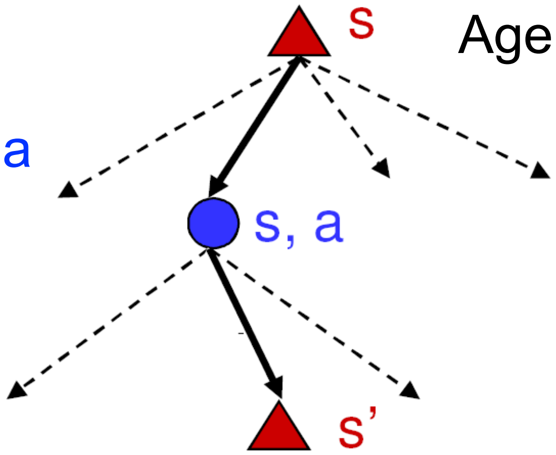
$$\sum_{s'} P(s'|s,a)V^*(s')$$

Image source: L. Lazbenik

The Bellman equation

Agent receives reward r(s)

Agent chooses action a



Environment returns

$s' \sim P(\cdot \mid s, a)$

- What is the recursive expression for $V^*(s)$ in terms of $V^*(s')$ - the utilities of its successors?

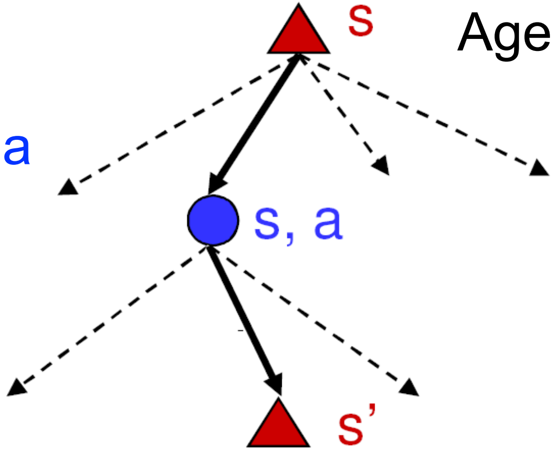
$$V^*(s) = r(s) + \gamma \sum_{s'} P(s'|s, \pi^*(s))V^*(s')$$

Image source: L. Lazbenik

The Bellman equation

Agent receives reward r(s)

Agent chooses action a



Environment returns

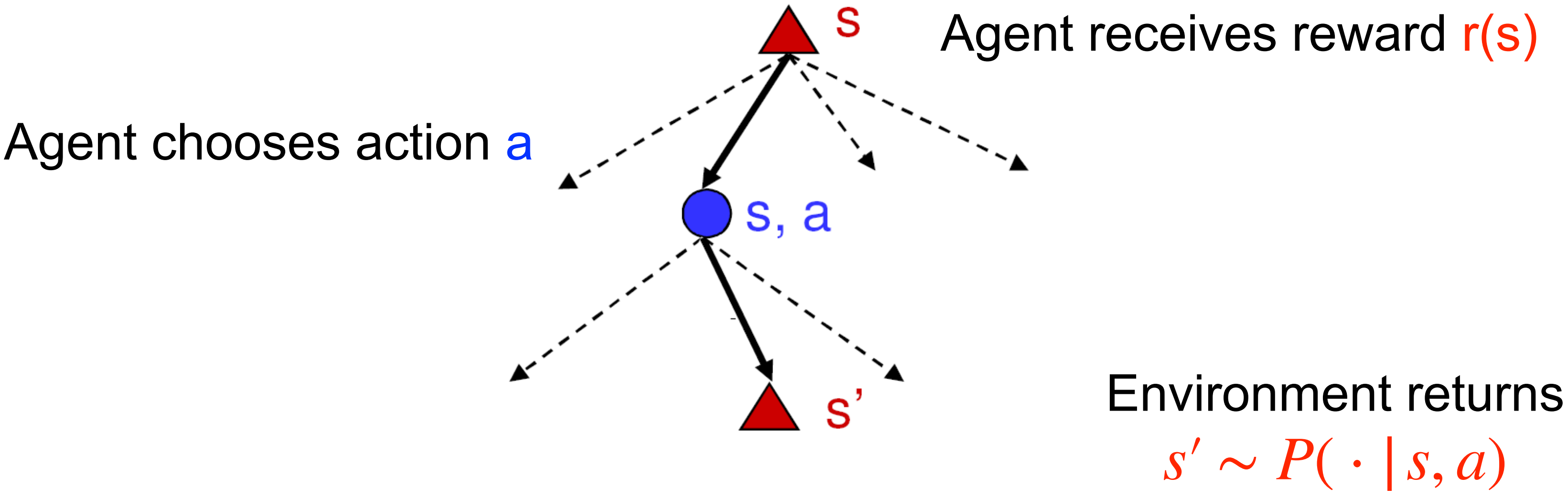
$$s' \sim P(\cdot \mid s, a)$$

- How do we choose the action?

$$\pi^*(s) = \arg\max_a \sum_{s'} P(s'|s,a) V^*(s')$$

Image source: L. Lazbenik

The Bellman equation



- What is the recursive expression for $V^*(s)$ in terms of $V^*(s')$ - the utilities of its successors?

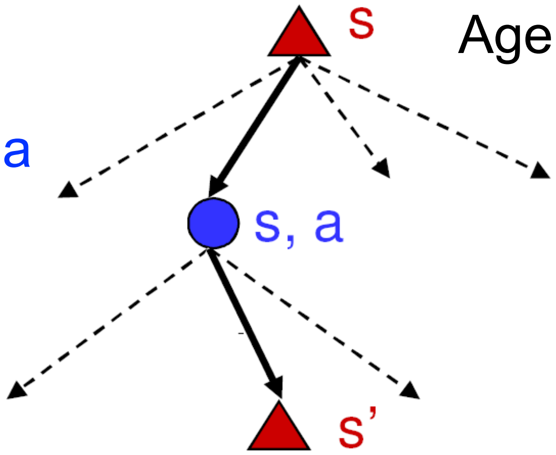
$$V^*(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s,a) V^*(s')$$

Image source: L. Lazbenik

The Bellman equation

Agent receives reward r(s)

Agent chooses action a



Environment returns

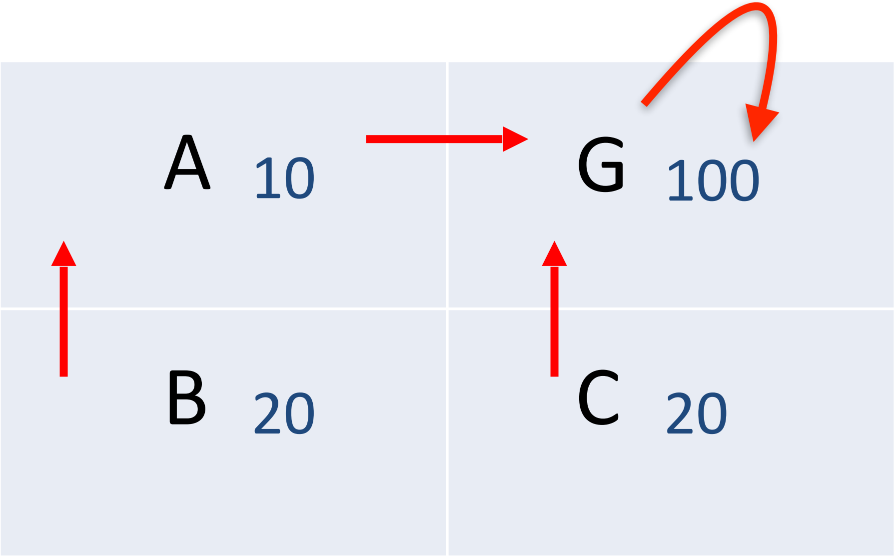
$$s' \sim P(\cdot \mid s, a)$$

- The same reasoning gives the Bellman equation for a general policy:

$$V^{\pi}(s) = r(s) + \gamma \sum_{s'} P(s'|s, \pi(s))V^{\pi}(s')$$

Image source: L. Lazbenik

Example



Deterministic transitions; $\gamma = 0.8$; policy shown with red arrows.

Value Iteration

Value Iteration

Q: how do we find $V^*(s)$?

Value Iteration

Q: how do we find $V^*(s)$?

- Why do we want it? Can use it to get the best policy

Value Iteration

Q: how do we find $V^*(s)$?

- Why do we want it? Can use it to get the best policy
- Know: reward $r(s)$, transition probability $P(s'|s,a)$

Value Iteration

Q: how do we find $V^*(s)$?

- Why do we want it? Can use it to get the best policy
- Know: reward $r(s)$, transition probability $P(s'|s,a)$
- Also know $V^*(s)$ satisfies Bellman equation:

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

Value Iteration

Q: how do we find $V^*(s)$?

- Why do we want it? Can use it to get the best policy
- Know: reward $r(s)$, transition probability $P(s'|s,a)$
- Also know $V^*(s)$ satisfies Bellman equation:

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

A: Use the property. Start with $V_0(s)=0$. Then, update

$$V_{i+1}(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s,a) V_i(s')$$

Value Iteration Algorithm

Input: Transition function P, reward function r, precision $\delta > 0$

1. For all states s, set V(s) = 0.
2. $\Delta \leftarrow \infty$
3. While $\Delta > \delta$:
4. Loop for each state s:
5. $V(s) \leftarrow r(s) + \max_a \gamma \sum_{s'} P(s' | s, a) V(s')$
6. $\Delta \leftarrow$ maximum change in $V(s)$ for any state s
7. End Loop
8. End While

Value Iteration Algorithm

Input: Transition function P, reward function r, precision $\delta > 0$

1. For all states s, set V(s) = 0.
2. $\Delta \leftarrow \infty$
3. While $\Delta > \delta$:
4. Loop for each state s:
5. $$V(s) \leftarrow r(s) + \max_a \gamma \sum_{s'} P(s' | s, a) V(s')$$
6. $\Delta \leftarrow$ maximum change in $V(s)$ for any state s
7. End Loop
8. End While

Here, P and r are known so no need for exploration or interaction with real world.

Value Iteration Demo

https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html

Break & Quiz

Q 2.1 Consider an MDP with 2 states {*A, B*} and 2 actions: **"stay"** at current state and "**move**" to other state. Let **r** be the reward function such that **r**(*A*) = 1, **r**(*B*) = 0. Let γ be the discounting factor. Let π: π(*A*) = π(*B*) = **move** (i.e., an "always move" policy). What is the value function $V^\pi(A)$?

- A. 0
- B. 1 / (1 -γ)
- C. 1 / (1 -γ^2)
- D. 1

Break & Quiz

Q 2.1 Consider an MDP with 2 states {*A*, *B*} and 2 actions: **"stay"** at current state and "**move**" to other state. Let **r** be the reward function such that **r**(*A*) = 1, **r**(*B*) = 0. Let γ be the discounting factor. Let π: π(*A*) = π(*B*) = **move** (i.e., an "always move" policy). What is the value function $V^{\pi}(A)$?

- A. 0
- B. $1/(1-\gamma)$
- **C. $1/(1-\gamma^2)$**
- D. 1

Break & Quiz

Q 2.1 Consider an MDP with 2 states {*A*, *B*} and 2 actions: **"stay"** at current state and "**move**" to other state. Let **r** be the reward function such that **r**(*A*) = 1, **r**(*B*) = 0. Let γ be the discounting factor. Let π: π(*A*) = π(*B*) = **move** (i.e., an "always move" policy). What is the value function $V^{\pi}(A)$?

- A. 0
- B. $1/(1-\gamma)$
- **C. $1/(1-\gamma^2)$** (States: A,B,A,B,… rewards 1,0, γ^2,0, γ^4,0, …)
- D. 1

Q-Learning

Q-Learning

Q-Learning

- Reinforcement learning without knowledge of r or P

Q-Learning

- Reinforcement learning without knowledge of r or P
- Learn from data of the form: $\{(s_t, a_t, r_t, s_{t+1})\}$.

Q-Learning

- Reinforcement learning without knowledge of r or P
- Learn from data of the form: $\{(s_t, a_t, r_t, s_{t+1})\}$.
- Learns an action-value function Q*(s,a) that tells us the expected value of taking a in state s.

Q-Learning

- Reinforcement learning without knowledge of r or P
- Learn from data of the form: $\{(s_t, a_t, r_t, s_{t+1})\}$.
- Learns an action-value function Q*(s,a) that tells us the expected value of taking a in state s.
  - Note: $V^*(s) = \max_a Q^*(s, a)$.

Q-Learning

- Reinforcement learning without knowledge of r or P
- Learn from data of the form: $\{(s_t, a_t, r_t, s_{t+1})\}$.
- Learns an action-value function Q*(s,a) that tells us the expected value of taking a in state s.
  - Note: $V^*(s) = \max_a Q^*(s, a)$.
- Optimal policy is formed as $\pi^*(s) = \arg\max_a Q^*(s, a)$

Q-Learning

Estimate Q*(*s*,*a*) from data $\{(s_t, a_t, r_t, s_{t+1})\}$:

1. Initialize Q(.,.) arbitrarily (eg all zeros)
   1. Except terminal states $Q(s_{terminal},.)=0$
2. Iterate over data until Q(.,.) converges:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_b Q(s_{t+1}, b))$$

Q-Learning

Estimate Q*(s,a) from data $\{(s_t, a_t, r_t, s_{t+1})\}$:

1. Initialize Q(.,.) arbitrarily (eg all zeros)
   1. Except terminal states Q($s_{terminal}$,.)=0
2. Iterate over data until Q(.,.) converges:

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_b Q(s_{t+1}, b))$$

Learning rate

Q-Learning

Estimate Q*(s,a) from data $\{(s_t, a_t, r_t, s_{t+1})\}$:

1. Initialize Q(.,.) arbitrarily (eg all zeros)
   1. Except terminal states $Q(s_{terminal},.)=0$
2. Iterate over data until Q(.,.) converges:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_b Q(s_{t+1}, b))$$

Learning rate

Equivalent update: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r(s_t) + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$

Q-learning Algorithm

Input: step size α, exploration probability ϵ

1. set Q(s,a) = 0 for all s, a.
2. For each episode:
3. Get initial state s.
4. While (s not a terminal state):
5. Perform a = ϵ-greedy(Q, s), receive r, s'
6. $Q(s,a) = (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a'))$
7. $s \leftarrow s'$
8. End While
9. End For

Q-learning Algorithm

Input: step size α, exploration probability ϵ

1. set Q(s,a) = 0 for all s, a.
2. For each episode:
3. Get initial state s.
4. While (s not a terminal state):
5. Perform a = ϵ-greedy(Q, s), receive r, s'
6. $Q(s,a) = (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a'))$
7. $s \leftarrow s'$
8. End While
9. End For

Explore: take action to see what happens.

Q-learning Algorithm

Input: step size α, exploration probability ϵ

1. set Q(s,a) = 0 for all s, a.
2. For each episode:
3. Get initial state s.
4. While (s not a terminal state):
5. Perform a = ϵ-greedy(Q, s), receive r, s'
6. $Q(s,a) = (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a'))$
7. $s \leftarrow s'$
8. End While
9. End For

Explore: take action to see what happens.

Update action-value based on result.

Q-learning Algorithm

Input: step size α, exploration probability ϵ

1. set Q(s,a) = 0 for all s, a.
2. For each episode:
3. Get initial state s.
4. While (s not a terminal state):
5. Perform a = ϵ-greedy(Q, s), receive r, s'
6. $Q(s,a) = (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a'))$
7. $s \leftarrow s'$
8. End While
9. End For

Explore: take action to see what happens.

Update action-value based on result.

Converges to Q*(s,a) in limit if all states and actions visited infinitely often.