CS 540 Introduction to Artificial Intelligence
Reinforcement Learning I

University of Wisconsin-Madison

Spring 2023

Announcements

Homeworks:

- Homework 9 due Thursday April 27
- Homework 10 due Thursday May 4

Course Evaluation:

- Complete by Friday May 5

Class roadmap:

Tuesday, April 25	Reinforcement Learning I
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

- Introduction to reinforcement learning
  - Basic concepts, mathematical formulation, MDPs, policies.

Outline

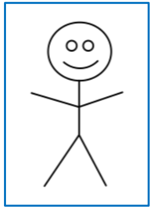
- Introduction to reinforcement learning
  - Basic concepts, mathematical formulation, MDPs, policies.
- Learning policies
  - Q-learning, action-values, exploration vs exploitation.

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**



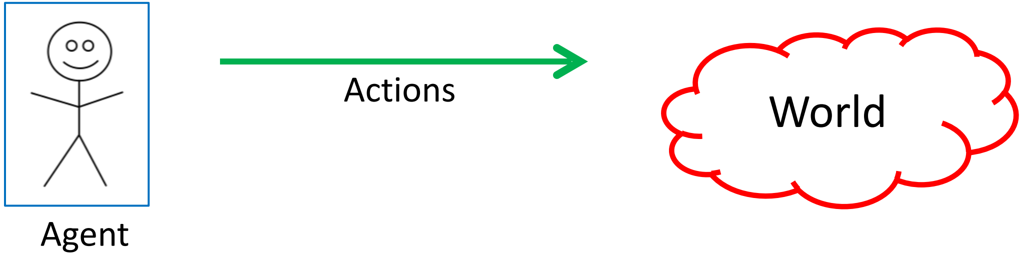
Agent

Back to Our General Model

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

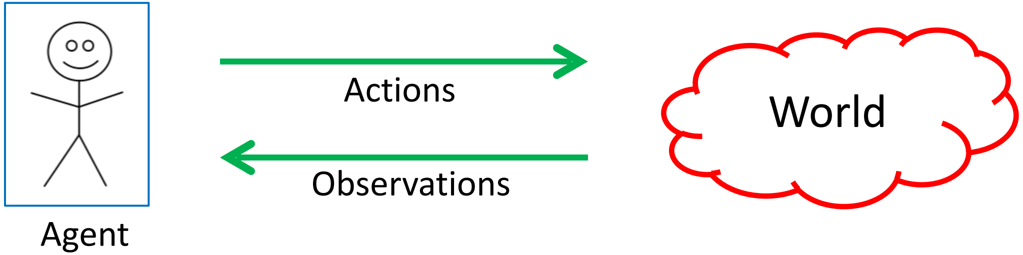
Back to Our General Model

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



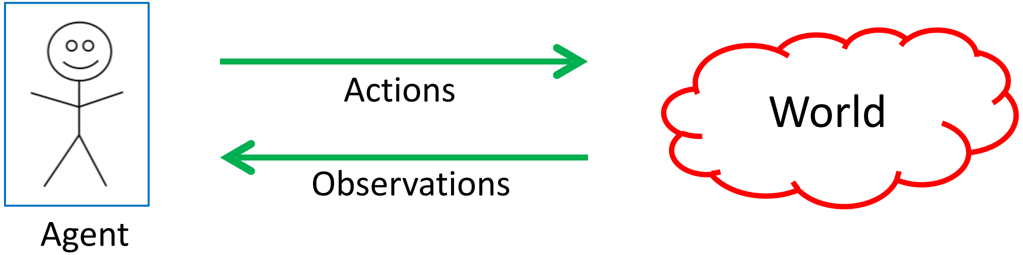
Back to Our General Model

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



Back to Our General Model

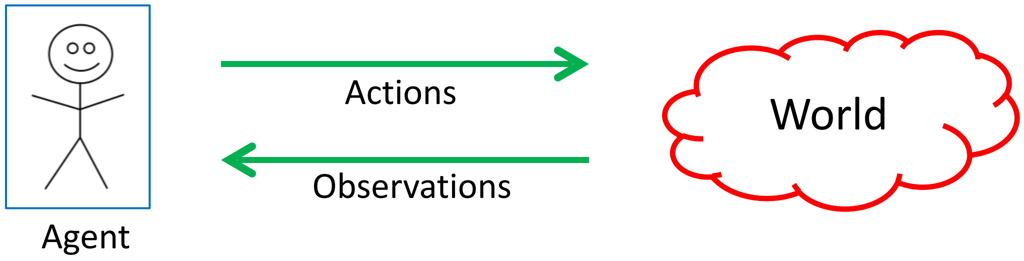
We have an **agent** **interacting** with the **world**



- Agent receives a reward based on state of the world

Back to Our General Model

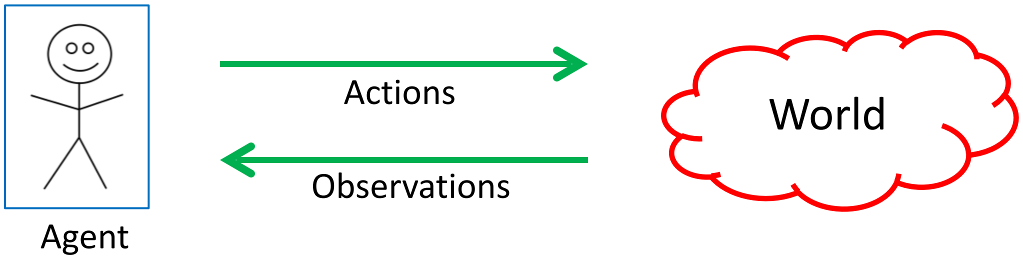
We have an **agent** **interacting** with the **world**



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

Back to Our General Model

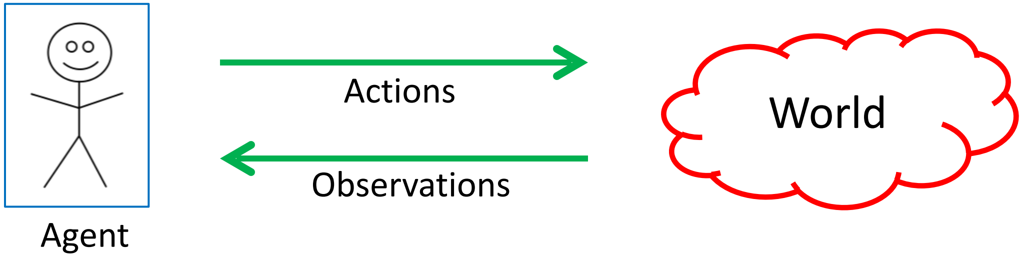
We have an **agent** **interacting** with the **world**



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

Back to Our General Model

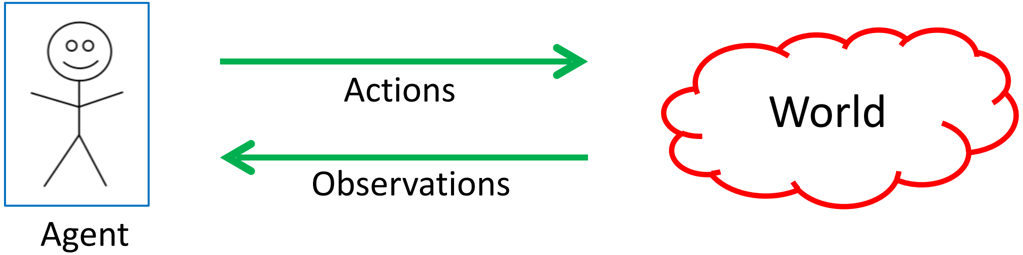
We have an agent interacting with the world



- 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

Examples: Gameplay Agents

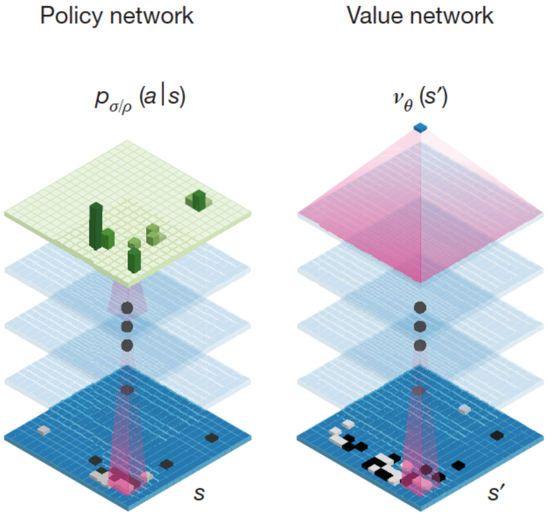
AlphaZero:

Examples: Gameplay Agents

AlphaZero:

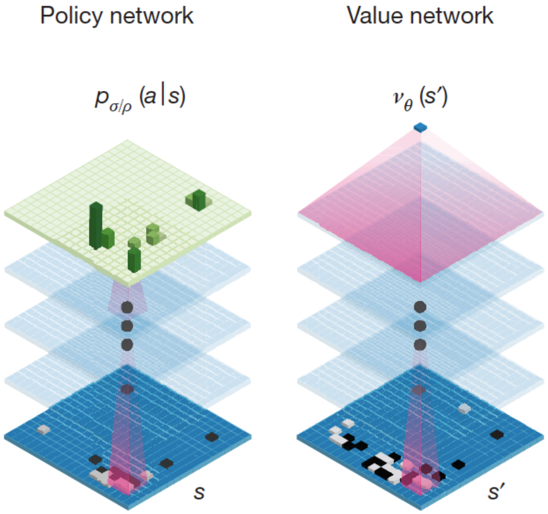
Examples: Gameplay Agents

AlphaZero:



Examples: Gameplay Agents

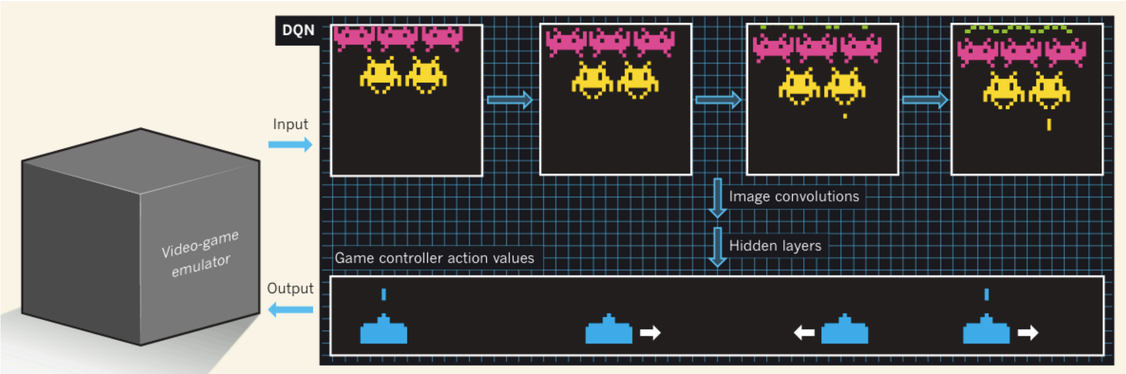
AlphaZero:



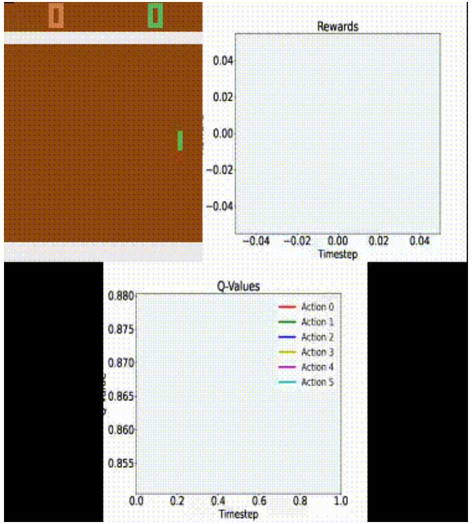
https://deepmind.com/research/alphago/

Examples: Video Game Agents

Pong, Atari



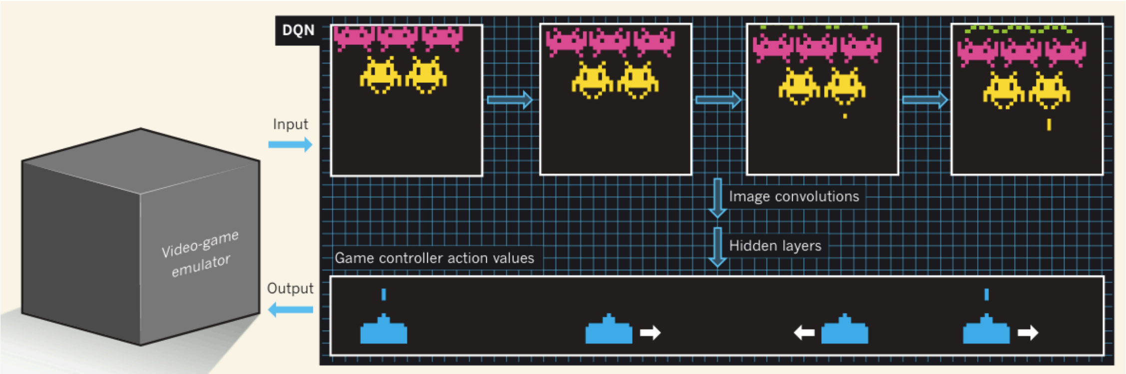
Mnih et al, “Human-level control through deep reinforcement learning”



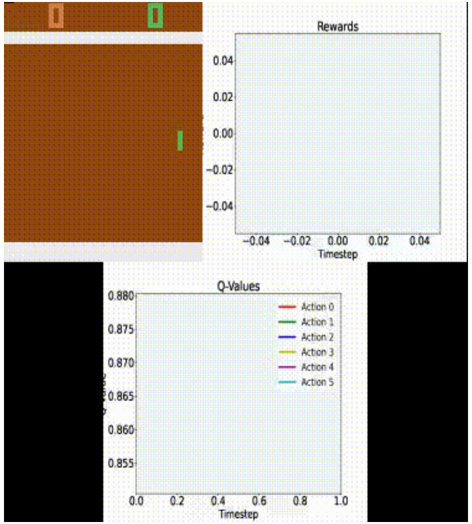
A. Nielsen

Examples: Video Game Agents

Pong, Atari



Mnih et al, "Human-level control through deep reinforcement learning"



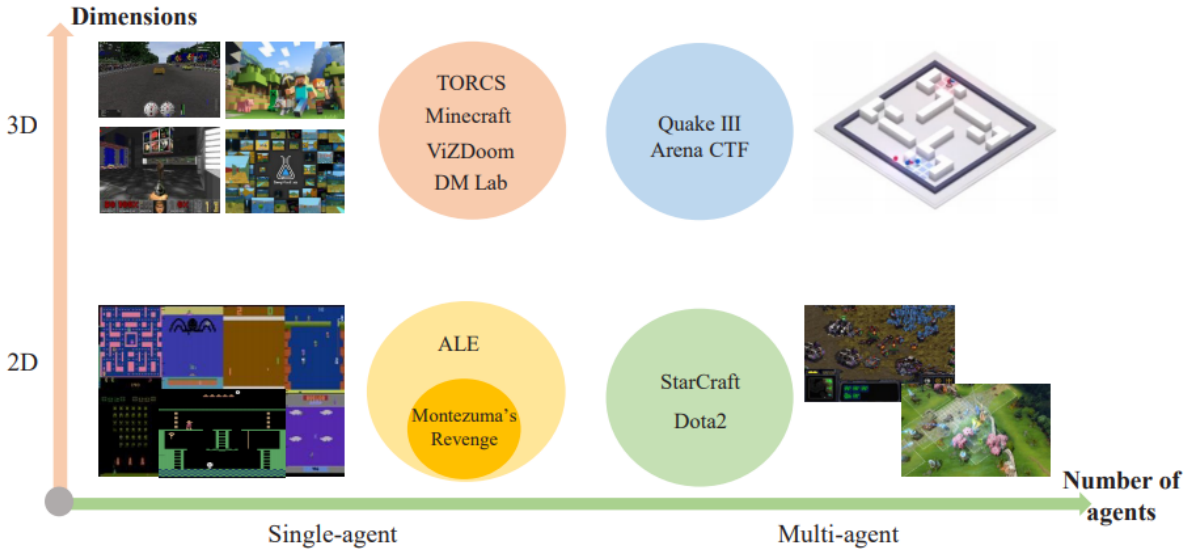
A. Nielsen

Examples: Video Game Agents

Minecraft, Quake, StarCraft, and more!

Examples: Video Game Agents

Minecraft, Quake, StarCraft, and more!



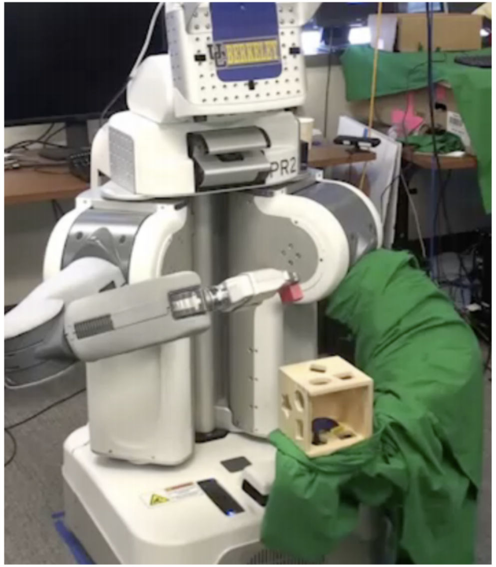
Shao et al, "A Survey of Deep Reinforcement Learning in Video Games"

Examples: Robotics

Training robots to perform tasks (e.g., grasp objects!)

Examples: Robotics

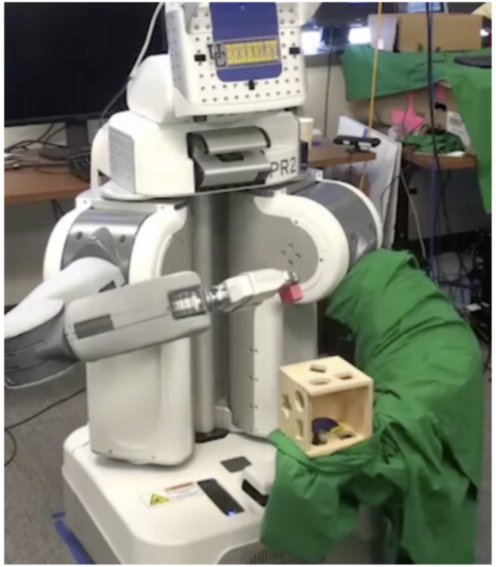
Training robots to perform tasks (e.g., grasp objects!)



Ibarz et al, " How to Train Your Robot with Deep Reinforcement Learning – Lessons We've Learned "

Examples: Robotics

Training robots to perform tasks (e.g., grasp objects!)



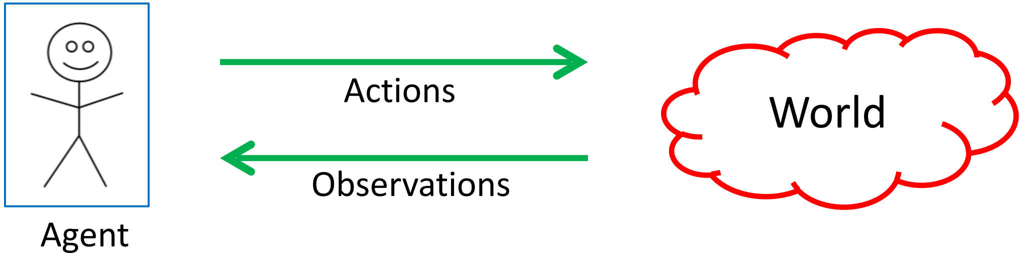
Ibarz et al, " How to Train Your Robot with Deep Reinforcement Learning – Lessons We’ve Learned "

Building The Theoretical Model

Basic setup:

Building The Theoretical Model

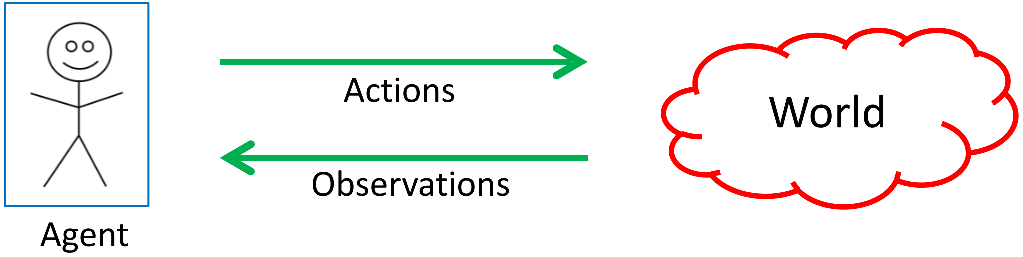
Basic setup:



Building The Theoretical Model

Basic setup:

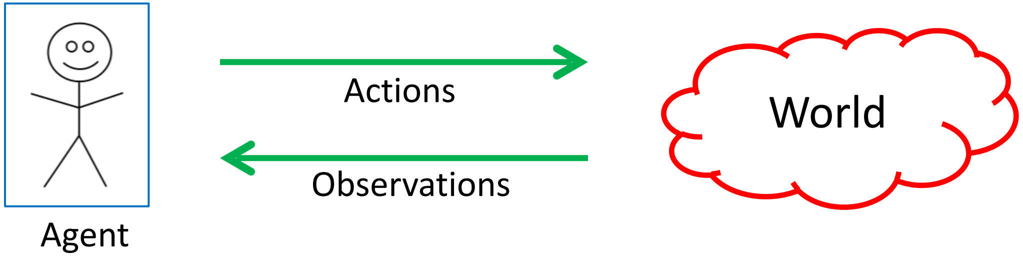
- Set of states, S



Building The Theoretical Model

Basic setup:

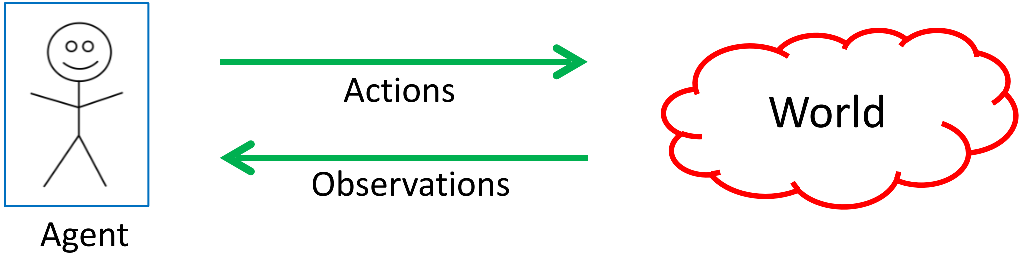
- Set of states, S
- Set of actions A



Building The Theoretical Model

Basic setup:

- Set of states, S
- Set of actions A

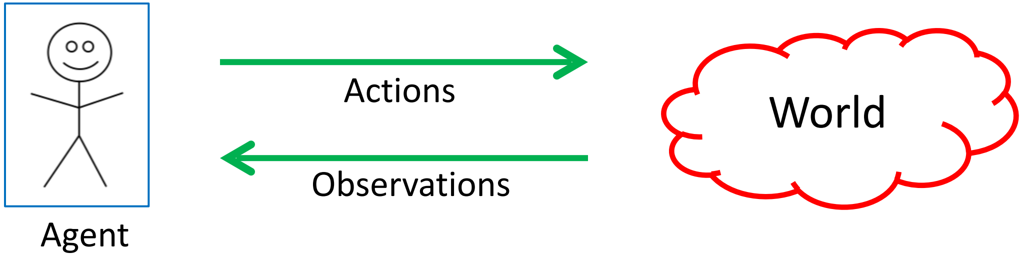


- Information: at time t, observe state $s_t \in S$. Get reward r_t

Building The Theoretical Model

Basic setup:

- Set of states, S
- Set of actions A



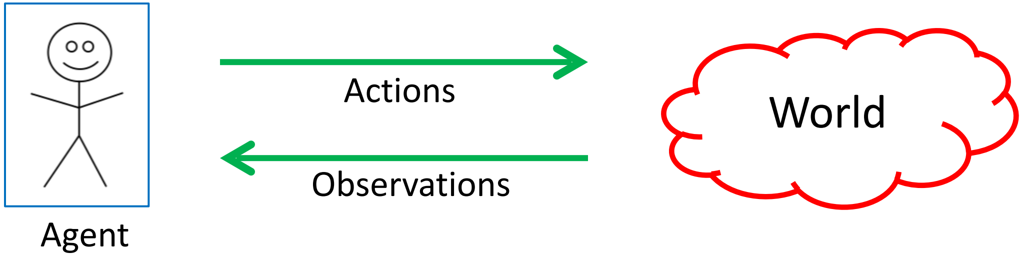
- Information: at time t, observe state $s_t \in$ S. Get reward r_t
- Agent makes choice $a_t \in$ A. State changes to $s_{t+1,}$ continue

Goal: find a map from **states to actions** that maximize rewards.

Building The Theoretical Model

Basic setup:

- Set of states, S
- Set of actions A



- Information: at time *t*, observe state $s_t \in$ S. Get reward r_t
- Agent makes choice $a_t \in$ A. State changes to s_{t+1}, continue

Goal: find a map from **states to actions** that maximize rewards.

A “policy”

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
- **Reward function:** $r(s_t)$

Markov Decision Process (MDP)

The formal mathematical model:

- **State set** S. Initial state $s_{0.}$ **Action set** A
- **Reward function:** $r(s_t)$
- **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
- **Reward function:** $r(s_t)$
- **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 earlier history (previous actions or states)

Markov Decision Process (MDP)

The formal mathematical model:

- **State set** S. Initial state s_0. **Action set** A
- **Reward function:** $r(s_t)$
- **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 earlier history (previous actions or states)
- More generally: $r(s_t, a_t)$, $P(r_t,\ s_{t+1} \mid s_t,\ a_t)$

Markov Decision Process (MDP)

The formal mathematical model:

- **State set** S. Initial state s_0. **Action set** A
- **Reward function:** $r(s_t)$
- **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 earlier history (previous actions or states)
- More generally: $r(s_t, a_t)$, $P(r_t,\ s_{t+1} | s_t,\ a_t)$
- **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
- **Reward function:** $r(s_t)$
- **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 earlier history (previous actions or states)
- More generally: $r(s_t, a_t)$, $P(r_t, s_{t+1} | s_t, a_t)$
- **Policy**: $\pi(s) : S \rightarrow A$ action to take at a particular state

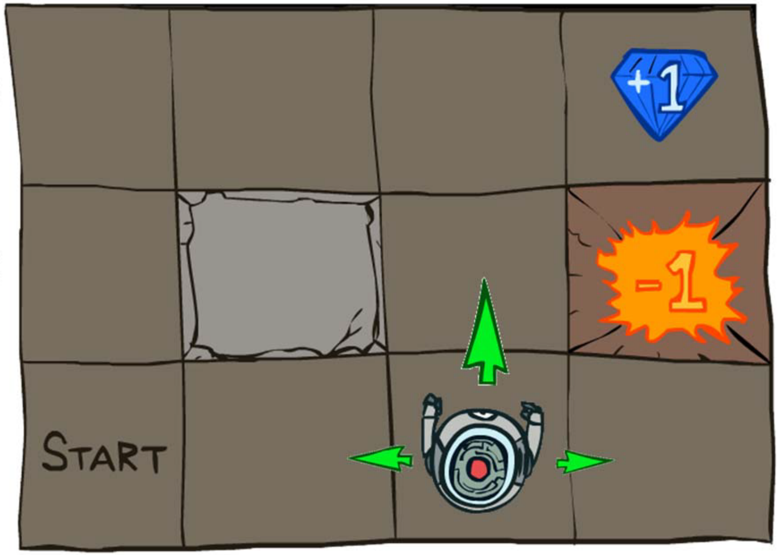
$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$$

Example of MDP: Grid World

Robot on a grid; goal: find the best policy

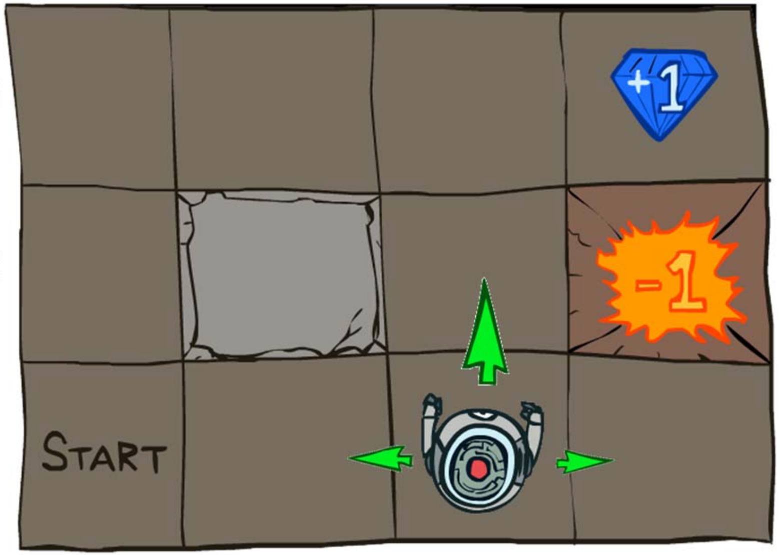
Example of MDP: Grid World

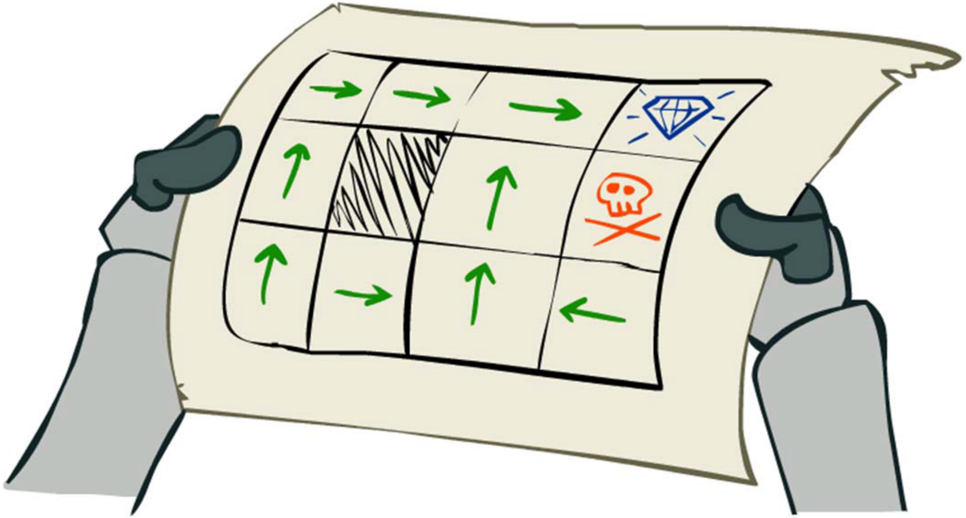
Robot on a grid; goal: find the best policy



Example of MDP: Grid World

Robot on a grid; goal: find the best policy





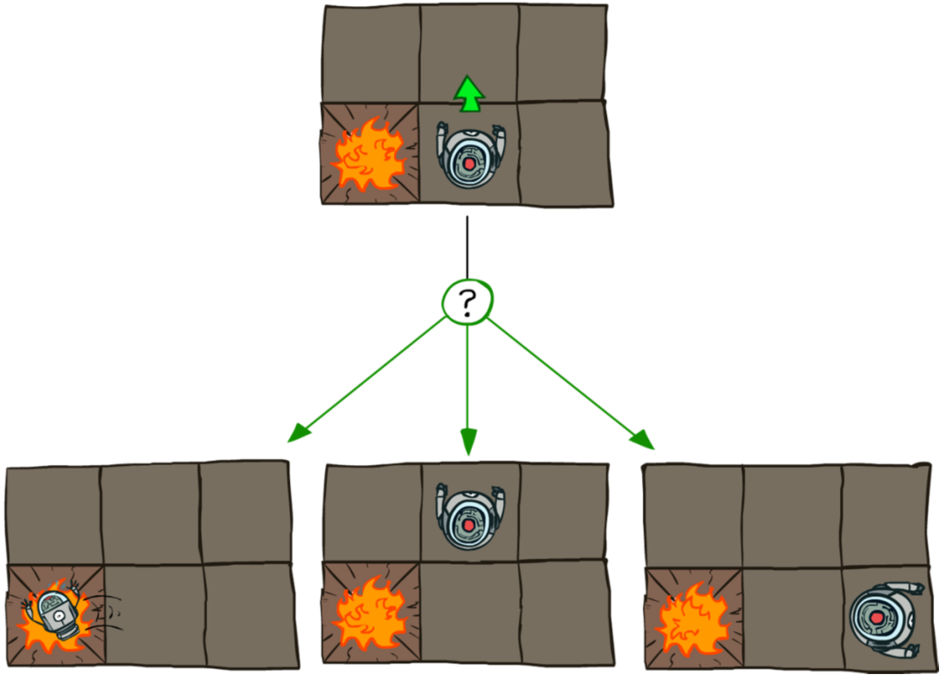
Source: P. Abbeel and D. Klein

Example of MDP: Grid World

Note: (i) Robot is unreliable (ii) Reach target fast

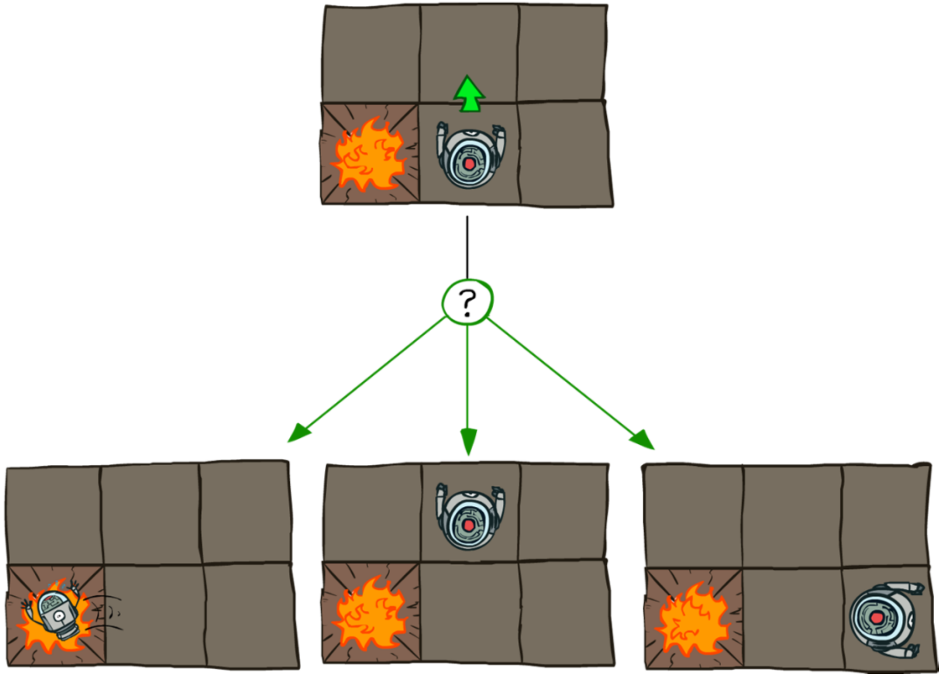
Example of MDP: Grid World

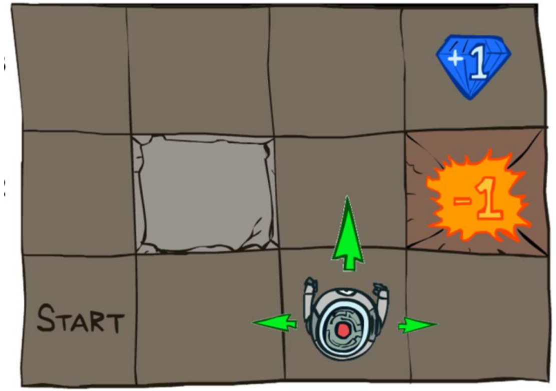
Note: (i) Robot is unreliable (ii) Reach target fast



Example of MDP: Grid World

Note: (i) Robot is unreliable (ii) Reach target fast





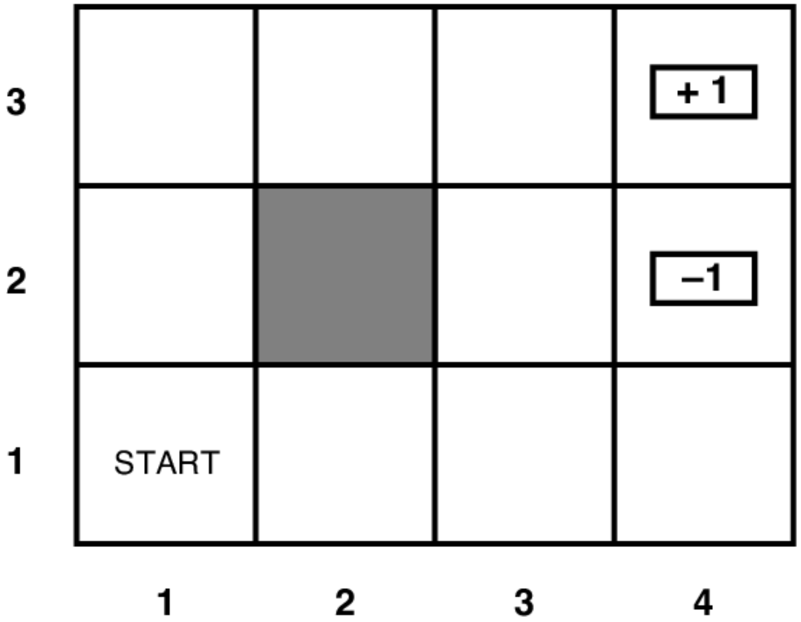
$r(s) = -0.04$ for every non-terminal state

Grid World Abstraction

Note: (i) Robot is unreliable (ii) Reach target fast

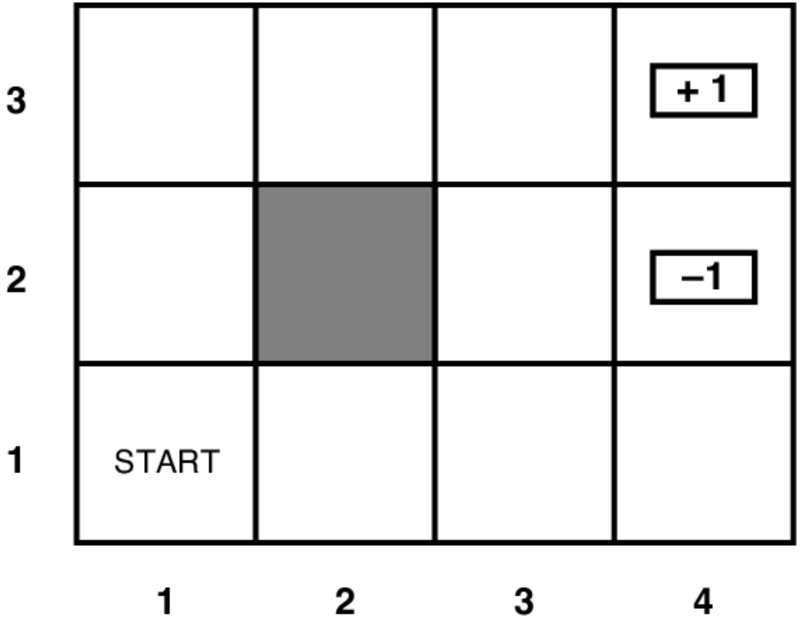
Grid World Abstraction

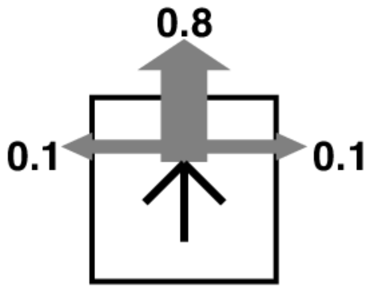
Note: (i) Robot is unreliable (ii) Reach target fast



Grid World Abstraction

Note: (i) Robot is unreliable (ii) Reach target fast

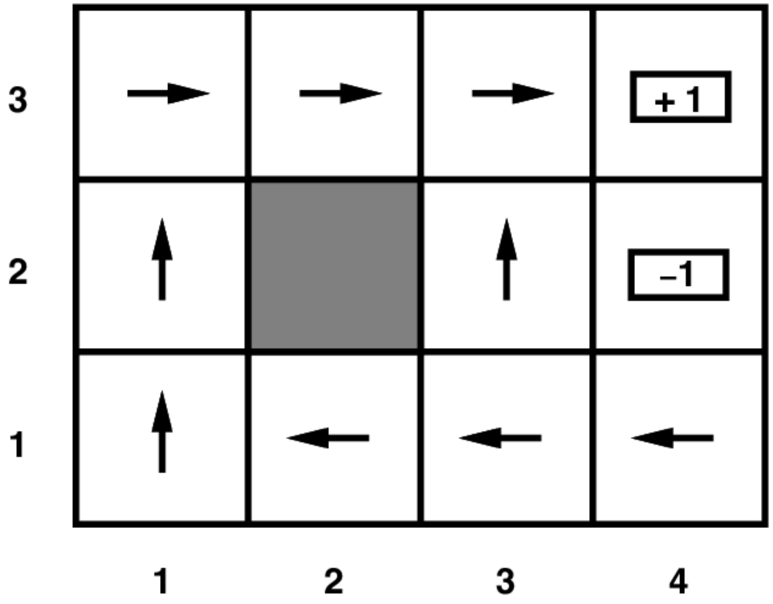


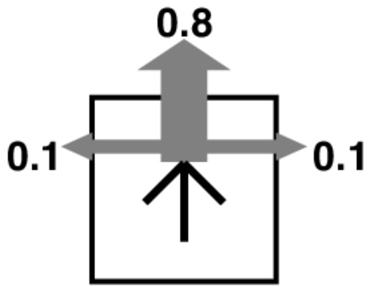


$r(s) = -0.04$ for every non-terminal state

Grid World Optimal Policy

Note: (i) Robot is unreliable (ii) Reach target fast





$r(s) = -0.04$ for every non-terminal state

Back to MDP Setup

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.

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

Back to MDP Setup

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.

How do we find the best policy?

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

Reinforcement Learning Challenges

Reinforcement Learning Challenges

Credit-assignment:

Reinforcement Learning Challenges

Credit-assignment:

- May take many actions before reward is received. Which ones were most important?

Reinforcement Learning Challenges

Credit-assignment:

- May take many actions before reward is received. Which ones were most important?
- Example: you study 15 minutes a day all semester. The morning of the final exam, you eat a bowl of yogurt. You receive an A on the final. Was it the studying or the yogurt that led to the A?

Reinforcement Learning Challenges

Credit-assignment:

- May take many actions before reward is received. Which ones were most important?
- Example: you study 15 minutes a day all semester. The morning of the final exam, you eat a bowl of yogurt. You receive an A on the final. Was it the studying or the yogurt that led to the A?

Exploration vs. Exploitation:

Reinforcement Learning Challenges

Credit-assignment:

- May take many actions before reward is received. Which ones were most important?
- Example: you study 15 minutes a day all semester. The morning of the final exam, you eat a bowl of yogurt. You receive an A on the final. Was it the studying or the yogurt that led to the A?

Exploration vs. Exploitation:

- Transition probabilities and reward may be unknown to the learner.

Reinforcement Learning Challenges

Credit-assignment:

- May take many actions before reward is received. Which ones were most important?
- Example: you study 15 minutes a day all semester. The morning of the final exam, you eat a bowl of yogurt. You receive an A on the final. Was it the studying or the yogurt that led to the A?

Exploration vs. Exploitation:

- Transition probabilities and reward may be unknown to the learner.
- Should you keep trying actions that led to reward in the past or try new actions that might lead to even more reward?

Break & Quiz

Q 1.1 Which of the following statement about MDP is **not** true?

- A. The reward function must output a scalar value
- B. The policy maps states to actions
- C. The probability of next state can depend on current and previous states
- D. The solution of MDP is to find a policy that maximizes the cumulative rewards

Break & Quiz

Q 1.1 Which of the following statement about MDP is **not** true?

- A. The reward function must output a scalar value
- B. The policy maps states to actions
- **C. The probability of next state can depend on current and previous states**
- D. The solution of MDP is to find a policy that maximizes the cumulative rewards

Break & Quiz

Q 1.1 Which of the following statement about MDP is **not** true?

- A. The reward function must output a scalar value (**True: need to be able to compare**)
- B. The policy maps states to actions (**True: a policy tells you what action to take for each state**).
- **C. The probability of next state can depend on current and previous states** (**False: Markov assumption**).
- D. The solution of MDP is to find a policy that maximizes the cumulative rewards (**True: want to maximize rewards overall**).

Defining the Optimal Policy

Defining the Optimal Policy

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

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

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)

Defining the Optimal Policy

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

Utility of sequence

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

Probability of sequence
when following π

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

Defining the Optimal Policy

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

Utility of sequence

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

Probability of sequence when following π

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

Discounting Rewards

One issue: these are possibly infinite series.

Convergence?

Discounting Rewards

One issue: these are possibly infinite series.

Convergence?

- Solution: discount future rewards.

Discounting Rewards

One issue: these are possibly infinite series.

Convergence?

- Solution: discount future rewards.

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

Discounting Rewards

One issue: these are possibly infinite series.

Convergence?

- Solution: discount future rewards.

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

- Discount factor γ between 0 and 1

Discounting Rewards

One issue: these are possibly infinite series.

Convergence?

- Solution: discount future rewards.

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

- Discount factor γ between 0 and 1
  - Set according to how important **present** is VS **future**

Discounting Rewards

One issue: these are possibly infinite series.
Convergence?

- Solution: discount future rewards.

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

- 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

From Value to Policy

From Value to Policy

Now that $V^{\pi}\left(s_0\right)$ is defined, what a should we take?

From Value to Policy

Now that $V^{\pi}\left(s_0\right)$ is defined, what a should we take?

- First, let π^* be the **optimal** policy for $V^{\pi}(s_0)$, and $V^{\star}(s_0)$ its expected utility.

From Value to Policy

Now that $V^{\pi}\left(s_0\right)$ is defined, what a should we take?

- First, let π^* be the **optimal** policy for $V^{\pi}(s_0),$ and $V^{\star}(s_0)$ its expected utility.
- What's the expected utility following an action?

From Value to Policy

Now that $V^{\pi}\left(s_0\right)$ is defined, what a should we take?

- First, let π^* be the **optimal** policy for $V^{\pi}(s_0),$ and $V^{\star}(s_0)$ its expected utility.
- What's the expected utility following an action?
  - Specifically, action a in state s?

From Value to Policy

Now that $V^{\pi}(s_0)$ is defined, what a should we take?

- First, let π^* be the **optimal** policy for $V^{\pi}(s_0)$, and $V^{\star}(s_0)$ its expected utility.
- What's the expected utility following 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

Slight Problem…

Now we can get the optimal policy by doing

Slight Problem...

Now we can get the optimal policy by doing

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

Slight Problem...

Now we can get the optimal policy by doing

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

- So we need to know $V^*(s)$ (and P).

Slight Problem…

Now we can get the optimal policy by doing

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

- So we need to know $V^*(s)$ (and P).
  - But it was defined in terms of the optimal policy!

Slight Problem…

Now we can get the optimal policy by doing

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

- So we need to know $V^*(s)$ (and P).
  - But it was defined in terms of the optimal policy!
  - So we need some other approach to get $V^*(s)$.

Slight Problem...

Now we can get the optimal policy by doing

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

- So we need to know $V^*(s)$ (and P).
  - But it was defined in terms of the optimal policy!
  - So we need some other approach to get $V^*(s)$.
  - Instead, learn about the utility of actions directly.

The Q*(s,a) function

The Q*(s,a) function

- Starting from state s, perform (perhaps suboptimal) action a. THEN follow the optimal policy

The Q*(s,a) function

- Starting from state s, perform (perhaps suboptimal) action a. THEN follow the optimal policy

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

The Q*(s,a) function

- Starting from state s, perform (perhaps suboptimal) action a. THEN follow the optimal policy

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

- Equivalent to

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

Q-Learning

Q-Learning

Q-Learning

- Our first reinforcement learning algorithm.

Q-Learning

- Our first reinforcement learning algorithm.
- Does not require knowing r or P. Learn from data of the form: $\{(s_t, a_t, r_t, s_{t+1})\}$.

Q-Learning

- Our first reinforcement learning algorithm.
- Does not require knowing 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

- Our first reinforcement learning algorithm.
- Does not require knowing 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

- Our first reinforcement learning algorithm.
- Does not require knowing 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:

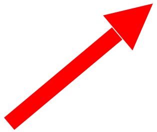
$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_b Q(s_{t+1}, b))$$

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 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.

Exploration Vs. Exploitation

General question!

Exploration Vs. Exploitation

General question!

- **Exploration:** take an action with unknown consequences

Exploration Vs. Exploitation

General question!

- **Exploration:** take an action with unknown consequences
  - **Pros**:
    - Get a more accurate model of the environment
    - Discover higher-reward states than the ones found so far

Exploration Vs. Exploitation

General question!

- **Exploration:** take an action with unknown consequences
  - **Pros**:
    - Get a more accurate model of the environment
    - Discover higher-reward states than the ones found so far
  - **Cons**:
    - When exploring, not maximizing your utility
    - Something bad might happen

Exploration Vs. Exploitation

General question!

- **Exploration:** take an action with unknown consequences
  - **Pros**:
    - Get a more accurate model of the environment
    - Discover higher-reward states than the ones found so far
  - **Cons**:
    - When exploring, not maximizing your utility
    - Something bad might happen
- **Exploitation:** go with the best strategy found so far

Exploration Vs. Exploitation

General question!

- **Exploration:** take an action with unknown consequences
  - Pros:
    - Get a more accurate model of the environment
    - Discover higher-reward states than the ones found so far
  - Cons:
    - When exploring, not maximizing your utility
    - Something bad might happen
- **Exploitation:** go with the best strategy found so far
  - Pros:
    - Maximize reward as reflected in the current utility estimates
    - Avoid bad stuff

Exploration Vs. Exploitation

General question!

- **Exploration:** take an action with unknown consequences
  - Pros:
    - Get a more accurate model of the environment
    - Discover higher-reward states than the ones found so far
  - Cons:
    - When exploring, not maximizing your utility
    - Something bad might happen
- **Exploitation:** go with the best strategy found so far
  - Pros:
    - Maximize reward as reflected in the current utility estimates
    - Avoid bad stuff
  - Cons:
    - Might prevent you from discovering the true optimal strategy

Exploration Vs. Exploitation

General question!

- **Exploration:** take an action with unknown consequences
  - Pros:
    - Get a more accurate model of the environment
    - Discover higher-reward states than the ones found so far
  - Cons:
    - When exploring, not maximizing your utility
    - Something bad might happen
- **Exploitation:** go with the best strategy found so far
  - Pros:
    - Maximize reward as reflected in the current utility estimates
    - Avoid bad stuff
  - Cons:
    - Might prevent you from discovering the true optimal strategy

Exploration Vs. Exploitation

General question!

- **Exploration:** take an action with unknown consequences
  - Pros:
    - Get a more accurate model of the environment
    - Discover higher-reward states than the ones found so far
  - Cons:
    - When exploring, not maximizing your utility
    - Something bad might happen
- **Exploitation:** go with the best strategy found so far
  - Pros:
    - Maximize reward as reflected in the current utility estimates
    - Avoid bad stuff
  - Cons:
    - Might prevent you from discovering the true optimal strategy

Q-Learning: ε-Greedy Behavior Policy

Q-Learning: ε-Greedy Behavior Policy

Getting data with both **exploration and exploitation**

- With probability ε, take a random action; else the action with the highest (current) Q(s,a) value.

Q-Learning: ε-Greedy Behavior Policy

Getting data with both **exploration and exploitation**

- With probability ε, take a random action; else the action with the highest (current) Q(*s*,*a*) value.

$$a = \begin{cases} \text{argmax}_{a \in A} Q(s, a) & \text{uniform}(0,1) > \epsilon \\ \text{random } a \in A & \text{otherwise} \end{cases}$$

Q-Learning Iteration

How do we get $Q(s,a)$?

- Iterative procedure

Idea: combine old value and new estimate of future value.

Note: We are using a policy to take actions; based on the estimated Q!

Q-Learning Iteration

How do we get Q(*s*,*a*)?

- Iterative procedure

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r(s_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Idea: combine old value and new estimate of future value.

Note: We are using a policy to take actions; based on the estimated Q!

Q-Learning Iteration

How do we get Q(*s*,*a*)?

- Iterative procedure

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r(s_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Learning rate

Idea: combine old value and new estimate of future value.

Note: We are using a policy to take actions; based on the estimated Q!

Break & Quiz

Q 2.1 For Q learning to converge to the true Q function, we must

- A. Visit every state and try every action
- B. Perform at least 20,000 iterations.
- C. Re-start with different random initial table values.
- D. Prioritize exploitation over exploration.

Break & Quiz

Q 2.1 For Q learning to converge to the true Q function, we must

- **A. Visit every state and try every action**
- B. Perform at least 20,000 iterations.
- C. Re-start with different random initial table values.
- D. Prioritize exploitation over exploration.

Break & Quiz

Q 2.1 For Q learning to converge to the true Q function, we must

- **A. Visit every state and try every action**
- B. Perform at least 20,000 iterations. (No: this is dependent on the particular problem, not a general constant).
- C. Re-start with different random initial table values. (No: this is not necessary in general).
- D. Prioritize exploitation over exploration. (No: insufficient exploration means potentially unupdated state action pairs).

Summary

- Reinforcement learning setup
- Mathematical formulation: MDP
- The Q-learning Algorithm

Obtaining the Optimal Policy

Obtaining the Optimal Policy

We know the expected utility of an action

- So, to get the optimal policy, compute

Obtaining the Optimal Policy

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

We know the expected utility of an action

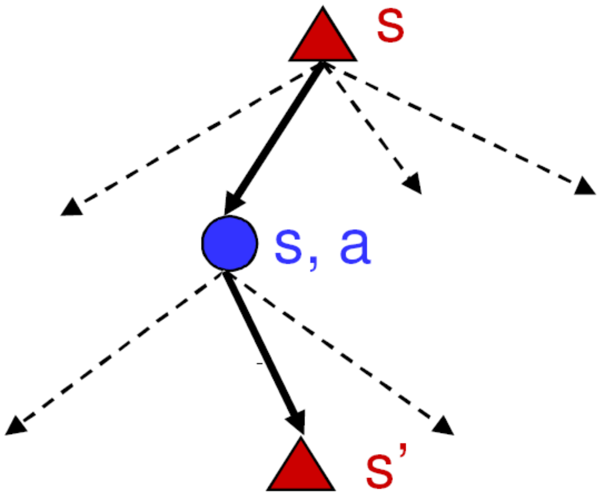
- So, to get the optimal policy, compute

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

All the states we could go to

Transition probability

Expected rewards



Credit L. Lazbenik

Slight Problem…

Now we can get the optimal policy by doing

Slight Problem...

Now we can get the optimal policy by doing

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

Slight Problem…

Now we can get the optimal policy by doing

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

- So we need to know $V^*(s)$.

Slight Problem…

Now we can get the optimal policy by doing

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

- So we need to know $V^*(s)$.
  - But it was defined in terms of the optimal policy!

Slight Problem...

Now we can get the optimal policy by doing

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

- So we need to know $V^*(s)$.
  - But it was defined in terms of the optimal policy!
  - So we need some other approach to get $V^*(s)$.

Slight Problem...

Now we can get the optimal policy by doing

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

- So we need to know *V**(*s*).
  - But it was defined in terms of the optimal policy!
  - So we need some other approach to get *V**(*s*).
  - Need some other **property** of the value function!

Bellman Equation

Bellman Equation

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

Bellman Equation

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

Current state reward

Bellman Equation

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

Current state reward

Discounted expected future **rewards**

Bellman Equation

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

Discounted expected future **rewards**

- Richard Bellman: inventor of dynamic programming

Bellman Equation

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

Discounted expected future **rewards**

- Richard Bellman: inventor of dynamic programming

Bellman Equation

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

Bellman Equation

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 Equation

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 Equation

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

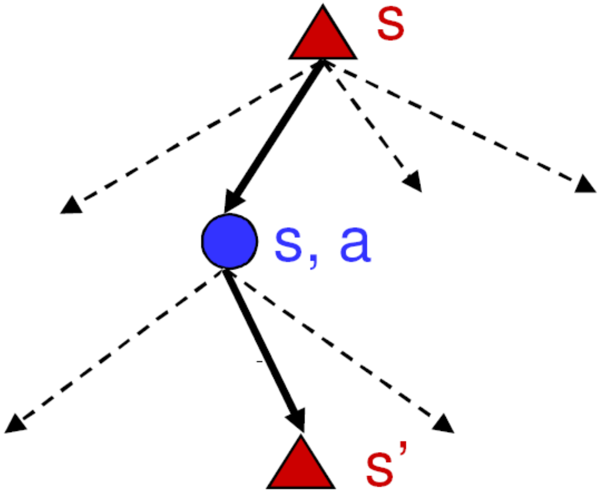
$$V^*(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s,a)V^*(s')$$

Current state reward

Discounted expected future **rewards**

Bellman Equation

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**

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 $P(s'|s,a)$
  - Knowing r and P is the "planning" problem. In reality r and P must be estimated from interactions : "reinforcement learning"

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 $P(s'|s,a)$
  - Knowing r and P is the “planning” problem. In reality r and P must be estimated from interactions : “reinforcement learning”
- Also know $V^*(s)$ satisfies Bellman equation (recursion above)

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 $P(s'|s,a)$
  - Knowing r and P is the “planning” problem. In reality r and P must be estimated from interactions : “reinforcement learning”
- Also know $V^*(s)$ satisfies Bellman equation (recursion above)

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

Value Iteration

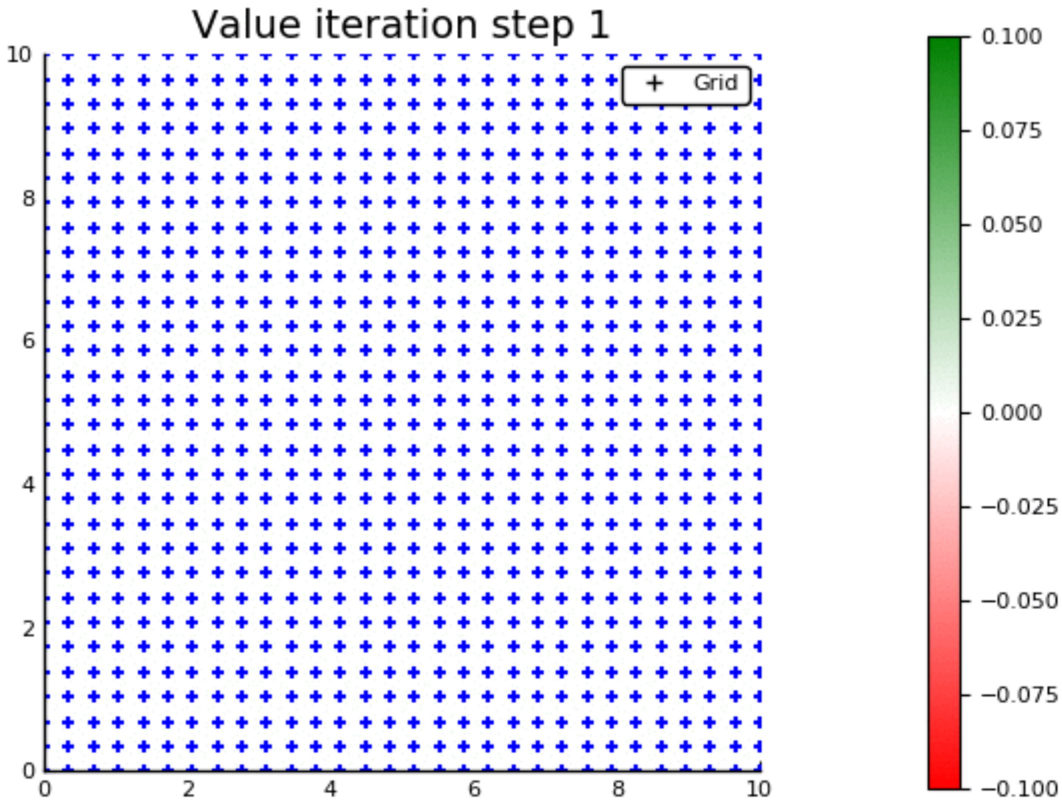
Q: how do do we find $V^*(s)$?

- Why do we want it? Can use it to get the best policy
- Know: reward $r(s)$, transition probability $P(s'|s,a)$
  - Knowing r and P is the "planning" problem. In reality r and P must be estimated from interactions : "reinforcement learning"
- Also know $V^*(s)$ satisfies Bellman equation (recursion above)

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

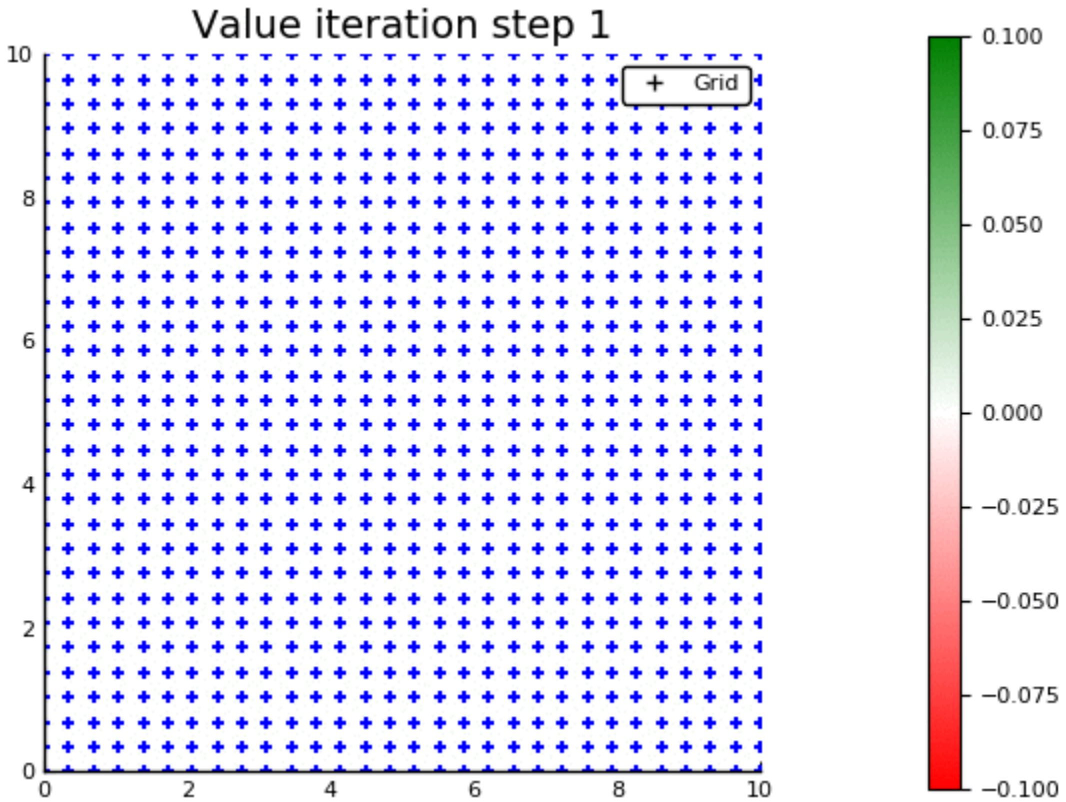
$$V_{i+1}(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s,a) V_i(s')$$

Value Iteration: Demo



Source: POMDPBGallery Julia Package

Value Iteration: Demo



Source: POMDPBGallery Julia Package

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. $1/(1-\gamma^2)$**
- 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

Summary

- Reinforcement learning setup
- Mathematica formulation: MDP
- Value functions & the Bellman equation
- Value iteration