Introduction to Artificial Intelligence

English practicals 9: Decision making

Marika Ivanová

Department of Theoretical Computer Science and Mathematical Logic (KTIML)
Faculty of Mathematics and Physics

April 19th 2022

A small reminder

- $Result(s, a)$: deterministic outcome of taking action a in state s

A small reminder

- *Result*(s, a) : deterministic outcome of taking action a in state s
- Now we assume nondeterministic partially observable environment - we don't know the current state and the outcome of a
- Formal model:
  - *Result*(a) : random variable describing possible outcome state

A small reminder

- $Result(s, a)$: deterministic outcome of taking action a in state s
- Now we assume nondeterministic partially observable environment - we don't know the current state and the outcome of a
- Formal model:
  - $Result(a)$: random variable describing possible outcome state
  - $P(Result(a) = s | a, e)$: probability of outcome s of action a, given evidence observation e

A small reminder

- $Result(s, a)$: deterministic outcome of taking action a in state s
- Now we assume nondeterministic partially observable environment - we don't know the current state and the outcome of a
- Formal model:
  - $Result(a)$: random variable describing possible outcome state
  - $P(Result(a) = s|a, e)$: probability of outcome s of action a, given evidence observation e
- **Utility function** $U(s)$ - number describing desirability of state s

A small reminder

- $Result(s, a)$: deterministic outcome of taking action a in state s
- Now we assume nondeterministic partially observable environment - we don't know the current state and the outcome of a
- Formal model:
    - $Result(a)$: random variable describing possible outcome state
    - $P(Result(a) = s|a, e)$: probability of outcome s of action a, given evidence observation e
- **Utility function** $U(s)$ - number describing desirability of state s
- **Expected utility** of an action a given evidence e: $EU(a|e) = \sum_s P(result(a) = s|a, e)U(s)$

A small reminder

- $Result(s, a)$: deterministic outcome of taking action a in state s
- Now we assume nondeterministic partially observable environment - we don't know the current state and the outcome of a
- Formal model:
  - $Result(a)$: random variable describing possible outcome state
  - $P(Result(a) = s|a, e)$: probability of outcome s of action a, given evidence observation e
- **Utility function** $U(s)$ - number describing desirability of state s
- **Expected utility** of an action a given evidence e:
  $EU(a|e) = \sum_s P(result(a) = s|a, e)U(s)$
- **Maximum expected utility** $action = \arg\max_a EU(a|e)$ - action that should be chosen

A small reminder

- $Result(s, a)$: deterministic outcome of taking action a in state s
- Now we assume nondeterministic partially observable environment - we don't know the current state and the outcome of a
- Formal model:
  - $Result(a)$: random variable describing possible outcome state
  - $P(Result(a) = s|a, e)$: probability of outcome s of action a, given evidence observation e
- **Utility function** $U(s)$ - number describing desirability of state s
- **Expected utility** of an action a given evidence e: $EU(a|e) = \sum_s P(result(a) = s|a, e)U(s)$
- **Maximum expected utility** $action = \arg\max_a EU(a|e)$ - action that should be chosen

We often need to solve a sequential decision problem: make decisions repeatedly (steps of a robot, operations of a space probe)

A small reminder

Markov decision process (MDP)

A small reminder

Markov decision process (MDP)

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

A small reminder

Markov decision process (MDP)

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

  Markov property:

  $P(S_{t+1}|S_0, \ldots, S_t, a_0, \ldots, a_t) = P(S_{t+1}|S_t, a_t)$

A small reminder

Markov decision process (MDP)

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

  Markov property:

  $P(S_{t+1}|S_0, \dots, S_t, a_0, \dots, a_t) = P(S_{t+1}|S_t, a_t)$
- $R(S)$ - real bounded value (”short term” reward, feedback from the environment)

A small reminder

Markov decision process (MDP)

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

  Markov property:

  $P(S_{t+1}|S_0, \ldots, S_t, a_0, \ldots, a_t) = P(S_{t+1}|S_t, a_t)$
- $R(S)$ - real bounded value ("short term" reward, feedback from the environment)
- $U[S_0, S_1, \ldots] = R(S_0) + \gamma R(S_1) + \gamma^2 R(S_2) + \ldots$

A small reminder

Markov decision process (MDP)

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

  Markov property:

  $P(S_{t+1}|S_0, \dots, S_t, a_0, \dots, a_t) = P(S_{t+1}|S_t, a_t)$
- $R(S)$ - real bounded value ("short term" reward, feedback from the environment)
- $U[S_0, S_1, \dots] = R(S_0) + \gamma R(S_1) + \gamma^2 R(S_2) + \dots$
  - $U[S_0, S_1, \dots]$ - utility ("long term" total reward)
  - γ - discount factor (future rewards are less significant), $0 < \gamma \leq 1$

A small reminder

Markov decision process (MDP)

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

  Markov property:

  $P(S_{t+1}|S_0, \ldots, S_t, a_0, \ldots, a_t) = P(S_{t+1}|S_t, a_t)$
- $R(S)$ - real bounded value ("short term" reward, feedback from the environment)
- $U[S_0, S_1, \ldots] = R(S_0) + \gamma R(S_1) + \gamma^2 R(S_2) + \ldots$
  - $U[S_0, S_1, \ldots]$ - utility ("long term" total reward)
  - γ - discount factor (future rewards are less significant), $0 < \gamma \leq 1$

- A solution to a MDP is a policy $\pi : S \mapsto A$
- Policy is stationary *(see later)

A small reminder

Markov decision process (MDP)

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

  Markov property:

  $P(S_{t+1}|S_0, \ldots, S_t, a_0, \ldots, a_t) = P(S_{t+1}|S_t, a_t)$
- $R(S)$ - real bounded value ("short term" reward, feedback from the environment)
- $U[S_0, S_1, \ldots] = R(S_0) + \gamma R(S_1) + \gamma^2 R(S_2) + \ldots$
  - $U[S_0, S_1, \ldots]$ - utility ("long term" total reward)
  - γ - discount factor (future rewards are less significant), $0 < \gamma \leq 1$

- A solution to a MDP is a policy $\pi : S \mapsto A$
- Policy is stationary *(see later)
- MDP is an extension of MC: in addition, MDP has actions and rewards

A small reminder

Markov decision process (MDP)

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

  Markov property:

  $P(S_{t+1}|S_0, \ldots, S_t, a_0, \ldots, a_t) = P(S_{t+1}|S_t, a_t)$
- $R(S)$ - real bounded value ("short term" reward, feedback from the environment)
- $U[S_0, S_1, \ldots] = R(S_0) + \gamma R(S_1) + \gamma^2 R(S_2) + \ldots$
  - $U[S_0, S_1, \ldots]$ - utility ("long term" total reward)
  - γ - discount factor (future rewards are less significant), $0 < \gamma \leq 1$

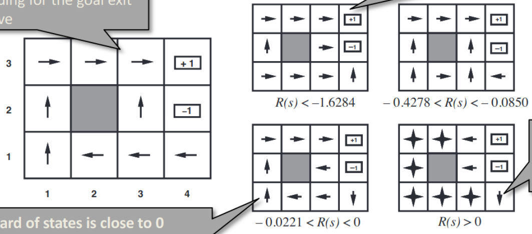
- A solution to a MDP is a policy $\pi : S \mapsto A$
- Policy is stationary *(see later)
- MDP is an extension of MC: in addition, MDP has actions and rewards
- MDP with only action "wait" and all rewards are equal reduces to MC

How to find a policy?

Some examples of optimal policies:

Reward of states is negative
Life is so painful that the agent heads straight for the nearest exist

Reward of states is -0.04
The agent is heading for the goal exit but is conservative

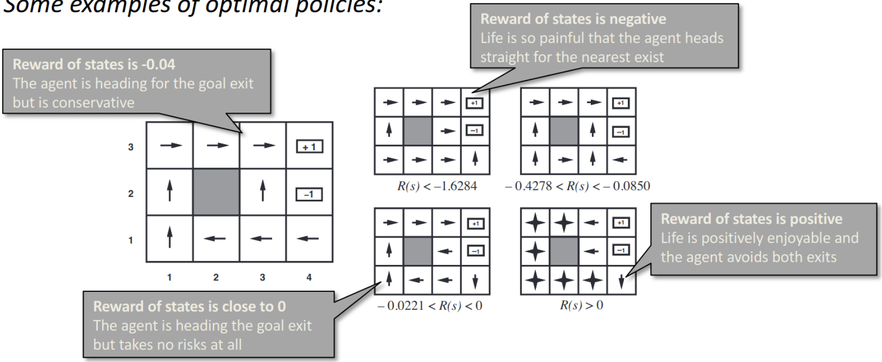


Reward of states is positive
Life is positively enjoyable and the agent avoids both exits

Reward of states is close to 0
The agent is heading the goal exit but takes no risks at all

How to find a policy?

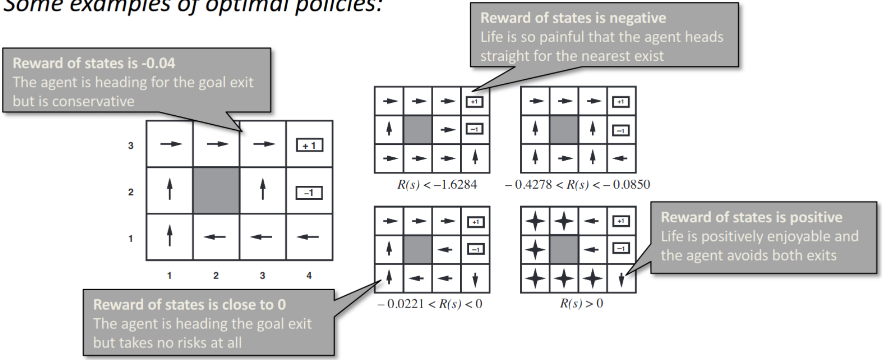
Some examples of optimal policies:



- choose the action maximizing the expected utility of the subsequent state:
  $\pi^*(s) = \arg\max_a \sum_{s'} P(s'|s,a)U(s')$

How to find a policy?

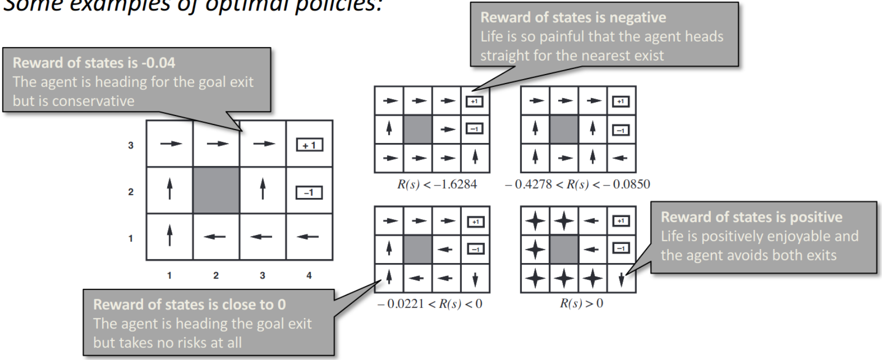
Some examples of optimal policies:



- choose the action maximizing the expected utility of the subsequent state:
  $\pi^*(s) = \arg\max_a \sum_{s'} P(s'|s,a)U(s')$
- utility of a state depends on the utility of its neighbors:
  $U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a)U(s')$ - **Bellman equation**

How to find a policy?

Some examples of optimal policies:



- choose the action maximizing the expected utility of the subsequent state:
  $\pi^*(s) = \arg\max_a \sum_{s'} P(s'|s,a)U(s')$
- utility of a state depends on the utility of its neighbors:
  $U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a)U(s')$ - **Bellman equation**
- two algorithms: **value iteration** and **policy iteration**

Bellman equation

Given a state I am in, assuming I take the best possible action now and at each subsequent step, what long term reward can I expect?

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s, a)U(s')$$

Bellman equation

Given a state I am in, assuming I take the best possible action now and at each subsequent step, what long term reward can I expect?

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s, a) U(s')$$

- What is the VALUE of a state?

Bellman equation

Given a state I am in, assuming I take the best possible action now and at each subsequent step, what long term reward can I expect?

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a)U(s')$$

- What is the VALUE of a state?
- Basic block for solving MDPs, omnipresent in reinforcement learning

Bellman equation

Given a state I am in, assuming I take the best possible action now and at each subsequent step, what long term reward can I expect?

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a)U(s')$$

- What is the VALUE of a state?
- Basic block for solving MDPs, omnipresent in reinforcement learning
- Dynamic programming

Bellman equation

Given a state I am in, assuming I take the best possible action now and at each subsequent step, what long term reward can I expect?

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a)U(s')$$

- What is the VALUE of a state?
- Basic block for solving MDPs, omnipresent in reinforcement learning
- Dynamic programming
- Choice of γ

Bellman equation

Given a state I am in, assuming I take the best possible action now and at each subsequent step, what long term reward can I expect?

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a) U(s')$$

- What is the VALUE of a state?
- Basic block for solving MDPs, omnipresent in reinforcement learning
- Dynamic programming
- Choice of γ
  - Low γ encourages the model to focus on getting reward quickly and ignore long-term reward
  - High γ encourages long-term rewards
  - Typically $0.9 \leq \gamma \leq 0.99$

Value iteration

1. Start with arbitrary initial values for utilities

Value iteration

1. Start with arbitrary initial values for utilities
2. Update the utility $U(s)$ for each state from utilities of neighbors - **Bellman update**:

$$U_{i+1} \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s, a) U_i(s')$$

Value iteration

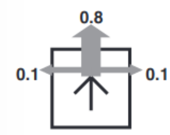
1. Start with arbitrary initial values for utilities
2. Update the utility $U(s)$ for each state from utilities of neighbors - **Bellman update**:

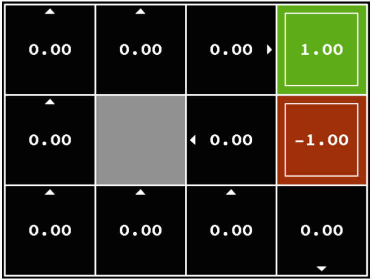
$$U_{i+1} \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) U_i(s')$$

3. Perform action (2) until the utility converges

Example: Value iteration

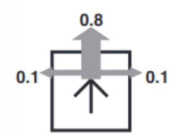
- Reward function:
  $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2 (transition function, as we had on the lecture)

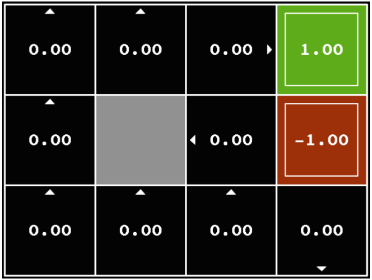




Example: Value iteration

- Reward function:
  $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2 (transition function, as we had on the lecture)



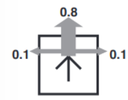


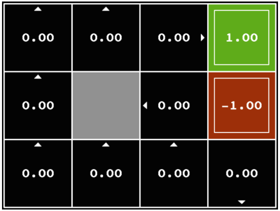
We iteratively update the utility for each state using **Bellman update:**

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) U_i(s')$$

Example: Value iteration

- Reward function:
  $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2

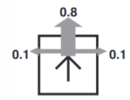


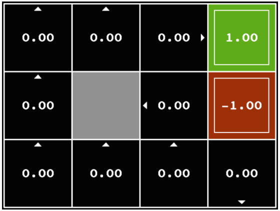


Iteration 1: What are the utilities of states?

Example: Value iteration

- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2



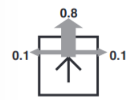


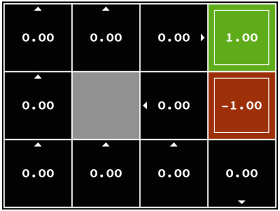
Iteration 1: What are the utilities of states?

State [3,3]:

Example: Value iteration

- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2





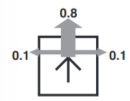
Iteration 1: What are the utilities of states?

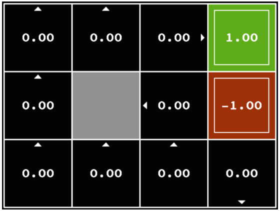
State [3,3]:

$$0 + 0.9 * \max(0.8 * 1 + 0.1 * 0 + 0.1 * 0, 0.8 * 0 + 0.1 * 1 + 0.1 * 0, 0.8 * 0 + 0.1 * 0 + 0.1 * 0) = 0.9 * 0.8 = 0.72$$

Example: Value iteration

- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2





Iteration 1: What are the utilities of states?

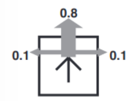
State [3,3]:

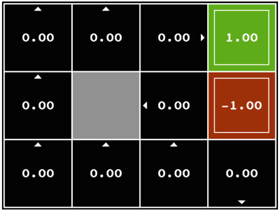
$$0 + 0.9 * \max(0.8 * 1 + 0.1 * 0 + 0.1 * 0, 0.8 * 0 + 0.1 * 1 + 0.1 * 0, 0.8 * 0 + 0.1 * 0 + 0.1 * 0) = 0.9 * 0.8 = 0.72$$

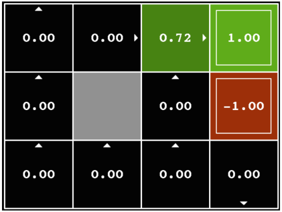
All remaining states have $U(s) = 0$.

Example: Value iteration

- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2







Iteration 1: What are the utilities of states?

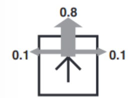
State [3,3]:

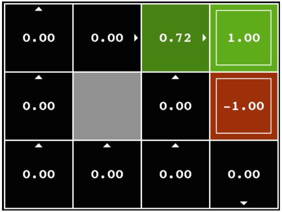
$0 + 0.9 * \max(0.8 * 1 + 0.1 * 0 + 0.1 * 0, 0.8 * 0 + 0.1 * 1 + 0.1 * 0, 0.8 * 0 + 0.1 * 0 + 0.1 * 0) =$
$0.9 * 0.8 = 0.72$

All remaining states have $U(s) = 0$.

Example: Value iteration

- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2

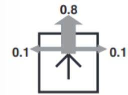


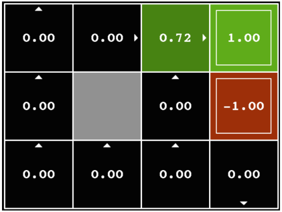


Iteration 2: What are the utilities of states?

Example: Value iteration

- Reward function:
  $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2



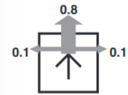


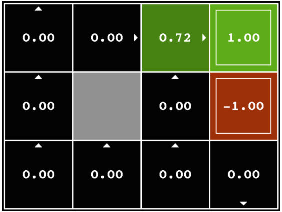
Iteration 2: What are the utilities of states?

State [3,3]:

Example: Value iteration

- Reward function:
  $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2





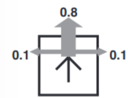
Iteration 2: What are the utilities of states?

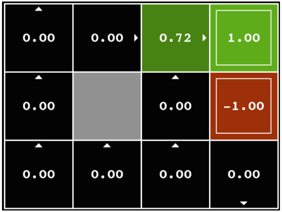
State [3,3]:

$0 + 0.9 *$
$\max(0.8 * 1 + 0.1 * 0.72 + 0.1 * 0, 0.8 * 0 + 0.1 * 1 + 0.1 * 0, 0.8 * 0 + 0.1 * 0.72 + 0.1 * 0) =$
$0.9 * (0.8 + 0.1 * 0.72) = 0.78$

Example: Value iteration

- Reward function:
  $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2





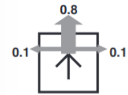
Iteration 2: What are the utilities of states?

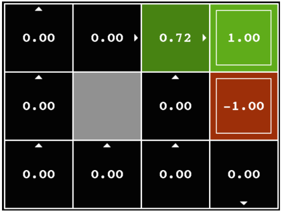
State [3,3]:
$0 + 0.9 *$
$\max(0.8 * 1 + 0.1 * 0.72 + 0.1 * 0, 0.8 * 0 + 0.1 * 1 + 0.1 * 0, 0.8 * 0 + 0.1 * 0.72 + 0.1 * 0) =$
$0.9 * (0.8 + 0.1 * 0.72) = 0.78$

State [3,2]:

Example: Value iteration

- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2





Iteration 2: What are the utilities of states?

State [3,3]:

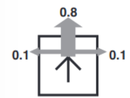
$0 + 0.9 * \max(0.8 * 1 + 0.1 * 0.72 + 0.1 * 0, 0.8 * 0 + 0.1 * 1 + 0.1 * 0, 0.8 * 0 + 0.1 * 0.72 + 0.1 * 0) = 0.9 * (0.8 + 0.1 * 0.72) = 0.78$

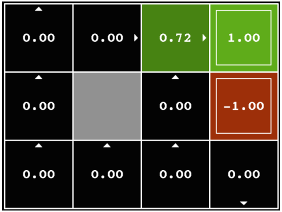
State [3,2]:

$0 + 0.9 * \max(0.8 * 0.72 + 0.1 * (-1) + 0.1 * 0, 0.8 * (-1) + 0.1 * 0.72 + 0.1 * 0, 0.8 * 0 + 0.1 * 0.72 + 0.1 * 0, 0.8 * 0 + 0.1 * (-1) + 0.1 * 0) = 0.9 * (0.8 * 0.72 + 0.1 * (-1)) = 0.43$

Example: Value iteration

- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2





Iteration 2: What are the utilities of states?

State [3,3]:

$0 + 0.9 *$
$\max(0.8 * 1 + 0.1 * 0.72 + 0.1 * 0, 0.8 * 0 + 0.1 * 1 + 0.1 * 0, 0.8 * 0 + 0.1 * 0.72 + 0.1 * 0) =$
$0.9 * (0.8 + 0.1 * 0.72) = 0.78$

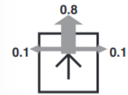
State [3,2]:

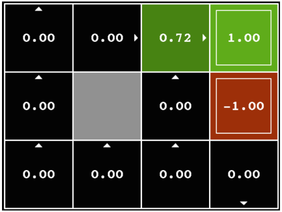
$0 + 0.9 * \max(0.8 * 0.72 + 0.1 * (-1) + 0.1 * 0, 0.8 * (-1) + 0.1 * 0.72 + 0.1 * 0,$
$0.8 * 0 + 0.1 * 0.72 + 0.1 * 0, 0.8 * 0 + 0.1 * (-1) + 0.1 * 0) = 0.9*(0.8*0.72+0.1*(-1)) = 0.43$

State [2,3]:

Example: Value iteration

- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2





Iteration 2: What are the utilities of states?

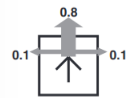
State [3,3]:
$0 + 0.9 *$
$\max(0.8 * 1 + 0.1 * 0.72 + 0.1 * 0, 0.8 * 0 + 0.1 * 1 + 0.1 * 0, 0.8 * 0 + 0.1 * 0.72 + 0.1 * 0) =$
$0.9 * (0.8 + 0.1 * 0.72) = 0.78$

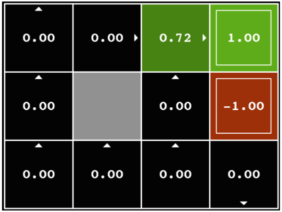
State [3,2]:
$0 + 0.9 * \max(0.8 * 0.72 + 0.1 * (-1) + 0.1 * 0, 0.8 * (-1) + 0.1 * 0.72 + 0.1 * 0,$
$0.8 * 0 + 0.1 * 0.72 + 0.1 * 0, 0.8 * 0 + 0.1 * (-1) + 0.1 * 0) = 0.9*(0.8*0.72+0.1*(-1)) = 0.43$

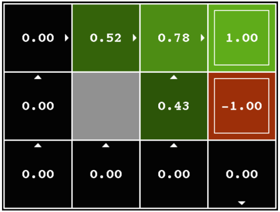
State [2,3]:
$0 + 0.9 *$
$\max(0.8 * 0.72 + 0.1 * 0 + 0.1 * 0, 0.8 * 0 + 0.1 * 0.72 + 0.1 * 0, 0.8 * 0 + 0.1 * 0 + 0.1 * 0) =$
$0.9 * (0.8 * 0.72) = 0.52$

Example: Value iteration

- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2







Iteration 2: What are the utilities of states?

State [3,3]:

$0 + 0.9 * \max(0.8*1+0.1*0.72+0.1*0, 0.8*0+0.1*1+0.1*0, 0.8*0+0.1*0.72+0.1*0) = 0.9*(0.8+0.1*0.72) = 0.78$

State [3,2]:

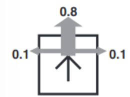
$0 + 0.9 * \max(0.8*0.72+0.1*(-1)+0.1*0, 0.8*(-1)+0.1*0.72+0.1*0, 0.8*0+0.1*0.72+0.1*0, 0.8*0+0.1*(-1)+0.1*0) = 0.9*(0.8*0.72+0.1*(-1)) = 0.43$

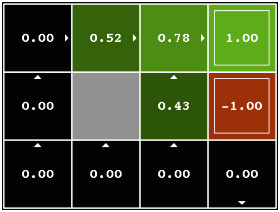
State [2,3]:

$0 + 0.9 * \max(0.8*0.72+0.1*0+0.1*0, 0.8*0+0.1*0.72+0.1*0, 0.8*0+0.1*0+0.1*0) = 0.9*(0.8*0.72) = 0.52$

Example: Value iteration

- Reward function:
  $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2



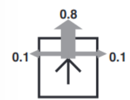


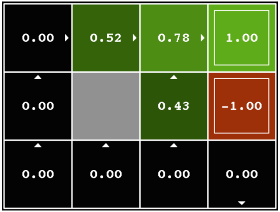
Iteration 3: What are the utilities of states?

..

Example: Value iteration

- Reward function:
  $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2





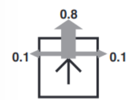
Iteration 3: What are the utilities of states?

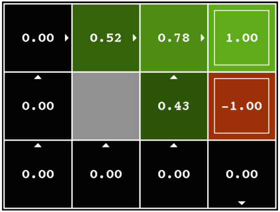
- **State [3,3]:**
  $0.9 * (0.8 * 1 + 0.1 * 0.78 + 0.1 * 0.43) = 0.83$

..

Example: Value iteration

- Reward function:
  $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2





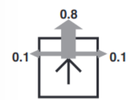
Iteration 3: What are the utilities of states?

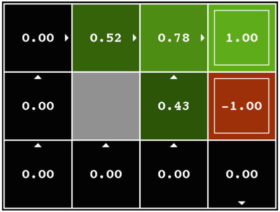
- **State [3,3]:**
  $0.9 * (0.8 * 1 + 0.1 * 0.78 + 0.1 * 0.43) = 0.83$
- **State [3,2]:**
  $0.9 * (0.8 * 0.78) + 0.1 * (-1) + 0.1 * 43) = 0.51$

..

Example: Value iteration

- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2





Iteration 3: What are the utilities of states?

- **State [3,3]:**

  $0.9 * (0.8 * 1 + 0.1 * 0.78 + 0.1 * 0.43) = 0.83$

- **State [3,2]:**

  $0.9 * (0.8 * 0.78) + 0.1 * (-1) + 0.1 * 43) = 0.51$

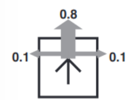
- **State [2,3]:**

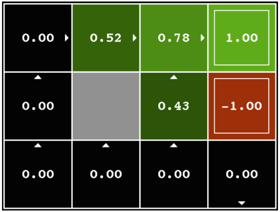
  $0.9 * (0.8 * 0.78 + 0.2 * 0.52) = 0.66$

..

Example: Value iteration

- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2





Iteration 3: What are the utilities of states?

- **State [3,3]:**

  $0.9 * (0.8 * 1 + 0.1 * 0.78 + 0.1 * 0.43) = 0.83$
- **State [3,2]:**

  $0.9 * (0.8 * 0.78) + 0.1 * (-1) + 0.1 * 43) = 0.51$
- **State [2,3]:**

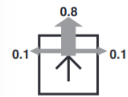
  $0.9 * (0.8 * 0.78 + 0.2 * 0.52) = 0.66$
- **State [3,1]:**

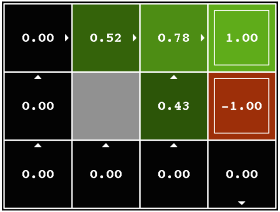
  $0.9 * (0.8 * 0.43) = 0.31$

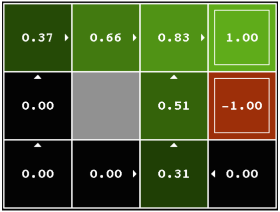
  ..

Example: Value iteration

- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2







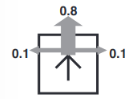
Iteration 3: What are the utilities of states?

- **State [3,3]:**
  $0.9 * (0.8 * 1 + 0.1 * 0.78 + 0.1 * 0.43) = 0.83$
- **State [3,2]:**
  $0.9 * (0.8 * 0.78) + 0.1 * (-1) + 0.1 * 43) = 0.51$
- **State [2,3]:**
  $0.9 * (0.8 * 0.78 + 0.2 * 0.52) = 0.66$
- **State [3,1]:**
  $0.9 * (0.8 * 0.43) = 0.31$

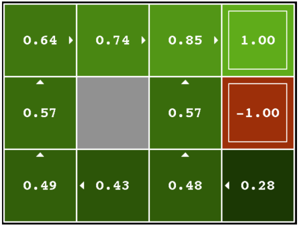
..

Example: Value iteration

- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2 (transition function, as we had on the lecture)

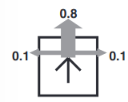


After 100 iterations, we obtain

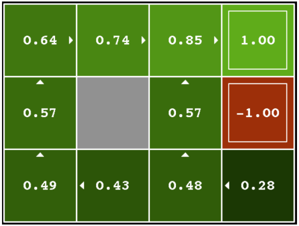


Example: Value iteration

- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2 (transition function, as we had on the lecture)



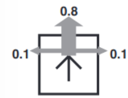
After 100 iterations, we obtain



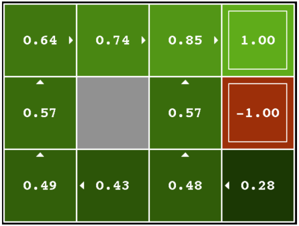
How do we know we converged?

Example: Value iteration

- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2 (transition function, as we had on the lecture)



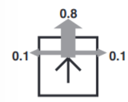
After 100 iterations, we obtain



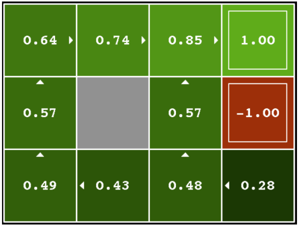
How do we know we converged?
Stopping criteria: $|U_{k+1} - U_k| < \frac{1-\gamma}{\gamma}$

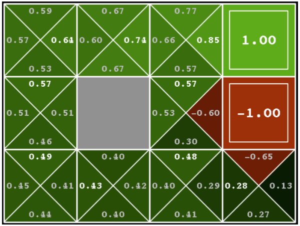
Example: Value iteration

- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2 (transition function, as we had on the lecture)



After 100 iterations, we obtain

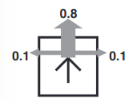


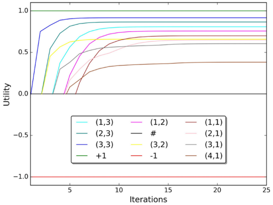


How do we know we converged?
Stopping criteria: $|U_{k+1} - U_k| < \frac{1-\gamma}{\gamma}$

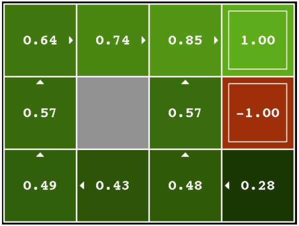
Example: Value iteration

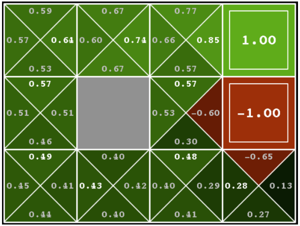
- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2 (transition function, as we had on the lecture)





After 100 iterations, we obtain





How do we know we converged?

Stopping criteria: $|U_{k+1} - U_k| < \frac{1-\gamma}{\gamma}$

Policy iteration

Policy iteration

1. Start with a random policy π

Policy iteration

1. Start with a random policy π
2. Alternate the following two steps

Policy iteration

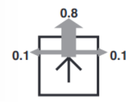
1. Start with a random policy π
2. Alternate the following two steps
    a) Policy evaluation - like the Bellman update, but without "max"

Policy iteration

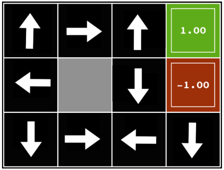
1. Start with a random policy π
2. Alternate the following two steps
   a) Policy evaluation - like the Bellman update, but without "max"
   b) Policy improvement - calculate MEU policy using one step look-ahead

Example: Policy iteration

- Reward function:
  $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2 (transition function, as we had on the lecture)

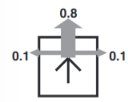


- Initiate all black cells with zeros

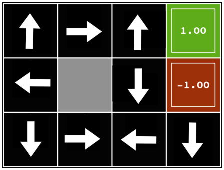


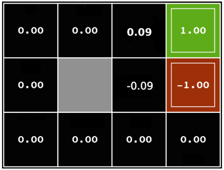
Example: Policy iteration

- Reward function:
  $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2 (transition function, as we had on the lecture)



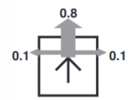
- Initiate all black cells with zeros



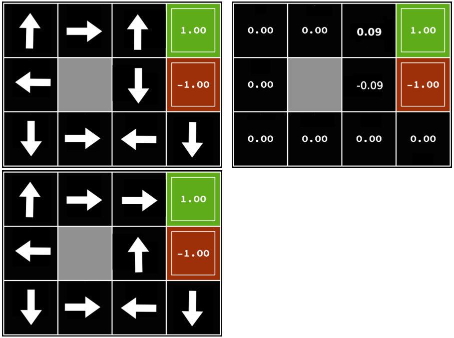


Example: Policy iteration

- Reward function: $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2 (transition function, as we had on the lecture)

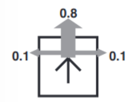


- Initiate all black cells with zeros

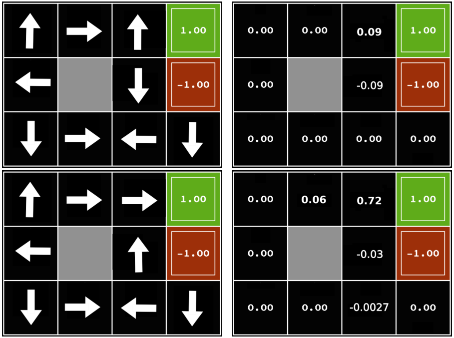


Example: Policy iteration

- Reward function:
  $\forall s \in S : R(s) = 0$
- Discount factor: $\gamma = 0.9$
- Noise: 0.2 (transition function, as we had on the lecture)



- Initiate all black cells with zeros



Value iteration vs Policy iteration

Value iteration

- Start with random utilities
- Iteratively find improved utilities, until reaching optimal value
- Optimal policy can be derived from optimal utilities
- A policy's utility function can be obtained using optimality Bellman operator

Policy iteration

- Evaluate current policy
- Find an improved policy
- Improvement is guaranteed (unless we found the optimum)
- Process based on Bellman operator
- Often converges faster

Selected quiz questions

- Can classical planning be used to solve MDP?

Selected quiz questions

- Can classical planning be used to solve MDP?

  No. Because in stochastic environment, after applying a fixed plan, we may end up in a nonterminal state.

Selected quiz questions

- Can classical planning be used to solve MDP?

  No. Because in stochastic environment, after applying a fixed plan, we may end up in a nonterminal state.

- Prove that utility is finite even for infinite sequences of states, if discount factor is < 1.

Selected quiz questions

- Can classical planning be used to solve MDP?

  No. Because in stochastic environment, after applying a fixed plan, we may end up in a nonterminal state.

- Prove that utility is finite even for infinite sequences of states, if discount factor is < 1.

$$U_h[s_0, s_1, s_2, \dots] = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = \frac{R_{max}}{1-\gamma}$$

Selected quiz questions

- Can classical planning be used to solve MDP?

  No. Because in stochastic environment, after applying a fixed plan, we may end up in a nonterminal state.

- Prove that utility is finite even for infinite sequences of states, if discount factor is < 1.

$$U_h[s_0, s_1, s_2, \ldots] = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = \frac{R_{max}}{1-\gamma}$$

The last equality follows from the sum of an infinite geometric series

Selected quiz questions

- Can classical planning be used to solve MDP?

  No. Because in stochastic environment, after applying a fixed plan, we may end up in a nonterminal state.

- Prove that utility is finite even for infinite sequences of states, if discount factor is < 1.

$$U_h[s_0, s_1, s_2, \ldots] = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = \frac{R_{max}}{1-\gamma}$$

  The last equality follows from the sum of an infinite geometric series

- Does the optimal policy depend on an initial state?

Selected quiz questions

- Can classical planning be used to solve MDP?

  No. Because in stochastic environment, after applying a fixed plan, we may end up in a nonterminal state.

- Prove that utility is finite even for infinite sequences of states, if discount factor is < 1.

$$U_h[s_0, s_1, s_2, \dots] = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = \frac{R_{max}}{1-\gamma}$$

  The last equality follows from the sum of an infinite geometric series

- Does the optimal policy depend on an initial state?

  Given that discounted utilities with infinite horizon are used, the optimal policy is independent of the initial state

- A stationary policy means that the optimal action in a given state cannot change over time.

- A stationary policy means that the optimal action in a given state cannot change over time.

  Is the optimal policy for a finite horizon stationary or nonstationary?

- A stationary policy means that the optimal action in a given state cannot change over time.

  Is the optimal policy for a finite horizon stationary or nonstationary?

  Nonstationary

- A stationary policy means that the optimal action in a given state cannot change over time.

  Is the optimal policy for a finite horizon stationary or nonstationary?

  Nonstationary

- How about for an infinite horizon (not necessarily infinite sequence of states, just no fixed deadline)?

- A stationary policy means that the optimal action in a given state cannot change over time.

  Is the optimal policy for a finite horizon stationary or nonstationary?

  Nonstationary

- How about for an infinite horizon (not necessarily infinite sequence of states, just no fixed deadline)?

  Stationary

- A stationary policy means that the optimal action in a given state cannot change over time.

  Is the optimal policy for a finite horizon stationary or nonstationary?

  Nonstationary
- How about for an infinite horizon (not necessarily infinite sequence of states, just no fixed deadline)?

  Stationary
- What is the difference between reward and utility?

- A stationary policy means that the optimal action in a given state cannot change over time.

  Is the optimal policy for a finite horizon stationary or nonstationary?

  Nonstationary

- How about for an infinite horizon (not necessarily infinite sequence of states, just no fixed deadline)?

  Stationary

- What is the difference between reward and utility?

  Reward: how much we "earn" for visiting a state

- A stationary policy means that the optimal action in a given state cannot change over time.

  Is the optimal policy for a finite horizon stationary or nonstationary?

  Nonstationary

- How about for an infinite horizon (not necessarily infinite sequence of states, just no fixed deadline)?

  Stationary

- What is the difference between reward and utility?

  Reward: how much we "earn" for visiting a state

  Utility: total reward from s onward

- A stationary policy means that the optimal action in a given state cannot change over time.

  Is the optimal policy for a finite horizon stationary or nonstationary?

  Nonstationary
- How about for an infinite horizon (not necessarily infinite sequence of states, just no fixed deadline)?

  Stationary
- What is the difference between reward and utility?

  Reward: how much we "earn" for visiting a state

  Utility: total reward from s onward
- Look at the graph of evolution of utility values. Explain what happened at time around 5.

- A stationary policy means that the optimal action in a given state cannot change over time.

  Is the optimal policy for a finite horizon stationary or nonstationary?

  Nonstationary
- How about for an infinite horizon (not necessarily infinite sequence of states, just no fixed deadline)?

  Stationary
- What is the difference between reward and utility?

  Reward: how much we "earn" for visiting a state

  Utility: total reward from s onward
- Look at the graph of evolution of utility values. Explain what happened at time around 5.

  Calculation of utility values "reached" state [1,1], whose utility started to grow