EE496 : COMPUTATIONAL INTELLIGENCE
RL01: REINFORCEMENT LEARNING : MDP

UGUR HALICI
METU: Department of Electrical and Electronics Engineering (EEE)
METU-Hacettepe U: Neuroscience and Neurotechnology (NSNT)

Reinforcement Learning

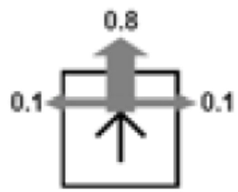
- You can think of supervised learning as theteacher providing answers (the class labels)
- In reinforcement learning, the agent learns based on a punishment/ reward scheme
- Before we can talk about reinforcement learning, we need to introduce Markov Decision Processes

Decision Processes: General Description

- Decide what action to take next given that your action will affect what happens in the future
- Real world examples:
  - Robot path planning
  - Elevator scheduling
  - Travel route planning
  - Aircraft navigation
  - Manufacturing processes
  - Network switching and routing

Sequential Decisions

3				+1
2				-1
1	START			
	1	2	3	4



Assume a fully observable, deterministic environment

- Each grid cell is a state
- The goal state is marked +1
- At each time step, agent must move Up, Right, Down, or Left
- How do you get from start to the goal state?

Sequential Decisions

3				+1
2				-1
1	START			
	1	2	3	4

- Suppose the environment is now stochastic
- With 0.8 probability you go in the direction you intend
- With 0.2 probability you move at right angles to the intended direction (0.1 in either direction – if you hit the wall you stay put)
- What is the optimal solution now?

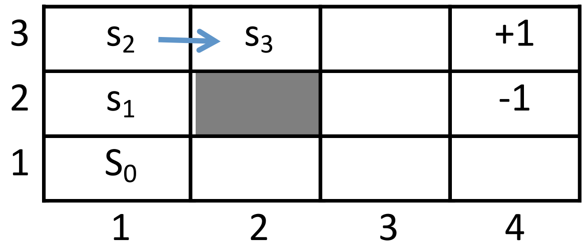
3				+1
2				-1
1	START			
	1	2	3	4

- Up, Up, Right, Right, Right reaches the goal state with probability 0.85=0.32768
- But in this stochastic world, going Up, Up, Right, Right, Right might end up with you actually going Right, Right, Up, Up, Right with probability $(0.1^4)(0.8)=0.00008$
- Even worse, you might end up in the -1 state accidentally

Transition Model

- Transition model: a specification of the outcome probabilities for each action in each possible state
- T(s, a, s’) = probability of going to state s’ if you are in state s and do action a
- The transitions are Markovian ie. the probability of reaching state s’ from s depends only on s and not on the s history of earlier states (aka The Markov Property)
- Mathematically: Suppose you visited the following states in chronological order: s_0, …, s_t

Markov Property Example



Suppose s_0 = (1,1), s_1 = (1,2), s_2 = (1,3)

If I go right from state s_2, the probability of going to s_3 only depends on the fact that I am at state s_2 and not the entire state history $\{s_0, s_1, s_2\}$

The Reward Function

- Depends on the sequence of states (known as the environment history)
- At each state s , the agent receives a reward R(s) which may be positive or negative (but must be bounded)
- For now, we'll define the utility of an environment history as the sum of the rewards receive

Utility Function Example

3				+1
2				-1
1	START			
	1	2	3	4

R(4,3) = +1 (Agent wants to get here)

R(4,2) = -1 (Agent wants to avoid this

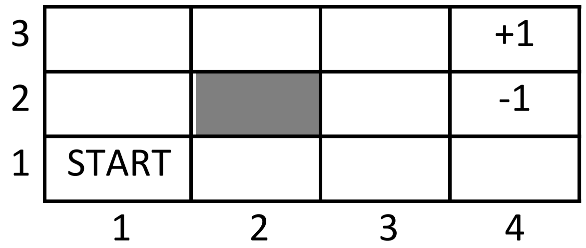
R(s) = -0.04 (for all other states)

$U(s_1, \ldots, s_n) = R(s_1) + \ldots + R(s_n)$

If the states an agent goes through are Up, Up, Right, Right, Right, the utility of this environment history is:

-0.04-0.04-0.04-0.04-0.04+1

Utility Function Example



If there's no uncertainty, then the agent would find the sequence of actions that maximizes the sum of the rewards of the visited states

Markov Decision Process

The specification of a sequential decision problem for a fully observable environment with a Markovian transition model and additive rewards is called a Markov Decision Process (MDP)

A MDP has the following components:

1. A finite set of states S along with an initial state S_0
2. A finite set of actions A
3. Transition Model: T(s, a, s') = P(s' | a, s)
4. Reward Function: R(s)

Solutions to an MDP

3				+1
2				-1
1	START			
	1	2	3	4

Why is the following not a satisfactory solution to the MDP?

[1,1]-Up

[1,2]-Up

[1,3]-Right

[2,3]-Right

[3,3]-Right

A Policy

- Policy: mapping from a state to an action

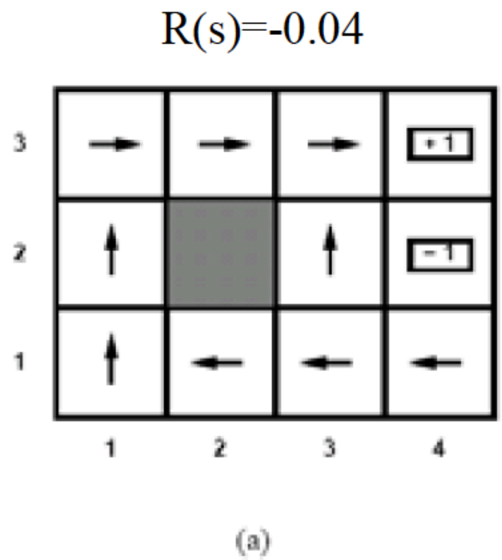
- Need this to be defined for all states so that the agent will always know what to do

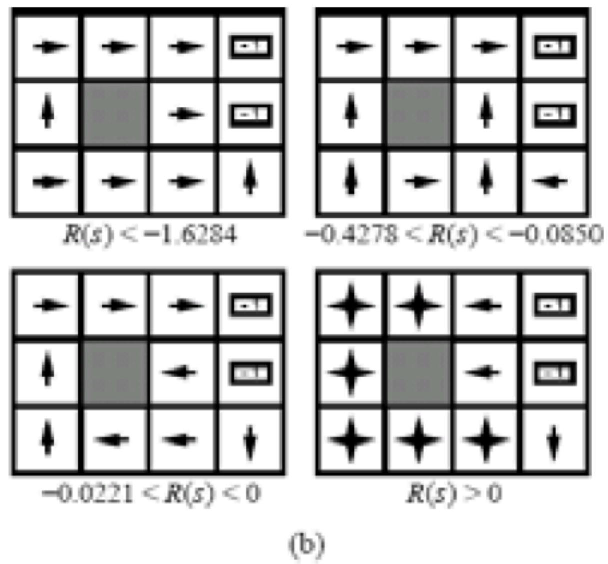
- Notation:
  - π denotes a policy
  - π(s) denotes the action recommended by the policy π for state s

Optimal Policy

- There are obviously many different policies for an MDP
- Some are better than others. The “best” one is called the optimal policy π^* (we will define best more precisely in later slides)
- Note: every time we start at the initial state and execute a policy we get a different environment history (due to the policy, stochastic nature of the environment)
- This means we get a different utility every time we execute a policy
- Need to measure expected utility ie. the average of the utilities of the possible environment histories generated by the policy

Optimal Policy Example





- Notice the tradeoff between risk and reward!

Roadmap for the Next Few Slides

We need to describe how to compute optimal policies

1. Before we can do that, we need to define the utility of a state
2. Before we can do (1), we need to explain stationarity assumption
3. Before we can do (2), we need to explain finite/infinite horizons

Finite/Infinite Horizons

3				+1
2		■		-1
1	START			
	1	2	3	4

- Finite horizon: fixed time N after which nothing matters (think of this as a deadline)
- Suppose our agent starts at (3,1), R(s)=-0.04, and N=3. Then to get to the +1 state, agent must go up.
- If N=100, agent can take the safe route around

Nonstationary Policies

- **Nonstationary policy**: the optimal action in a given state changes over time
- With a finite horizon, the optimal policy is nonstationary
- With an infinite horizon there is no incentive to horizon, behave differently in the same state at different times
- With an infinite horizon, the optimal policy is stationary
- We will assume infinite horizons

Utility of a State Sequence

Under stationarity, there are two ways to assign utilities to sequences:

1. Additive rewards: The utility of a state sequence is:

   $U(s_0, s_1, s_2, \ldots) = R(s_0) + R(s_1) + R(s_2) + \ldots$

2. Discounted rewards: The utility of a state sequence is:

   $U(s0, s1, s2, \ldots) = R(s0) + \gamma R(s1) + \gamma 2R(s2) + \ldots$

   Where $0 \leq \gamma \leq 1$ is the discount factor

The Discount Factor

- Describes preference for current rewards over future rewards
- Compensates for uncertainty in available time (models mortality)
- Eg. Being promised $10000 next year is only worth 90% of being promised $10000 now
- γ near 0 means future rewards don't mean anything
- $\gamma = 1$ makes discounted rewards equivalent to additive rewards

Utilities

- We assume infinite horizons. This means that if the agent doesn't get to a terminal state, then environmental histories are infinite, and utilities with additive rewards are infinite. How do we deal with this?
- Discounted rewards makes utility finite.
- Assuming largest possible reward is Rmax and γ < 1,

$$U(s_0, s_1, s_2, \ldots) = \sum_{t=0}^{\infty} \gamma^t R(s_t)$$

$$\leq \sum_{t=0}^{\infty} \gamma^t R_{\max} = \frac{R_{\max}}{(1-\gamma)}$$

Optimal Policy

- A policy π generates a sequence of states
- But the world is stochastic, so a policy π has a range of possible state sequences, each of which has some probability of occurring
- The value of a policy is the expected sum of discounted rewards obtained
- Given a policy π , we write the expected sum of discounted rewards obtained as:

$$E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi\right]$$

- An optimal policy π * is the policy that maximizes the expected sum above

$$\pi^* = \arg\max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi\right]$$

The Optimal Policy

- For every MDP, there exists an optimal policy
- There is no better option (in terms of expected sum of rewards) than to follow this policy
- How do you calculate this optimal policy? Can't evaluate all policies…too many of them
- First, need to calculate the utility of each state
- Then use the state utilities to select an optimal action in each state

Utilities in the Maze Example

3				+1
2				-1
1	START			
	1	2	3	4

- Start at state (1,1). Let's suppose we choose the action Up.

U(1,1) = R(1,1) + ...

Reward for the current state

Utilities in the Maze Example

3				+1
2				-1
1	START			
	1	2	3	4

- Start at state (1,1). Let's suppose we choose the action Up.

Prob of moving right

$$U(1,1) = R(1,1) + 0.8*U(1,2) + 0.1*U(2,1) + 0.1*U(1,1)$$

Prob of moving up

Prob of moving left (into the wall) and staying same

Utilities in the Maze Example

3				+1
2				-1
1	START			
	1	2	3	4

- Now let's throw in the discounting factor

$$U(1,1) = R(1,1) + \gamma*0.8*U(1,2) + \gamma*0.1*U(2,1) + \gamma*0.1*U(1,1)$$

The Utility of a State

- If we choose action a at state s, expected future rewards (discounted) are:

$$\underbrace{U(s)}_{\text{Expected sum of future discounted rewards starting at state s}} = \underbrace{R(s)}_{\text{Reward at current state s}} + \gamma \sum_{s'} \underbrace{T(s,a,s')}_{\text{Probability of moving from state s to state s' by doing action a}} \underbrace{U(s')}_{\text{Expected sum of future discounted rewards starting at state s'}}$$

The Utility of a State : Bellmann Equation

- In the previous example, we chose the action a then determined the utility of the state.
- The utility is really defined in terms of the optimal action.
- We modify the previous formula slightly by adding a max term over actions

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U(s')$$

- This is the famous **Bellman Equation**: The utility of a state is the immediate reward for that state plus the expected discounted utility of the next state, assuming the agent chooses the optimal action

The Optimal Policy

- Selection of the action π *(s) = a which maximizes the expected utility U(s)
- Intuitively, π * gives us the best action we can take from any state to maximize our future discounted rewards

$$\pi^*(s) = \arg\max_a \sum_{s'} T(s,a,s')U(s')$$