REINFORCEMENT LEARNING

Lara J. Martin (she/they)
TA: Aydin Ayanzadeh (he)

10/24/2023

CMSC 671

By the end of class today, you will be able to:

- Explain why we would use reinforcement learning instead of value/policy iteration
- Describe the relationship between utility and Q-values
- Compare and contrast the effects of exploration and exploitation

Modified from slides by Dr. Cassandra Kent

MDPS IN THE REAL WORLD

- Consider a complex task environment: *Dungeons & Dragons* battle

Screencap from Netflix's Stranger Things via https://www.thetimes.co.uk/article/dungeons-and-dragons-fans-enter-the-land-of-popularity-j0q53lkcw

WHAT DO WE NEED TO DEFINE AN MDP?

Recall that an MDP has 5 components: $(S, A(s), T(s, a, s'), R(s), \gamma)$

- State S:
  - Character stats, health, injuries, inventory, location, skills
  - Monster stats, health, location
  - Not too bad, all of this is finite and known to players
- Actions $A(s)$:
  - Any combination of weapons/spells in inventory used on a certain # of monsters
  - Spells to help team
  - But also…using items
  - Starts to get complicated if you include saying things as the character, but we can ignore this for now

https://www.pngmart.com/files/23/D20-Transparent-PNG.png

WHAT DO WE NEED TO DEFINE AN MDP?

Recall that an MDP has 5 components: $(S, A(s), T(s,a,s'), R(s), \gamma)$

- Transition function $T(s,a,s')$:
  - Some sort of model of the rules of the game
    - E.g., What happens when I cast fireball on this goblin?
- Reward $R(s)$:
  - What do we need to define this for our agent?
  - The experience we get for slaying the monsters?
    - What if we wanted to spare them or they got away?
  - Did we have any other objectives?
    - E.g., Did we get the magical amulet?
  - Note that these rewards don't occur at every step, just at the end!
- Discount factor γ: No problem, just a parameter

https://www.pngmart.com/files/23/D20-Transparent-PNG.png

HOW CAN WE SOLVE THIS TYPE OF MDP?

How can we avoid difficult-to-specify transition functions?

- The world is our transition function!
- We can try an action, and see what happens
- No need to specify everything

How can we avoid difficult-to-specify reward functions?

- Similar idea as above
- Find a way to observe a reward signal
- No need to formulate a function

INTUITION BEHIND RL

Learning by **trial-and-error**

1. Try doing something
2. Leverage the real world or a simulator to observe the transition function and reward function
3. Record the results
4. Repeat, focusing more on what worked in past trials

HIGH-LEVEL RL ALGORITHM

Goal: Compute a policy $\pi(s) \rightarrow a$

1. Start in initial state s_t ($t =$ current time)
2. Observe the reward r
3. Pick an action a_t to execute
4. Executing a_t will cause the agent to go to state s_{t+1}
5. Repeat steps 1-4, until we have a lot of data
6. Compute the best policy $\pi(s)$ we can given what we've observed

We call this a **trial**

MODEL-BASED VS. MODEL-FREE

Model-Based RL: learn $T(s, a, s')$, then compute $\pi(s)$

- Interacting with the environment produces samples of the transition function and the reward function
- Build **explicit** models of T and R
  - $p(s'|s, a) = (\#\ times\ doing\ a\ in\ s\ led\ to\ s')/(\#\ times\ tried\ a\ in\ s)$
  - $R(s) = average\ reward\ for\ s$
- Compute $\pi(s)$ with value/policy iteration

MODEL-BASED VS. MODEL-FREE

Model-Free RL: don't try to learn $T(s, a, s')$, but compute $\pi(s)$ directly

- Interacting with the environment produces samples of the transition function and the reward function
- We **only** care about T and R as a means of computing utility and an optimal policy
- **Directly compute optimal policy** from samples of T and R

MODEL-BASED VS. MODEL-FREE

Model-based RL

- Useful if we need to know the transition function
- See problems such as system identification
- Many model-based approaches – see the first half of Section 21.3 for some examples

Model-Free RL

- More straightforward approach if all you want is a policy
- Approach we will focus on in detail
- Commonly used model-free approach: Q-learning

Q-VALUES AND UTILITY

How Q-learning works:

- Since we're **not** modeling $T(s, a, s')$, move action a into our utility estimate
- We call this a Q-value: Utility of doing action a in state s

$$Q(s, a)$$

- Goal of Q-learning: learn enough Q-values to compute an effective policy

Q-VALUES AND UTILITY

Value Iteration

$$U^{\pi}(s)$$

Long-term expected discounted reward of being in state **s** and following policy π

$$\pi^*(s) = \arg\max_a \sum_{s'} p(s' \mid s, a) U^{\pi^*}(s')$$

Optimal policy in state s

Reinforcement Learning

$$Q^{\pi}(s, a)$$

Long-term expected discounted reward of being in state s, taking action a, *then* following policy π

$$\pi^*(s) = \arg\max_a Q^{\pi^*}(s, a)$$

Optimal policy in state s

Q-VALUES AND UTILITY

$U(s)$ and $\mathrm{Q}(s, a)$ represent the same thing!

- $\mathrm{U}(s)$ is useful for Value and Policy Iteration
- $Q(s, a)$ is easier to work with for model-free reinforcement learning

We can relate the two values directly:

$$U(s) = \max_a Q(s, a)$$

$$Q(s, a) = R(s) + \boxed{\gamma \sum_{s'} \underline{p(s'|s, a)}\, U(s')}$$

Discounted expected utility of doing action a in state s

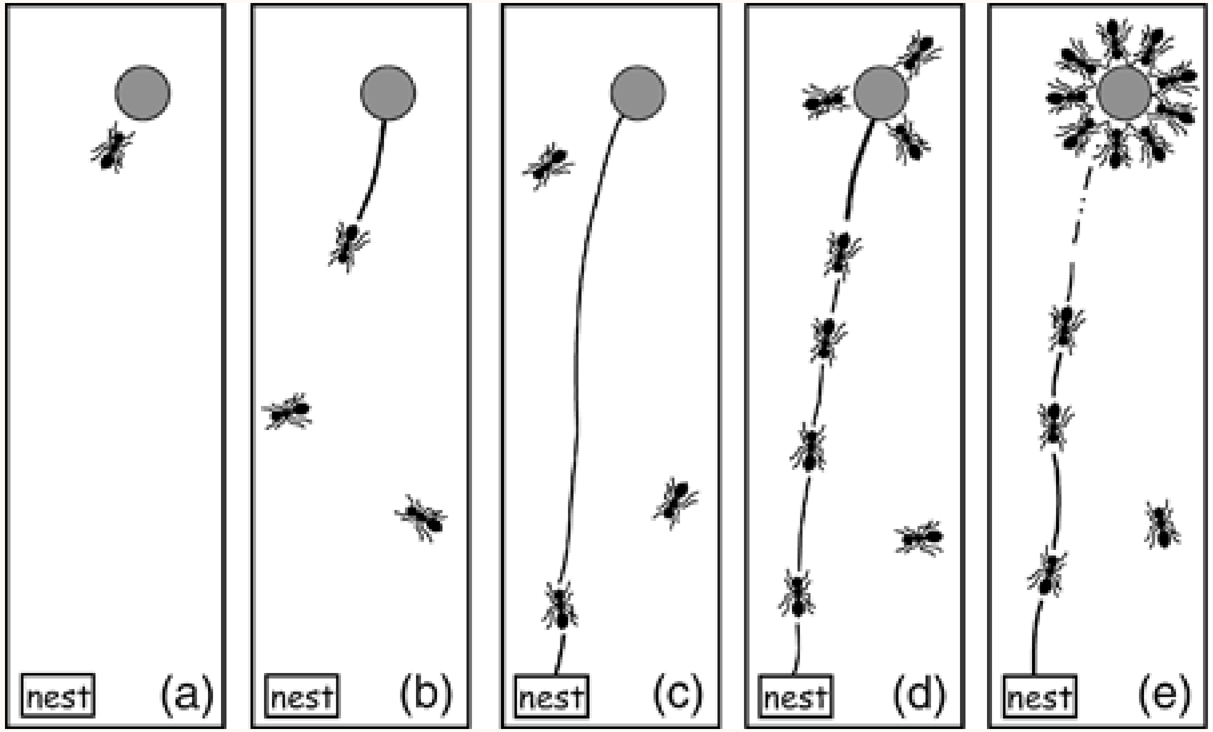
Implicitly encodes transition function

THE Q-LEARNING ALGORITHM

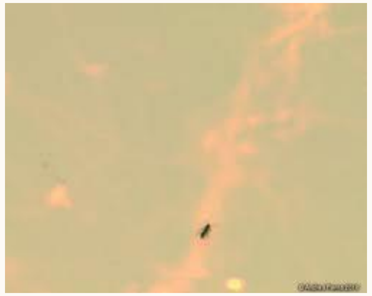
1. Start in initial state s_t (t = current time)
2. Observe the reward r
3. Pick an action a_t to execute
4. Executing a_t will cause the agent to go to state s_{t+1}
5. Update Q-value for $Q(s_t, a_t)$
6. Repeat steps 1-5, until we have a lot of data
7. Compute $\pi(s)$ from $Q(s, a)$

$$\pi^*(s) = \arg\max_a Q^{\pi^*}(s, a)$$

ANALOGY: ANTS



https://resjournals.onlinelibrary.wiley.com/doi/10.1111/j.1365-3032.2008.00658.x



https://i.makeagif.com/media/6-14-2016/3eCU2f.gif

UPDATING Q-VALUES

The Q-value update equation:

Observed reward

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r + \gamma \max_{a'}\left(\overbrace{Q(s_{t+1}, a')}^{\text{The best value of doing } a' \text{ in } s_{t+1}}\right) - Q(s_t, a_t)\right)$$

Update the state we came from

Learning rate $0 < \alpha \leq 1$

All actions available from s_{t+1}

Temporal difference

UPDATING Q-VALUES

The Q-value update equation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\underbrace{\Big(r + \gamma \overbrace{\max_{a'}\big(Q(s_{t+1}, a')\big)}^{U(s_{t+1})}}_{\text{New measurement of } Q(s_t, a_t)} - Q(s_t, a_t)\Big)$$

Recall: $U(s) = \max_a Q(s, a)$

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

We're updating the Q-Value based on its observed change in value

UPDATING Q-VALUES

The Q-value update equation (for terminal states):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r + \cancel{\gamma \max_{a'}\left(Q(s_{t+1}, a')\right)}^{\,0} - Q(s_t, a_t)\right)$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r - Q(s_t, a_t))$$

Why not set this directly to the observed reward?

- We can set $Q(s_t, a_t) = r$ if our rewards are deterministic
- We should use the update equation if our rewards are stochastic!

D&D Q-TABLE EXAMPLE

	Use Sword	Move forward	Move back	Heal
S1 *Have sword* *Monster alive*	5	2	0	0
S2 *No sword* *Monster alive*	0	0	10	11
S3 *Have sword* *Monster dead*	0.001	2	1	5
S4 *No Sword* *Monster dead*	0	1	0	3

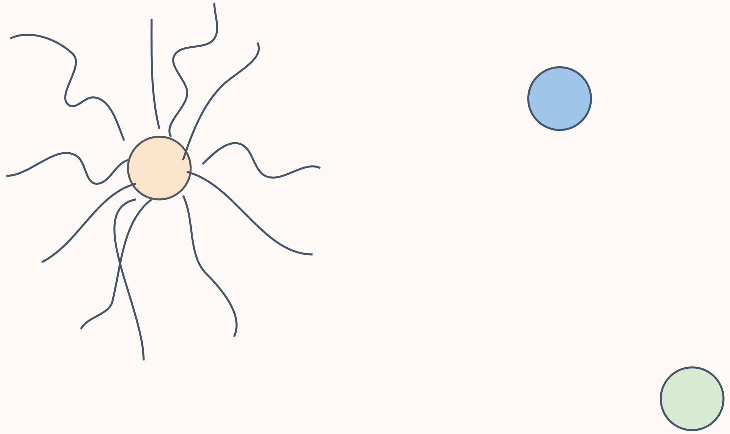
SELECTING ACTIONS DURING TRIALS

Option 1: Random Actions

Always pick an action at random

This provides the extensive **exploration** of the state space

Start
Medium reward
High reward



SELECTING ACTIONS DURING TRIALS - EXAMPLE

Scenario: We just moved to a new city for college. We need to learn a policy for ourselves to get from our apartment to campus.

Option 1: Random Actions

Always pick an action at random

Every day, we'll take a random turn at every intersection.

What's going to happen in this case?