

# 50.007

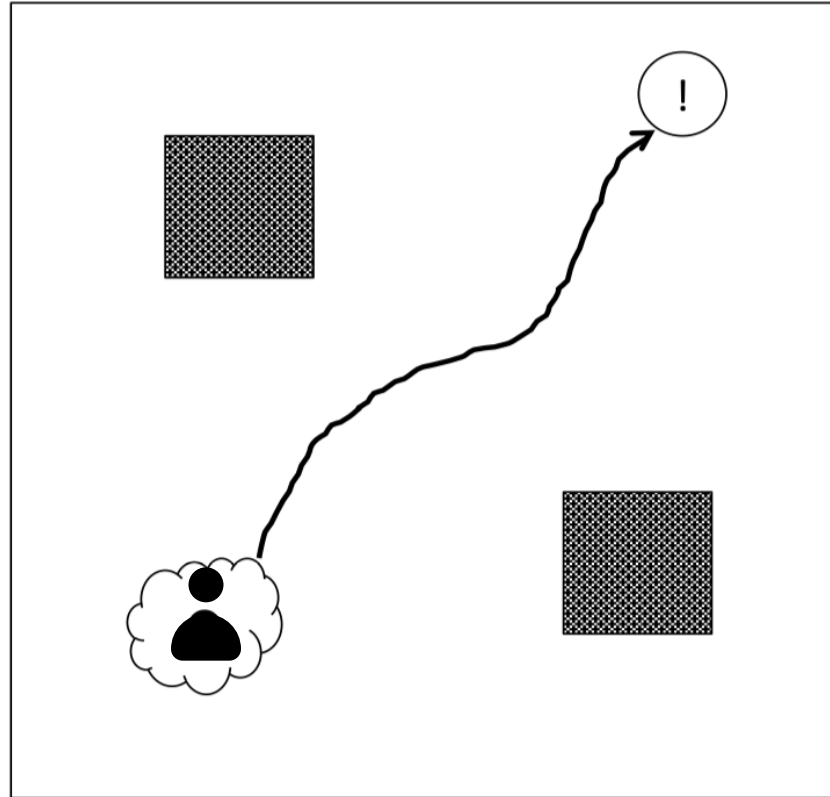
# Machine Learning

Lu, Wei



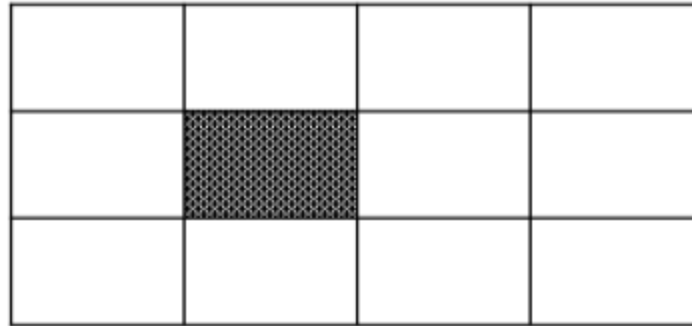
# Reinforcement Learning (II)

# Learn How to Act



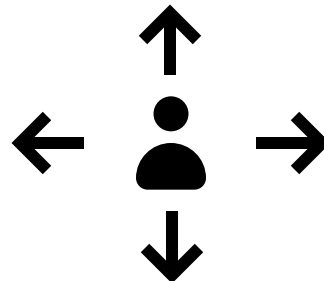
How do we teach a robot how to act optimally in a complex environment?

# Block World Environment



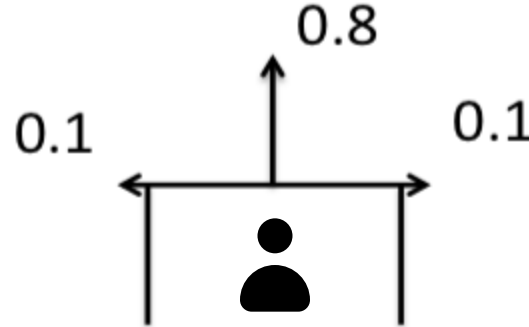
The robot can take an action from a set of predefined possible actions at each state.

There is a set of actions  $A$



# Block World

## Transition Probabilities



If the robot moves towards a particular direction, there is a 0.8 chance that it would reach the block in front, and there is a 0.1 chance to reach a state to its left (right).

A transition probability function

$$T(s, a, s') = p(s'|s, a)$$

# Block World

## Rewards



-0.6	+1.2	+0.1	<b>+1.0</b>
-0.1		-0.1	<b>-1.0</b>
+0.9	-0.7	-2.0	+1.3

Each block is associated with a reward. The two blocks at the upper-right corner are assigned rewards +1 and -1.

The reward is  $R(s)$  for each state.

In general it can be defined as  $R(s, a, s')$

action

old state

new state

# Markov Decision Process



-0.6	+1.2	+0.1	<b>+1.0</b>
-0.1		-0.1	<b>-1.0</b>
+0.9	-0.7	-2.0	+1.3

- a set of states  $S$
- a set of actions  $A$
- a transition probability function  $T(s, a, s') = p(s'|s, a)$
- a reward function  $R(s, a, s')$  (or just  $R(s')$ )

How do we  
learn the policy?

# Block World

## Utility (Long Term Reward)



-0.6	+1.2	+0.1	<b>+1.0</b>
0.1		-0.1	<b>-1.0</b>
+0.9	-0.7	-2.0	+1.3

$$U([s_0, s_1, s_2, \dots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots = \sum_{t=0}^{\infty} \gamma^t R(s_t)$$

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

Lower  
Bound

↓  
Smallest Reward

↙  
Largest Reward

Upper  
Bound



# Policy, Value, Q-Value

$$\pi^*(s)$$

The *optimal policy*  $\pi^*(s)$  specifies the optimal action we should take in state  $s$ .

$$V^*(s)$$

The *value* of state  $s$  under the optimal policy  $\pi^*$

$$Q^*(s, a)$$

The *Q-value* of state  $s$  and action  $a$  under the optimal policy  $\pi^*$

# Policy, Value, Q-Value

$$V^*(s) = \max_a Q^*(s, a) = Q^*(s, \pi^*(s))$$

$$Q^*(s, a) = \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

# Policy, Value, Q-Value

$$V^*(s) = \max_a Q^*(s, a) = Q^*(s, \pi^*(s))$$

$$Q^*(s, a) = \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

$$V^*(s) = \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

# Value Iteration

1. Start with  $V_0^*(s) = 0$ , for all  $s \in S$
2. Given  $V_i^*$ , calculate the values for all states  $s \in S$

$$V_{i+1}^*(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_i^*(s')]$$

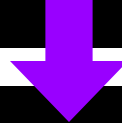
3. Repeat the above until convergence

There is a guarantee that this process will converge

# Learning Optimal Policy

Step 1

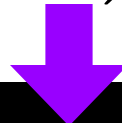
Run Value Iteration Algorithm



Step 2

Calculate the Q values

$$Q^*(s, a) = \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$



Step 3

Find the optimal action for each state

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

# Value Iteration

Step 1

Since the procedure relies on  $Q$ -values,  
is it possible to design an algorithm  
that directly computes these  $Q$ -values?

Step 2

Find the optimal action for each state

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

# Value Iteration

Step 1

Since the procedure relies on  $Q$ -values,  
is it possible to design an algorithm  
that directly computes these  $Q$ -values?

Step 2

Find the optimal action for each state

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

# Policy, Value, Q-Value

$$V^*(s) = \max_a Q^*(s, a) = Q^*(s, \pi^*(s))$$

$$Q^*(s, a) = \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

$$V^*(s) = \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$



# Policy, Value, Q-Value

$$V^*(s) = \max_a Q^*(s, a) = Q^*(s, \pi^*(s))$$

$$Q^*(s, a) = \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$



Can we have an equation for which both sides involve the Q terms only?

# Policy, Value, Q-Value

$$V^*(s) = \max_a Q^*(s, a) = Q^*(s, \pi^*(s))$$

$$Q^*(s, a) = \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$



Can we have an equation for which both sides involve the Q terms only?

# Policy, Value, Q-Value

$$V^*(s) = \max_a Q^*(s, a) = Q^*(s, \pi^*(s))$$

$$Q^*(s, a) = \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

$$V^*(s') = \max_{a'} Q^*(s', a')$$


# Policy, Value, Q-Value

$$Q^*(s, a) \leftarrow \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma \max_{a'} Q^*(s', a')]$$

# Policy, Value, Q-Value

1. Start with  $Q_0^*(s, a) = 0$  for all  $s \in S, a \in A$
2. Given  $Q_i^*(s, a)$ , calculate the Q-values for all states (depth  $i + 1$ ) and for all actions  $a$ :  
$$Q_{i+1}^*(s, a) \leftarrow \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma \max_{a'} Q_i^*(s', a')]$$
3. Repeat the above step until convergence.

# Q-Value Iteration

1. Start with  $Q_0^*(s, a) = 0$  for all  $s \in S, a \in A$
2. Given  $Q_i^*(s, a)$ , calculate the Q-values for all states (depth  $i + 1$ ) and for all actions  $a$ :  
$$Q_{i+1}^*(s, a) \leftarrow \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma \max_{a'} Q_i^*(s', a')]$$
3. Repeat the above step until convergence.

# Q-Value Iteration

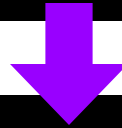
1. Start with  $Q_0^*(s, a) = 0$  for all  $s \in S, a \in A$
2. Given  $Q_i^*(s, a)$ , calculate the Q-values for all states (depth  $i + 1$ ) and for all actions  $a$ :  
$$Q_{i+1}^*(s, a) \leftarrow \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma \max_{a'} Q_i^*(s', a')]$$
3. Repeat the above step until convergence.

Again, there is a guarantee it will converge.

# Learning Optimal Policy

Step 1

Run Q-Value Iteration Algorithm



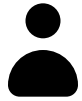
Step 2

Find the optimal action for each state

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



# Markov Decision Process



-0.6	+1.2	+0.1	<b>+1.0</b>
-0.1		-0.1	<b>-1.0</b>
+0.9	-0.7	-2.0	+1.3

- a set of states  $\mathcal{S}$
- a set of actions  $\mathcal{A}$
- a transition probability function  $T(s, a, s') = p(s'|s, a)$
- a reward function  $R(s, a, s')$  (or just  $R(s')$ )

How do we  
learn the policy?

# Reinforcement Learning



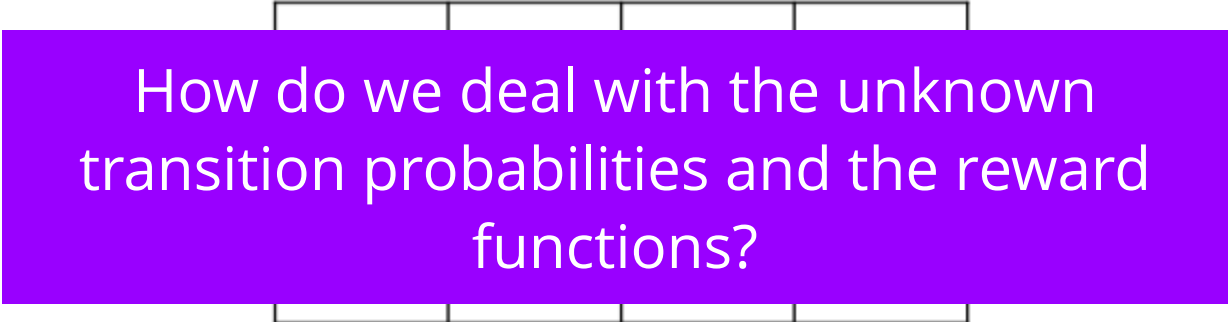
-0.6	+1.2	+0.1	<b>+1.0</b>
-0.1		-0.1	<b>-1.0</b>
+0.9	-0.7	-2.0	+1.3

- a set of states  $\mathcal{S}$
- a set of actions  $\mathcal{A}$

How do we  
learn the policy?

- a transition probability function  $T(s, a, s') = p(s'|s, a)$
- a reward function  $R(s, a, s')$  (or just  $R(s')$ )

# Reinforcement Learning

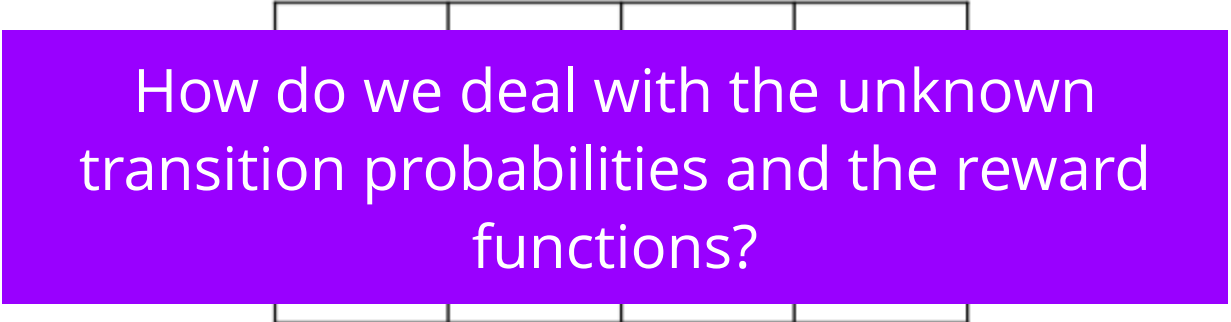


How do we deal with the unknown transition probabilities and the reward functions?

- a set of states  $\mathcal{S}$
- a set of actions  $\mathcal{A}$
- a transition probability function  $T(s, a, s') = p(s'|s, a)$
- a reward function  $R(s, a, s')$  (or just  $R(s')$ )

How do we learn the policy?

# Reinforcement Learning



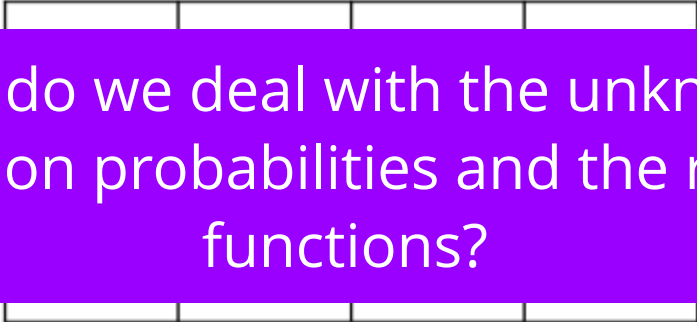
How do we deal with the unknown transition probabilities and the reward functions?

Imagine the robot is moving around...

$$T(s'|s, a) = \frac{\text{Count}(s, a, s')}{\text{Count}(s, a)} \quad R(s, a, s') = \frac{\sum_t R_t(s, a, s')}{\text{Count}(s, a, s')}$$

Model-based Approach  
Estimating the model parameters first

# Reinforcement Learning



How do we deal with the unknown transition probabilities and the reward functions?

Imagine the robot is moving around...

$$T(s'|s, a) = \frac{\text{Count}(s, a, s')}{\text{Count}(s, a)} \quad R(s, a, s') = \frac{\sum_t R_t(s, a, s')}{\text{Count}(s, a, s')}$$

Model-free Approach

Does not explicitly estimate parameters

# Model-Free Approach

## A Simple Game

Flipping a coin, if it is head (H), you receive \$200; otherwise you pay \$100. Would you like to play the game?

# Model-Free Approach

## A Simple Game

Flipping a coin, if it is head (H), you receive \$200, otherwise you pay \$100. Would you like to play the game?

$$p(\text{T}) \times (-100) + p(\text{H}) \times (+200)$$



Model  
Parameters

# Model-Free Approach

H

H

T

T

H

T

T

T

T

T

T

$$p(H) = 0.3$$

$$p(T) = 0.7$$

$$\begin{aligned} & p(T) \times (-100) + p(H) \times (+200) \\ &= -70 + 60 = -10 \end{aligned}$$



# Model-Free Approach

	Outcome	Gain/Loss	Average Return
1	H	+200	+200

# Model-Free Approach

	Outcome	Gain/Loss	Average Return
1	H	+200	+200
2	H	+200	+200

# Model-Free Approach

	Outcome	Gain/Loss	Average Return
1	H	+200	+200
2	H	+200	+200
3	T	−100	+100

# Model-Free Approach

	Outcome	Gain/Loss	Average Return
1	H	+200	+200
2	H	+200	+200
3	T	−100	+100
	T	−100	
	H	+200	
	T	−100	
	T	−100	⋮
	T	−100	⋮
	T	−100	
9	T	−100	
10	T	−100	−10

# Model-Free Approach

	Outcome	Gain/Loss	Average Return
--	---------	-----------	----------------

1	H	+200	+200
---	---	------	------

2	H	+200	+200
---	---	------	------

3	T	−100	+100
---	---	------	------

⋮

⋮

⋮

$k - 1$

T

−100

$v$

# Model-Free Approach

	Outcome	Gain/Loss	Average Return
--	---------	-----------	----------------

1	H	+200	+200
---	---	------	------

2	H	+200	+200
---	---	------	------

3	T	−100	+100
---	---	------	------

⋮

⋮

⋮

$k - 1$	T	−100	$v$
---------	---	------	-----

$k$	H	+200	??
-----	---	------	----

# Model-Free Approach

	Outcome	Gain/Loss	Average Return
--	---------	-----------	----------------

1	H	+200	+200
---	---	------	------

2	H	+200	+200
---	---	------	------

3	T	−100	+100
---	---	------	------

⋮

⋮

⋮

$k - 1$

T

−100

$v$

$k$

H

+200

$$\frac{v \times (k-1) + (+200)}{k}$$

# Model-Free Approach

$$v \leftarrow \frac{v \times (k-1) + (+200)}{k}$$



# Model-Free Approach

$$v \leftarrow v + \frac{1}{k} ((+200) - v)$$

# Model-Free Approach

$$v \leftarrow v + \frac{1}{k} ((+200) - v)$$



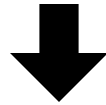
Current estimate  
of Q-Value



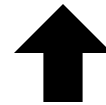
From a newly  
collected sample

# Model-Free Approach

"Learning Rate"



$$v \leftarrow v + \frac{1}{k} ((+200) - v)$$



"Gradient"

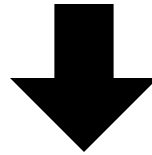
# Updating Q-Values

We have arrived at the following model-free based estimation of the Q-values  
(after taking action  $a$  from state  $s$ )

$$Q(s, a) \leftarrow Q(s, a) + \frac{1}{k} [R(s, a, s') + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

# Q Learning

Collect a sample:  $s, a, s'$  and  $R(s, a, s')$ .

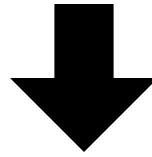


Update Q-values, by incorporating the new sample into a running average over samples:

$$Q(s, a) \leftarrow Q(s, a) + \frac{1}{k} [R(s, a, s') + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

# Q Learning

Collect a sample:  $s, a, s'$  and  $R(s, a, s')$ .



Update Q-values, by incorporating the new sample into a running average over samples:

$$Q(s, a) \leftarrow Q(s, a) + \frac{1}{k} [R(s, a, s') + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

↑

Can be replaced by  $\alpha$

# Q Learning

Collect a sample:  $s, a, s'$  and  $R(s, a, s')$ .

How shall we  
choose the action?

Update Q-values, by incorporating  
the new sample into a running average over samples:

$$Q(s, a) \leftarrow Q(s, a) + \frac{1}{k} [R(s, a, s') + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

# Exploration vs Exploitation

Collect a sample:  $s, a, s'$  and  $R(s, a, s')$ .

Initially, we may randomly pick the action  $a$  (exploration). Later, when we are more confident about the learned Q-values, we may follow the action based on Q-values (exploitation)

$$Q(s, a) \leftarrow Q(s, a) + \frac{1}{k} [R(s, a, s') + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$



# See you ...

<https://statnlp.org>

Hope to see many of you at NLP next Summer!