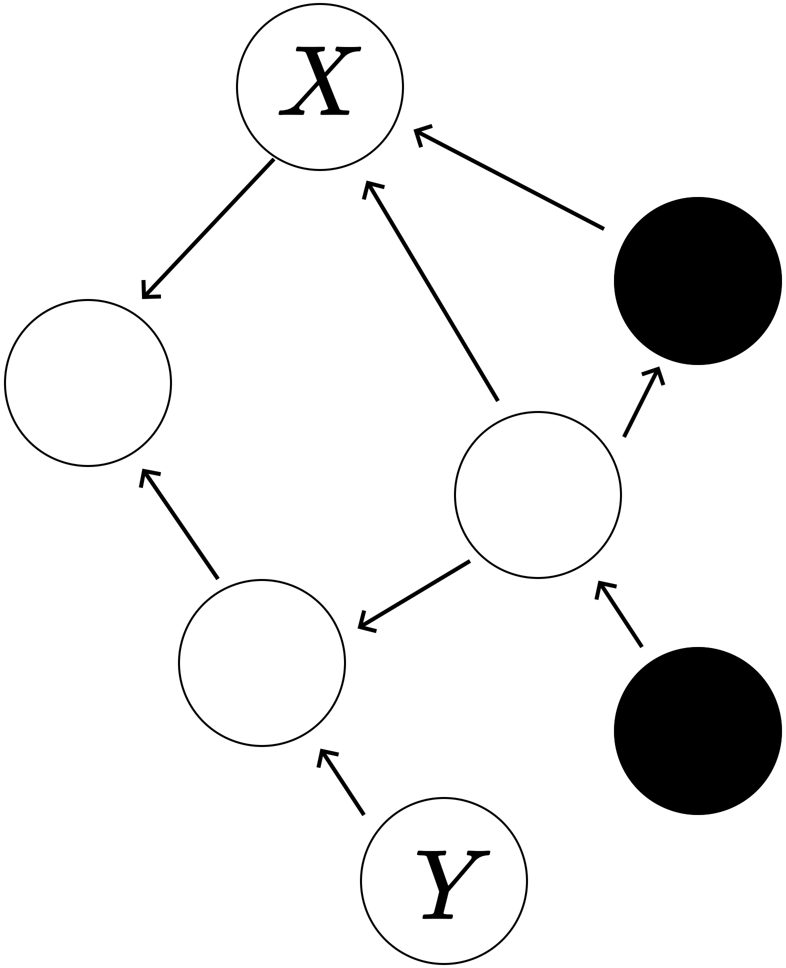
50.007
Machine Learning

Lu, Wei

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

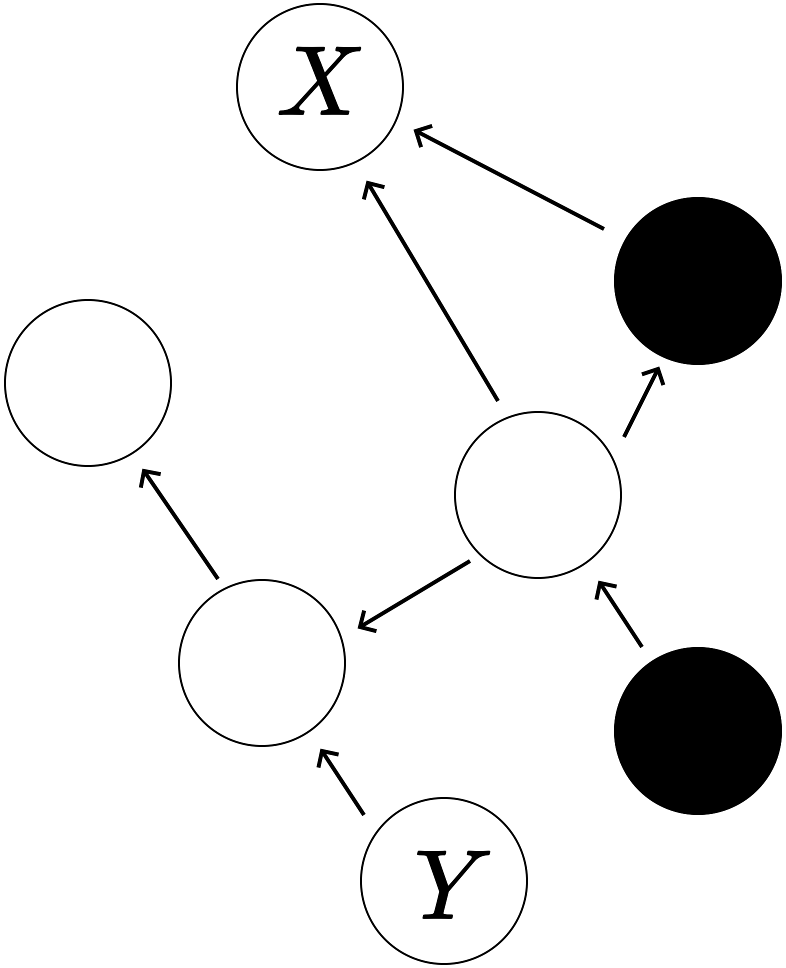
Reinforcement Learning (I)

Bayes' Ball Algorithm



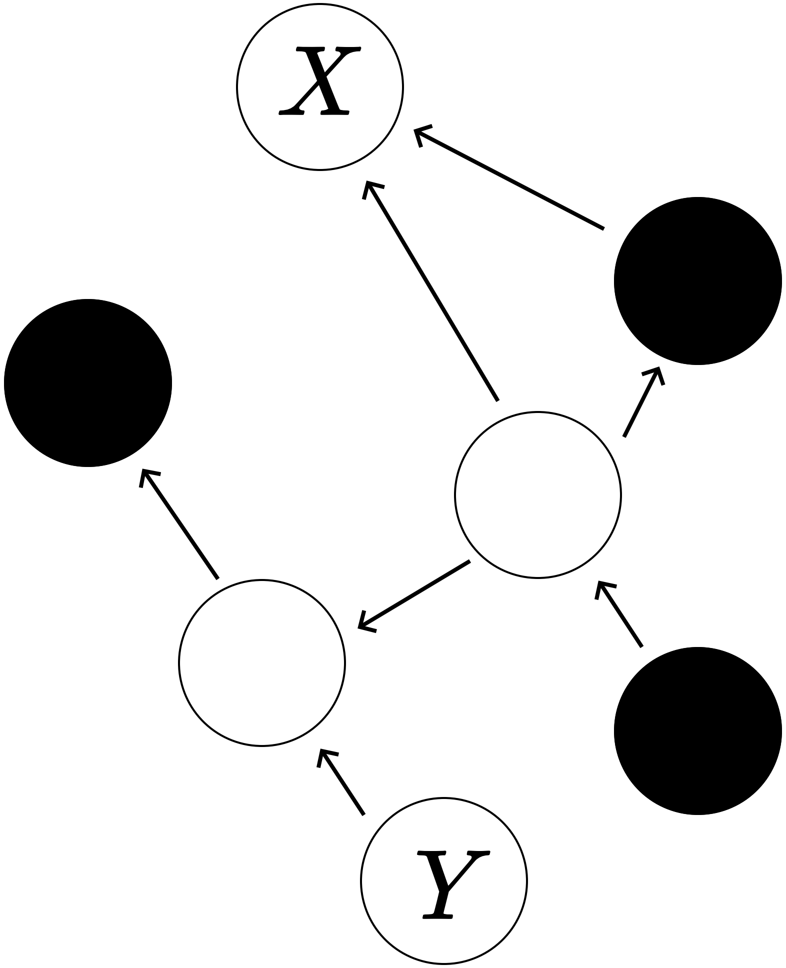
Are X and Y independent?

Bayes' Ball Algorithm



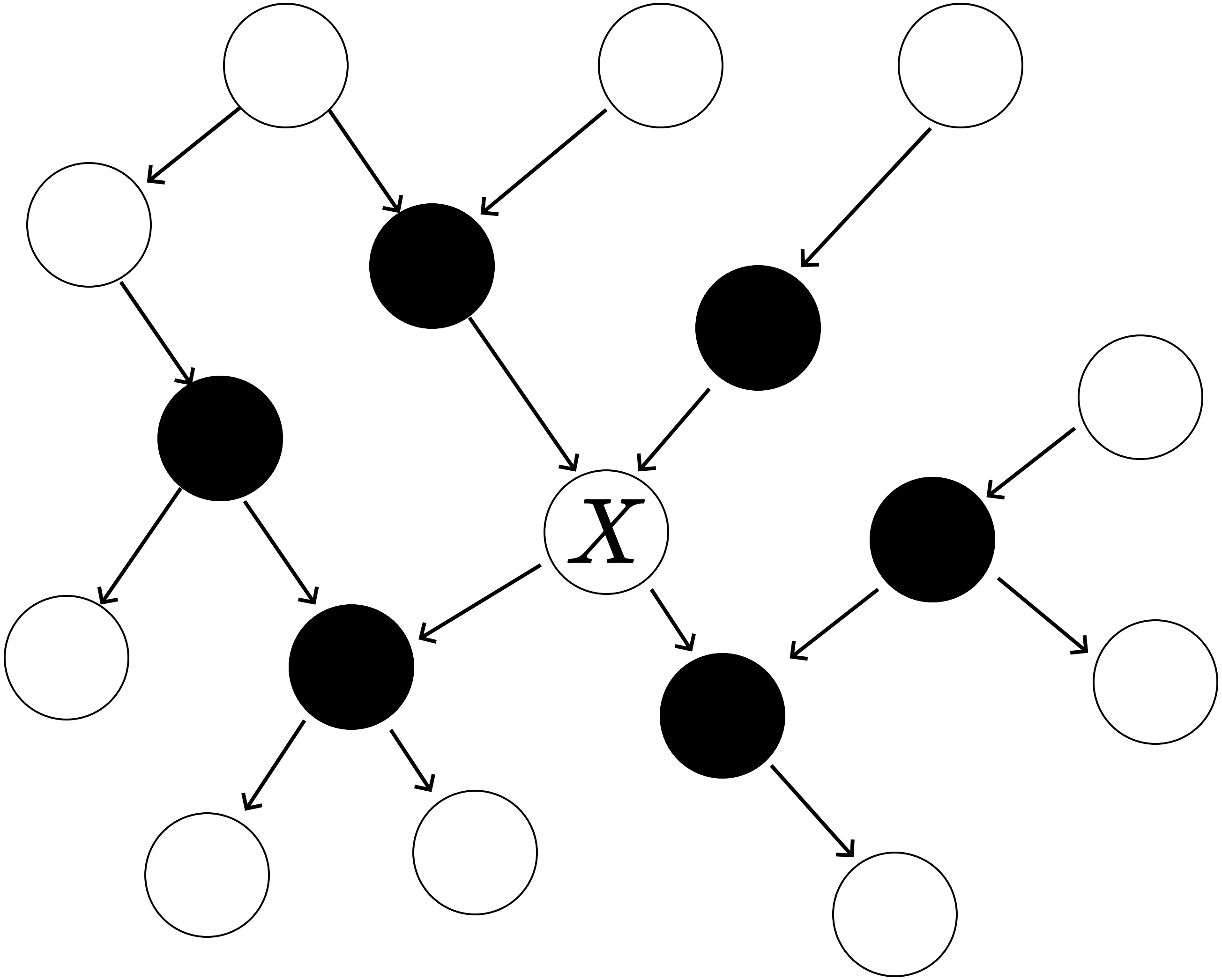
Are X and Y independent?

Bayes' Ball Algorithm



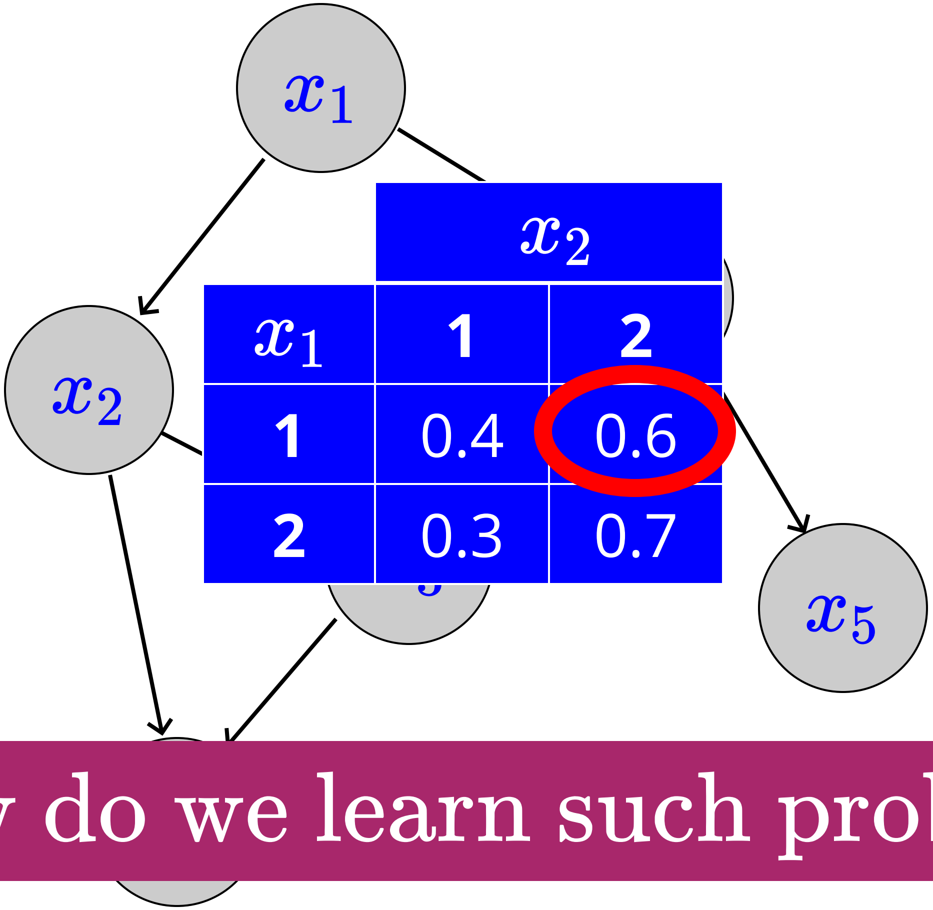
Are X and Y independent?

Markov Blanket



$$p(X|\mathbf{V}_{-X})$$

Bayesian Networks

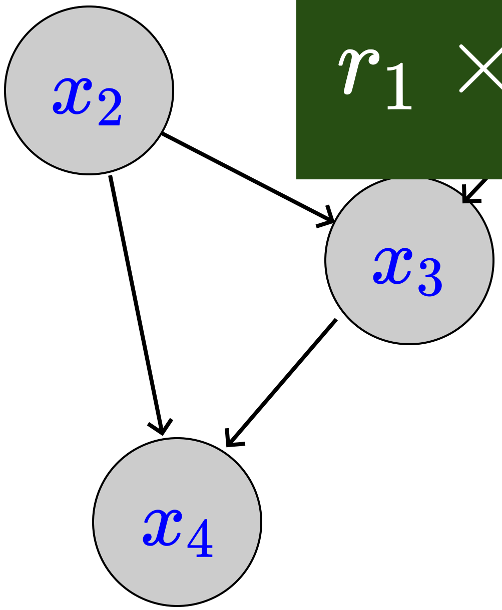


	x_2	
x_1	1	2
1	0.4	0.6
2	0.3	0.7

How do we learn such probability values?

$$p(x_1 = 1, x_2 = 2, x_3 = 1, x_4 = 3, x_5 = 5, x_6 = 2)$$
$$= p(x_1 = 1) \times p(x_2 = 2 | x_1 = 1) \times p(x_4 = 3 | x_1 = 1)$$
$$\times\, p(x_3 = 1 | x_2 = 2, x_4 = 3) \times p(x_6 = 2 | x_2 = 2, x_3 = 1) \times p(x_5 = 5 | x_4 = 3)$$

Bayesian Networks



$$r_1 \times r_2 \times (r_3 - 1)$$

$\in \{1, 2, \ldots, r_1\}$

$\in \{1, 2, \ldots, r_2\}$

$x_3 \in \{1, 2, \ldots, r_3\}$

$\ldots, r_4\}$

$\ldots, r_5\}$

		x_3			
x_1	x_2	1	2	...	r_3
1	1				
1	2				
...	...				
1	r_2				
2	1				
2	2				
...	...				
2	r_2				
...	...				
r_1	1				
r_1	2				
...	...				
r_1	r_2				

What is the number of free parameters involved in this table?

$(x_1, x_2$

$(x_3|x_1$

$x_1)$

Overview of ML

Supervised Learning

Unsupervised Learning

Overview of ML

Supervised Learning

Unsupervised Learning

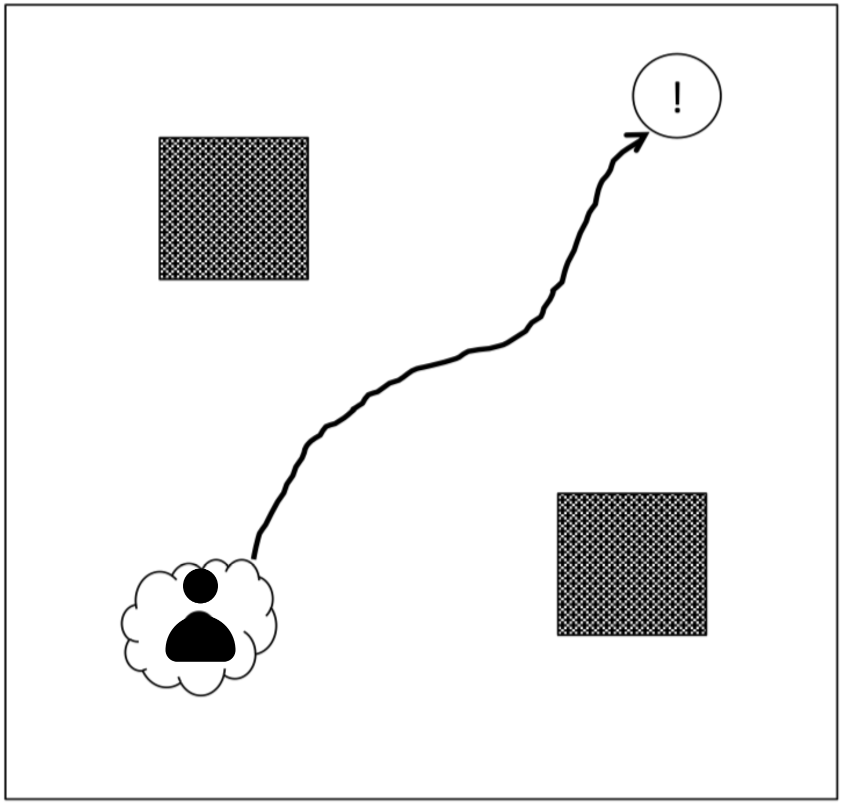
A set of states

A set of actions

$$f : S \to A$$

Reinforcement Learning

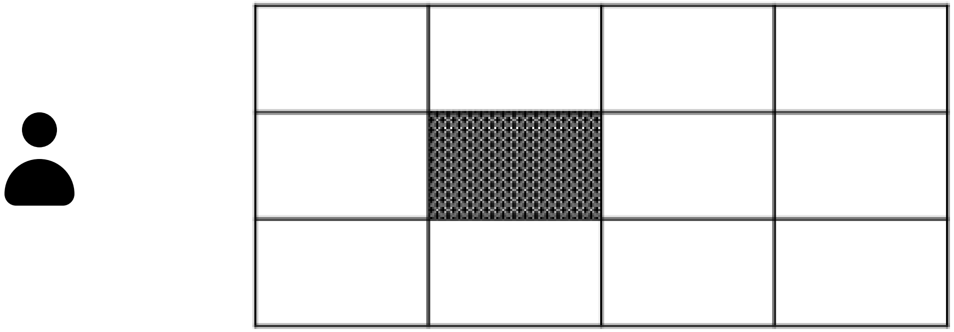
Learn How to Act



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

Block World

Environment

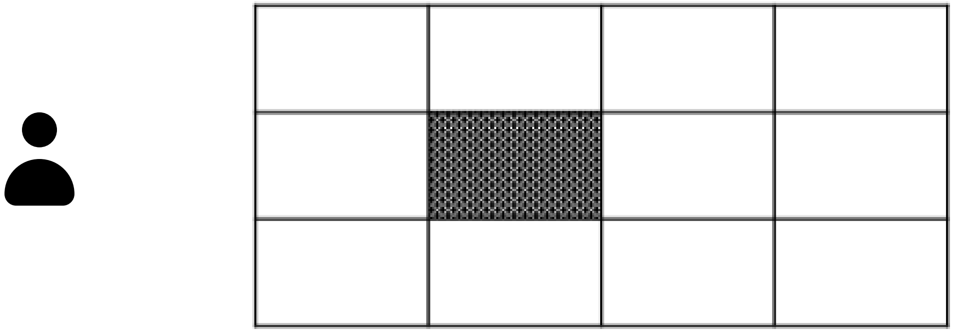


The robot can be at one block at any given time.

$$\text{There is a set of states } S$$

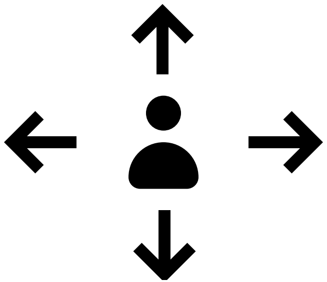
Block World

Environment

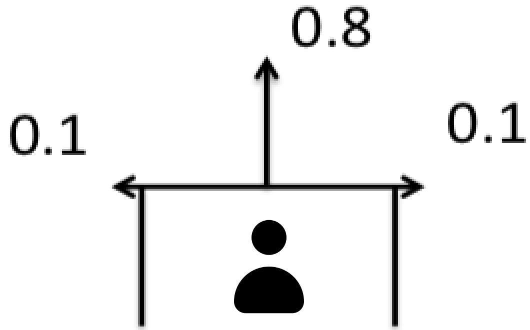


The robot can take an action from a set of predefined possible actions at each state.

$$\text{There is a set of actions } A$$



Block World
Transition Probabilities



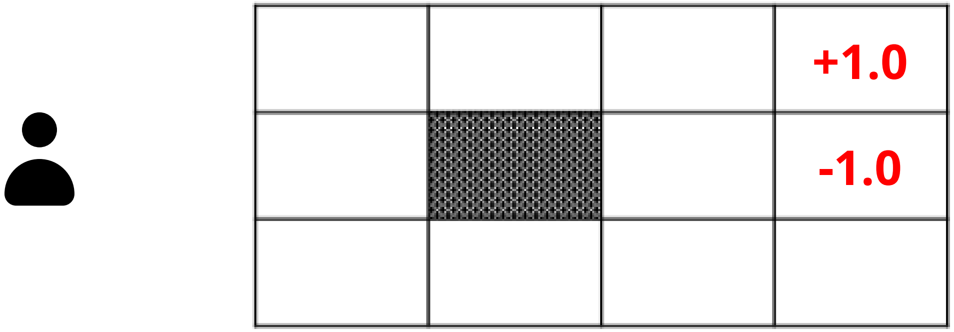
If the robot moves towards a particular direction, there is a 0.8 chance that it would reach the block in front, and there is a 0.1 chance to reach a state to its left (right).

A transition probability function

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

Block World

Rewards



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

Block World

Rewards

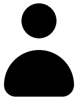
-0.6	+1.2	+0.1	+1.0
-0.1		-0.1	-1.0
+0.9	-0.7	-2.0	+1.3

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 $R(\underbrace{s}_{\text{old state}}, \overbrace{a}^{\text{action}}, \underbrace{s'}_{\text{new state}})$

Markov Decision Process



-0.6	+1.2	+0.1	+1.0
-0.1		-0.1	-1.0
+0.9	-0.7	-2.0	+1.3

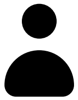
- a set of states S

- 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 $R(s')$)

Markov Decision Process

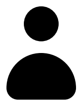


-0.6	+1.2	+0.1	+1.0
-0.1		-0.1	-1.0
+0.9	-0.7	-2.0	+1.3

How do we learn the policy?

- 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 $R(s')$)

Block World
Utility (Long Term Reward)



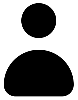
-0.6	+1.2	+0.1	+1.0
0.1		-0.1	-1.0
+0.9	-0.7	-2.0	+1.3

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

Will this be a good way of defining long term reward?

Block World

Utility (Long Term Reward)



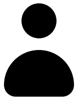
-0.6	→ +1.2	← +0.1	+1.0
0.1		-0.1	-1.0
+0.9	-0.7	-2.0	+1.3

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

Will this be a good way of defining long term reward?

Block World

Utility (Long Term Reward)

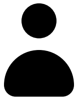


-0.6	+1.2	+0.1	+1.0
0.1		-0.1	-1.0
+0.9	-0.7	-2.0	+1.3

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

discount factor

Block World
Utility (Long Term Reward)



-0.6	+1.2	+0.1	+1.0
0.1		-0.1	-1.0
+0.9	-0.7	-2.0	+1.3

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

$$\frac{R_{min}}{1-\gamma} = \sum_{t=0}^{\infty} \gamma^t \underbrace{R_{min}}_{\text{Smallest Reward}} \leq U([s_0, s_1, s_2, \ldots]) \leq \sum_{t=0}^{\infty} \gamma^t \underbrace{R_{max}}_{\text{Largest Reward}} = \frac{R_{max}}{1-\gamma}$$

Lower Bound

Upper Bound

Block World
Learning the Policy

-0.01	-0.01	-0.01	+1.0
-0.01		-0.01	-1.0
-0.01	-0.01	-0.01	-0.01

How to learn the policy?

→	→	→	+1.0
↓		←	-1.0
↑	→	↑	←

An example policy

Policy, Value, Q-Value

$\pi(s)$

a particular *policy* that specifies

the action we should take in state s.

Policy, Value, Q-Value

$\pi(s)$

a particular *policy* that specifies

the action we should take in state s.

$V^{\pi}(s)$

The *value* of state s under policy π.

It is the expected long-term reward of starting in state s

and act based on policy π thereafter.

Policy, Value, Q-Value

$\pi(s)$

a particular *policy* that specifies
the action we should take in state s.

$V^\pi(s)$

The *value* of state s under policy π.
It is the expected long-term reward of starting in state s
and act based on policy π thereafter.

$Q^\pi(s, a)$

The *Q-value* of state s and action a under policy π.
It is the expected long-term reward of starting in state s,
taking action a and acting based on policy π thereafter.

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 π^*

$Q^*(s, a)$

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

Policy, Value, Q-Value

$$V^*(s) = ??$$

$V^*(s)$

The *value* of state s under the optimal policy π^*

$Q^*(s, a)$

The Q-*value* of state s

and action a under the optimal policy π^*

Policy, Value, Q-Value

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

$V^*(s)$

The *value* of state s under the optimal policy π^*

$Q^*(s,a)$

The Q-*value* of state s

and action a under the optimal policy π^*

Policy, Value, Q-Value

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

$\pi^*(s)$

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 π^*

$Q^*(s,a)$

The Q-*value* of state s

and action a under the optimal policy π^*

Policy, Value, Q-Value

$$Q^*(s,a) = \ ??$$

$\pi^*(s)$

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 π^*

$Q^*(s,a)$

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

Policy, Value, Q-Value

$$Q^*(s,a) = \sum_{s'} \underbrace{T(s,a,s')}_{\boxed{\text{probability of next state}}} [\underbrace{R(s,a,s')}_{\boxed{\text{immediate reward}}} + \gamma \underbrace{V^*(s')}_{\boxed{\text{long-term reward}}}]$$

$\pi^*(s)$

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 π^*

$Q^*(s,a)$

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

Policy, Value, Q-Value

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

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

$\pi^*(s)$

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 π^*

$Q^*(s,a)$

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

Policy, Value, Q-Value

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

$\pi^*(s)$

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 π^*

$Q^*(s,a)$

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

Policy, Value, Q-Value

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

Updating Value

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

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

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

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

Question

How do we recover the optimal policy based on the values?

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

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

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

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

Policy, Value, Q-Value

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

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

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

Policy, Value, Q-Value

$$\pi^*(s) = ??$$

$\pi^*(s)$

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

$Q^*(s, a)$

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

Policy, Value, Q-Value

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

$\pi^*(s)$

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

$Q^*(s, a)$

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

Learning Optimal Policy

Step 1
Run Value Iteration Algorithm

Step 2
Calculate the Q values

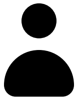
$$Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

Step 3
Find the optimal action for each state

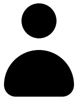
$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

Value Iteration

$$R(s) = -0.01$$



-0.01	-0.01	-0.01	+1.0
-0.01		-0.01	-1.0
-0.01	-0.01	-0.01	-0.01



			+1.0
			-1.0

Value Iteration
Learned Policy

-0.01	-0.01	-0.01	+1.0
-0.01		-0.01	-1.0
-0.01	-0.01	-0.01	-0.01

→	→	→	+1.0
↑		←	-1.0
↑	←	←	↓

Value Iteration

$$R(s) = -2.0$$

-2.0	**-2.0**	**-2.0**	**+1.0**
-2.0		**-2.0**	**-1.0**
-2.0	**-2.0**	**-2.0**	**-2.0**

			+1.0
			-1.0

Value Iteration

Learned Policy

-2.0	-2.0	-2.0	+1.0
-2.0		-2.0	-1.0
-2.0	-2.0	-2.0	-2.0

→	→	→	+1.0
↑		→	-1.0
→	→	→	↑

Value Iteration
Learned Policy

-2.0	-2.0	-2.0	+1.0
-2.0		-2.0	-1.0
-2.0	-2.0	-2.0	-2.0

Why?

→	→	→	+1.0
↑		→	-1.0
→	→	→	↑

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?

We will discuss Q-value iteration next time!

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