COMP47590: Advanced Machine Learning

Reinforcement Learning - Fundamentals

Dr. Brian Mac Namee

1

RL Fundamentals

Today we will look at

- Intelligent agents
- The fundamental building blocks of reinforcement learning
- Markov Decision Processes

2

Intelligent Agents

3

Intelligent Agents

In reinforcement learning we take an agent-based view on machine learning and talk about training an **intelligent agent**

- The **goal** of the agent is to complete the task as successfully as possible.
- Each attempt at the task is referred to as an **episode**.

4

Intelligent Agents

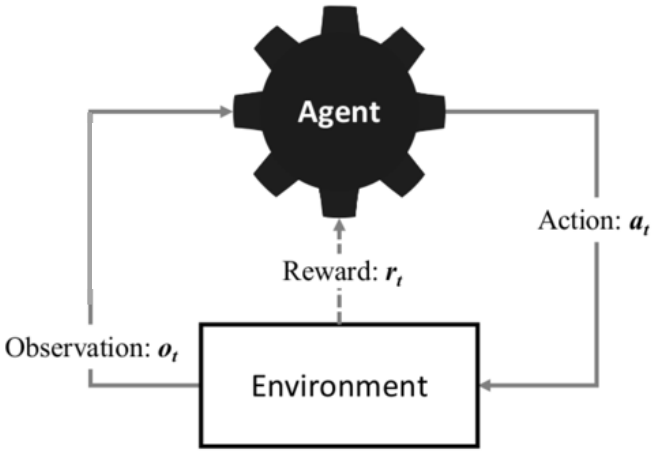
At any point in time, ***t***, the agent

- observes the current state of its environment, o_t;
- considers these observations to select an action, a_t;
- and takes this action, receiving an immediate reward signal, r_t, from the environment about whether this was a good or bad action to take.

$$(o_1, a_1, r_1), (o_2, a_2, r_2), (o_3, a_3, r_3), \ldots, (o_e, a_e, r_e)$$

5

Intelligent Agents



$$(o_1, a_1, r_1), (o_2, a_2, r_2), (o_3, a_3, r_3), \ldots, (o_e, a_e, r_e)$$

Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples and Case Studies , Second edition
John D. Kelleher, Brian Mac Namee, Aoife D'Arcy MIT Press, 2020

6

Histories & States

The sequence of observations, actions and rewards that precede any time-step, *t*, is referred to as a **history**, H_t

The agent makes a decisions at each time-step, *t*, about what action to take next on the basis of its current observations of the environment, o_t, and the history, H_t

agent consider the history before making any action

$$(o_1, a_1, r_1), (o_2, a_2, r_2), (o_3, a_3, r_3), \ldots, (o_e, a_e, r_e)$$

7

Histories & States

Maintaining long histories of actions, rewards, and observations is not very efficient so we collapse this information into a single representation, referred to as a **state**.

- The state at time-step *t*, s_t, should contain all the important information about the environment at that time-step, any important information from preceding time-steps, and any important information about the internal composition of the agent.

8

State Generation Function

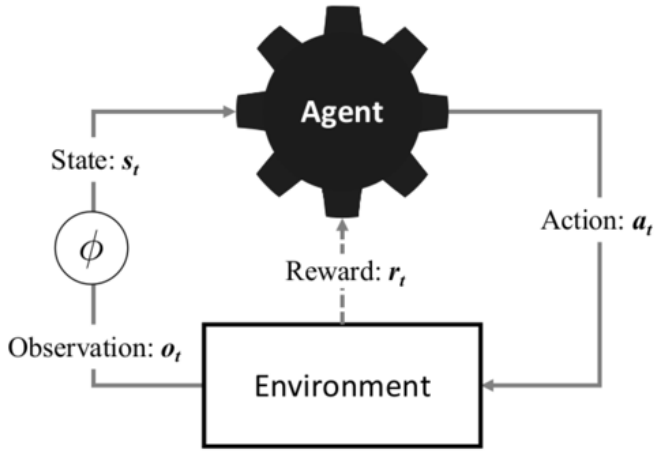
The observations made about the environment at time-step *t* are converted into a state, s_t, using a **state generation function, ϕ**

$$(s_1, a_1, r_1), (s_2, a_2, r_2), (s_3, a_3, r_3), \ldots, (s_e, a_e, r_e)$$

Designing good state representations is one of the *arts* of reinforcement learning.

9

Intelligent Agents



$$(s_1, a_1, r_1), (s_2, a_2, r_2), (s_3, a_3, r_3), \ldots, (s_e, a_e, r_e)$$

Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples and Case Studies , Second edition
John D. Kelleher, Brian Mac Namee, Aoife D'Arcy MIT Press, 2020

10

Intelligent Agents

In reinforcement learning the goal of the intelligent agent is to complete a task as *successfully* as possible.

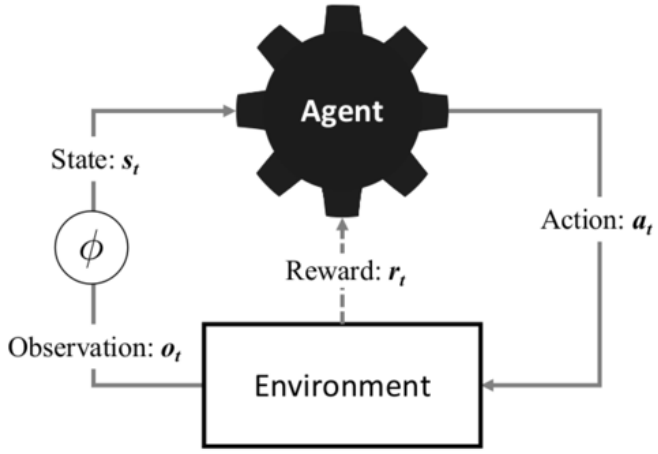
To frame the reinforcement learning problem this needs to be more formally defined - what does it mean to successfully complete a task?

11

THE BUILDING BLOCKS OF REINFORCEMENT LEARNING - LEARNING BY REWARD

12

Intelligent Agents



$$(s_1, a_1, r_1), (s_2, a_2, r_2), (s_3, a_3, r_3), \ldots, (s_e, a_e, r_e)$$

Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples and Case Studies , Second edition
John D. Kelleher, Brian Mac Namee, Aoife D'Arcy MIT Press, 2020

13

Sutton's Reward Hypothesis

"That all of what we mean by goals and purposes can be well thought of as maximization of the expected value of the cumulative sum of a received scalar signal (reward)."

Sutton's reward hypothesis

The fundamental idea underpinning reinforcement learning is that the only goal of an intelligent agent is to maximize **cumulative reward** across an episode

$$G = r_t + r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_e$$

"Reinforcement Learning An Introduction", 2nd Edition, Richard S. Sutton and Andrew G. Barto, MIT Press, 2018.
www.incompleteideas.net/book/the-book-2nd.html

14

The goal of reinforcement learning is to find the best policy that maximizes rewards over time.

Reward

That *intelligent* behavior can be driven by the singular goal of maximizing return is a bold statement

- Reward is often delayed, and the real value of an action often only becomes apparent later.
- Early moves in a game of chess
- Eating cake

It has been shown repeatedly, however, that it is in fact possible to learn sophisticated, long-term behaviors using the maximization of cumulative reward alone.

The second *art* of reinforcement learning is the design of effective reward functions.

15

Success

In reinforcement learning the goal of the intelligent agent is to complete a task as *successfully* as possible.

To frame the reinforcement learning problem this needs to be more formally defined - what does it mean to successfully complete a task?

Question: How would you measure success of a taxi driver agent?

Petar Milošević, CC BY-SA 4.0, via Wikimedia Commons

16

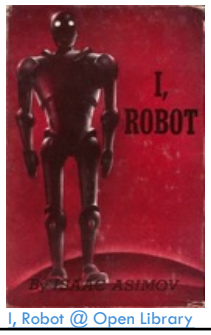
Success

In reinforcement learning the goal of the intelligent agent is to complete a task as *successfully* as possible.

To frame the reinforcement learning problem this needs to be more formally defined - what does it mean to successfully complete a task?

Question: How would you measure success of a taxi driver agent?

Petar Milošević, CC BY-SA 4.0, via Wikimedia Commons



I, Robot @ Open Library

17

Policies

policy is essentially the strategy or rulebook that the agent follows to decide what actions to take in different situations (states)

The key decision making component of a reinforcement learning agent is referred to as a **policy**, which is simply a mapping from states to actions

policy pi greco

$$a_t = \pi(s_t)$$

We can also define a policy in probabilistic terms

$$P(A_t = a \mid S_t = s) = \pi(S_t = s)$$

The policy can be thought of as a simple lookup table that records the action that should be taken in every state

Reinforcement learning problems can be framed as an effort to learn this table directly – **WE WILL NOT LOOK AT THESE APPROACHES**

18

It can be deterministic (one action per state) or probabilistic (a probability distribution over actions).

The goal of reinforcement learning is to learn the best policy that maximizes rewards over time.

Policies

Policies can also be encoded as a rule used to choose an action from those available in a particular state – **WE WILL LOOK AT THIS KIND OF APPROACH FIRST**

like masturbation

We could use a **greedy action selection policy** that always takes the action that gives the highest reward.

- This ignores the fact that sometimes reward is delayed
- An action that gives a low immediate reward could lead to states that give large rewards later

Instead we use a **value function**

to fix the problems of greedy policies

19

Value Function

The value function helps the agent think about long-term rewards

A **value function** returns the cumulative reward that an agent can expect to earn if it starts from a particular state, s_t, and follows a policy, π, all the way to the end of an episode.

state value function

$$V_\pi(s_t) = E_\pi[r_t + r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_e \mid s_t]$$

The value function returns the **expected cumulative reward** that an agent will earn if it follows policy π starting from state s_t

20

Value Function

Alternatively an **action-value function** returns the cumulative reward that an agent can expect to earn if it takes an action a_t in state s_t and then continues to select actions using policy, π, to the end of an episode.

The action-value function

$$Q_\pi(s_t, a_t) = E_\pi[r_t + r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_e \mid s_t, a_t]$$

The action-value function outputs the **expected return** of pursuing action a_t in state s_t

21

Yes, the action-value function is a specific type of a value function. The difference is that while the value function considers only states, the action-value function also considers actions taken from those states.

Discounted Return

One extra consideration is whether we should consider expected future rewards to be as valuable as immediate rewards

Usually we don't, and instead implement **discounted return**

$$G_\gamma = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \ldots + \gamma^{e-t} r_e$$

γ is a **discount rate** and is in [0,1]

- This implements an exponential discounting - future expected rewards have less and less expected value

22

Action Value Function

To use discounted return the action-value function is restated

$$Q_\pi(s_t, a_t) = E_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \ldots + \gamma^{e-t} r_e \mid s_t, a_t]$$

Discounting makes intuitive sense, makes some of the maths associated with reinforcement learning more straightforward, and is widely used in reinforcement learning.

23

Summary

In reinforcement learning an **intelligent agent** inhabiting an **environment** attempts to achieve a **goal** by taking a sequence of **actions** to move it between **states**

On completion of each action the agent receives an immediate scalar **reward** indicating whether the outcome of the action was positive or negative, and to what degree.

The degree to which an agent has achieved a goal is measured only by the **cumulative rewards** it has received from each action taken in pursuit of that goal.

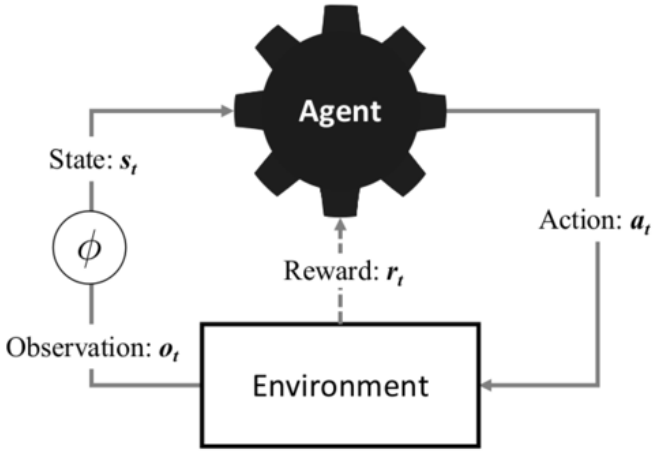
To choose which action to take in a given state the agent uses a **policy**.

Policies rely on being able to asses the **expected return** of taking an action in a particular state - an **action-value function** can be used to calculate this.

24

the discounted return is a way to calculate the total future rewards the agent expects to receive, but with a twist: rewards that are further in the future are worth less than rewards that are closer in time.

Intelligent Agents



$$(s_1, a_1, r_1), (s_2, a_2, r_2), (s_3, a_3, r_3), \ldots, (s_e, a_e, r_e)$$

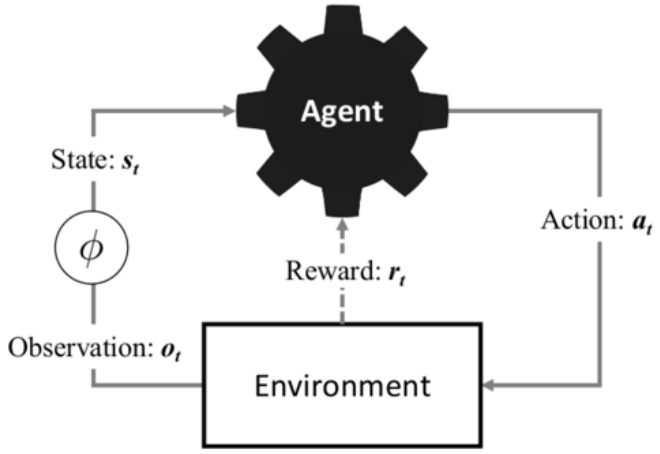
Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples and Case Studies , Second edition
John D. Kelleher, Brian Mac Namee, Aoife D'Arcy MIT Press, 2020

25

MARKOV DECISION PROCESSES

26

Intelligent Agents



$$(s_1, a_1, r_1), (s_2, a_2, r_2), (s_3, a_3, r_3), \ldots, (s_e, a_e, r_e)$$

Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples and Case Studies , Second edition
John D. Kelleher, Brian Mac Namee, Aoife D'Arcy MIT Press, 2020

27

Markov Decision Processes

Markov decision processes (MDPs) are an attractive mathematical framework within which to reason about decision making scenarios in which outcomes are partly under the control of a decision maker, but also partly random.

- Used in financial modeling, robot control, modeling the flow of human conversation…

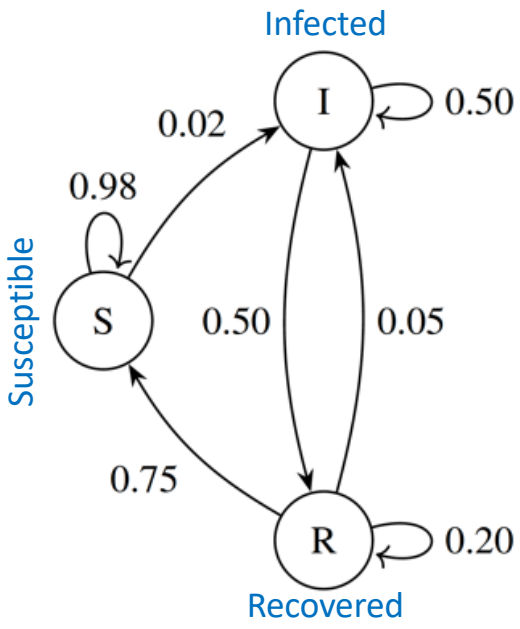
Ideal for reasoning about reinforcement learning.

28

Markov Processes

A **Markov process** is a more basic framework that does not include decision making

A Markov process models a discrete random process that transitions through a finite set of states, *S*



29

Markov Assumption

Markov processes rely on the **Markov assumption:** the probability of transitioning to a state at the next time-step relies only on the current state

so history doesnt matter in state transitioning

$$P(S_{t+1} \mid S_t, S_{t-1}, S_{t-2}, \ldots) = P(S_{t+1} \mid S_t)$$

Given the Markov assumption, we can write the probability of transitioning between two states

P is the transition probability

$$P(s_1 \rightarrow s_2) = P(S_{t+1} = s_2 \mid S_t = s_1)$$

30

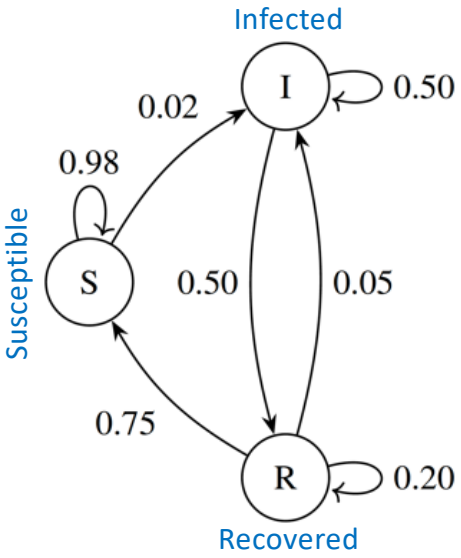
Transition Matrix

The full dynamics of a Markov process can be captured in a transition matrix

$$\mathcal{P} = \begin{bmatrix} P(s_1 \to s_1) & P(s_1 \to s_2) & \dots & P(s_1 \to s_n) \\ P(s_2 \to s_1) & P(s_2 \to s_2) & \dots & P(s_2 \to s_n) \\ \vdots & \vdots & \ddots & \vdots \\ P(s_n \to s_1) & P(s_n \to s_2) & \dots & P(s_n \to s_n) \end{bmatrix}$$

31

Markov Process



P_a =
| P(s1 | s1, a) P(s2 | s1, a) ... P(sn | s1, a) |
| P(s1 | s2, a) P(s2 | s2, a) ... P(sn | s2, a) |
| ... ... |
| P(s1 | sn, a) P(s2 | sn, a) ... P(sn | sn, a) |

$$\mathcal{P} = \begin{array}{c} \\ \text{S} \\ \text{I} \\ \text{R} \end{array} \begin{array}{c} \begin{array}{ccc} \text{S} & \text{I} & \text{R} \end{array} \\ \begin{bmatrix} 0.98 & 0.02 & 0.00 \\ 0.00 & 0.50 & 0.50 \\ 0.75 & 0.05 & 0.20 \end{bmatrix} \end{array}$$

somma nella riga =1
non vale per colonna

32

Markov Decision Process

A **Markov decision process** (MDP) extends the Markov process by adding decision making and rewards.

We extend the Markov process to add actions from this set to the formulation of transition probabilities

Relying on the Markov assumption the probability of transitioning to a particular state depends only on the current state and the action just taken.

$$P(s_1 \xrightarrow{a} s_2) = P(S_{t+1} = s_2 \mid S_t = s_1, A_t = a)$$

33

Markov Decision Process

Each transition to a new state based on a particular action now also carries with it a reward

$$R(s_1 \xrightarrow{a} s_2) = E(r_t \mid S_t = s_1, S_{t+1} = s_2, A_t = a)$$

34

Example: Recycling Robot

A mobile robot has the job of collecting empty cans in an office environment

- The robot has sensors for detecting cans, and an arm and gripper that can pick them up and place them in an onboard bin
- The robot runs on a rechargeable battery
- The robot's control system has components for interpreting sensory information, for navigating, and for controlling the arm and gripper
- High-level decisions about how to search for cans are made by an agent based on the current charge level of the battery
  1. actively search for a can for a certain period of time
  2. remain stationary and wait for someone to bring it a can
  3. head back to its home base to recharge its battery

"Reinforcement Learning An Introduction", 2nd Edition, Richard S. Sutton and Andrew G. Barto, MIT Press, 2018. www.incompleteideas.net/book/the-book-2nd.html

35

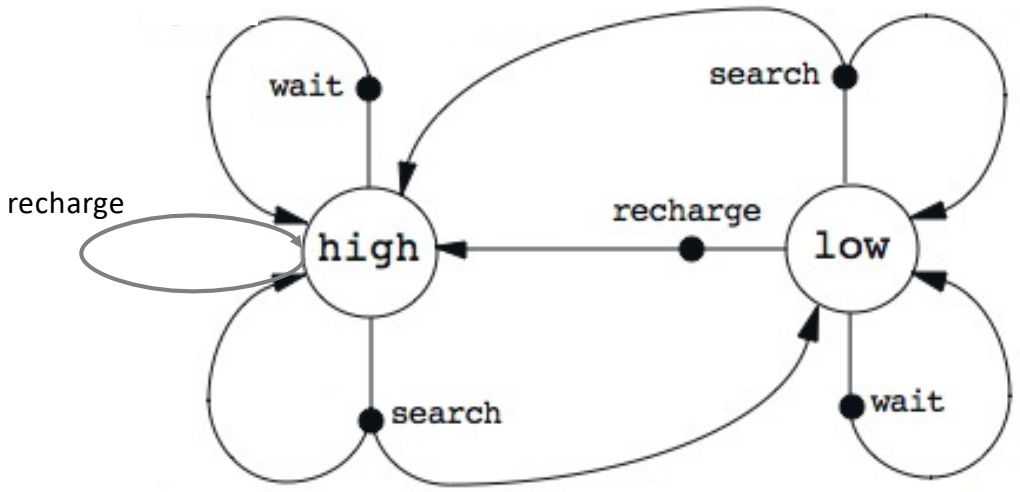
Example: Recycling Robot

Consider a finite MDP for the simple recycling robot

- At each step, robot has to decide whether it should
  1. actively **search** for a can
  2. **wait** for someone to bring it a can
  3. go to home base and **recharge**
- Searching is better but runs down the battery
  - If the robot runs out of power while searching it has to be rescued (which is bad)
- Decisions made on basis of current energy level: **`high`** or **`low`**
- **Reward** = number of cans collected

"Reinforcement Learning An Introduction", 2nd Edition, Richard S. Sutton and Andrew G. Barto, MIT Press, 2018. www.incompleteideas.net/book/the-book-2nd.html

36



"Reinforcement Learning An Introduction", 2nd Edition, Richard S. Sutton and Andrew G. Barto, MIT Press, 2018.
www.incompleteideas.net/book/the-book-2nd.html

37

Example: Recycling Robot

Modelling battery energy in the robot

- If the energy level is high, then a search can always be completed without risk of depleting the battery.
- A period of searching that begins with a high energy level leaves the energy level high with probability α and reduces it to low with probability 1 − α
- A period of searching undertaken when the energy level is low leaves it low with probability β and depletes the battery with probability 1 − β.
- If the robot must be rescued the battery is then recharged

"Reinforcement Learning An Introduction", 2nd Edition, Richard S. Sutton and Andrew G. Barto, MIT Press, 2018.
www.incompleteideas.net/book/the-book-2nd.html

38

Example: Recycling Robot

Modelling battery energy in the robot

- If the energy level is high, then a search can always be completed without risk of depleting the battery.
- A period o… a high energy level leave… probability α and reduc… 1 − α
- A period of searching undertaken when the energy level is low leaves it low with probability β and depletes the battery with probability 1 − β.
- If the robot must be rescued the battery is then recharged

Let's let α = 0.2 and β = 0.7

"Reinforcement Learning An Introduction", 2nd Edition, Richard S. Sutton and Andrew G. Barto, MIT Press, 2018. www.incompleteideas.net/book/the-book-2nd.html

39

Example: Recycling Robot

Modelling reward

- Each can collected by the robot counts as a unit reward,
- A reward of −3 results whenever the robot has to be rescued
- Let r_{search} and r_{wait}, with $r_{search} > r_{wait}$, respectively denote the expected reward while searching and waiting.

No cans can be collected during a run home for recharging or when being rescued

"Reinforcement Learning An Introduction", 2nd Edition, Richard S. Sutton and Andrew G. Barto, MIT Press, 2018. www.incompleteideas.net/book/the-book-2nd.html

40

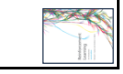
Example: Recycling Robot

Modelling reward

- Each can collected by the robot counts as a unit reward,
- A reward (ot has to be rescued
- Let r_{search} a ectively denote th ning and waiting.

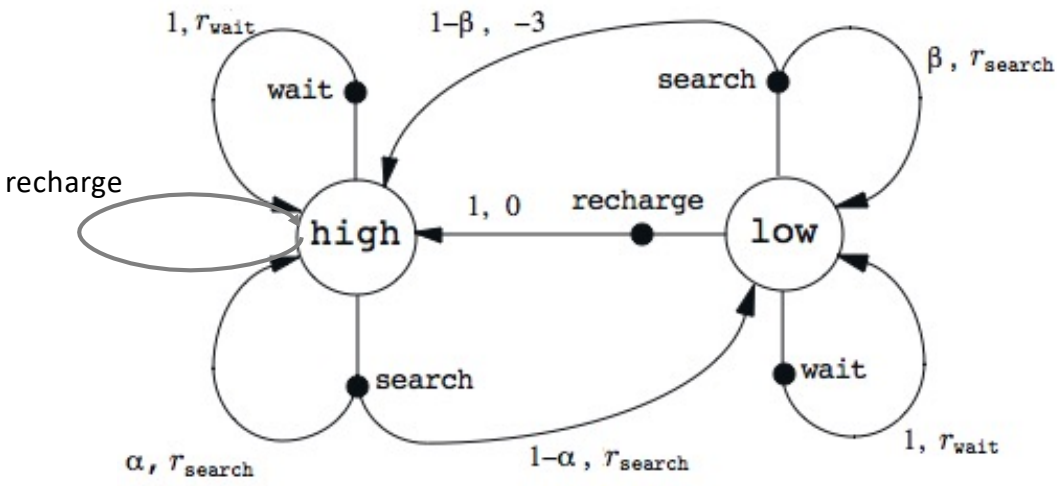
Let's let $r_{search} = 5$ and $r_{wait} = 1$

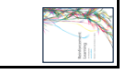
No cans can be collected during a run home for recharging or when being rescued



41

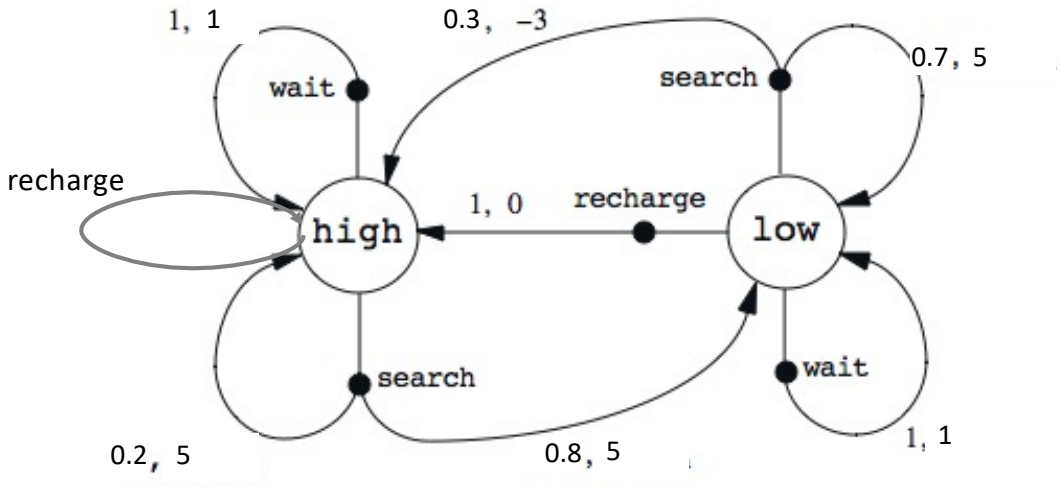
Example: Recycling Robot





42

Example: Recycling Robot



"Reinforcement Learning An Introduction", 2nd Edition, Richard S. Sutton and Andrew G. Barto, MIT Press, 2018. www.incompleteideas.net/book/the-book-2nd.html

43

Example: Recycling Robot

$$\mathcal{P}^{search} = \begin{bmatrix} P(\text{LOW} \xrightarrow{search} \text{LOW}) & P(\text{LOW} \xrightarrow{search} \text{HIGH}) \\ P(\text{HIGH} \xrightarrow{search} \text{LOW}) & P(\text{HIGH} \xrightarrow{search} \text{HIGH}) \end{bmatrix} = \begin{bmatrix} \beta & 1-\beta \\ 1-\alpha & \alpha \end{bmatrix}$$

$$\mathcal{P}^{wait} = \begin{bmatrix} P(\text{LOW} \xrightarrow{wait} \text{LOW}) & P(\text{LOW} \xrightarrow{wait} \text{HIGH}) \\ P(\text{HIGH} \xrightarrow{wait} \text{LOW}) & P(\text{HIGH} \xrightarrow{wait} \text{HIGH}) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathcal{P}^{recharge} = \begin{bmatrix} P(\text{LOW} \xrightarrow{recharge} \text{LOW}) & P(\text{LOW} \xrightarrow{recharge} \text{HIGH}) \\ P(\text{HIGH} \xrightarrow{recharge} \text{LOW}) & P(\text{HIGH} \xrightarrow{recharge} \text{HIGH}) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

44

Example: Recycling Robot

$$\mathcal{P}^{search} = \begin{bmatrix} P(\text{LOW} \xrightarrow{search} \text{LOW}) & P(\text{LOW} \xrightarrow{search} \text{HIGH}) \\ P(\text{HIGH} \xrightarrow{search} \text{LOW}) & P(\text{HIGH} \xrightarrow{search} \text{HIGH}) \end{bmatrix} = \begin{bmatrix} 0.7 & 0.3 \\ 0.8 & 0.2 \end{bmatrix}$$

$$\mathcal{P}^{wait} = \begin{bmatrix} P(\text{LOW} \xrightarrow{wait} \text{LOW}) & P(\text{LOW} \xrightarrow{wait} \text{HIGH}) \\ P(\text{HIGH} \xrightarrow{wait} \text{LOW}) & P(\text{HIGH} \xrightarrow{wait} \text{HIGH}) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathcal{P}^{recharge} = \begin{bmatrix} P(\text{LOW} \xrightarrow{recharge} \text{LOW}) & P(\text{LOW} \xrightarrow{recharge} \text{HIGH}) \\ P(\text{HIGH} \xrightarrow{recharge} \text{LOW}) & P(\text{HIGH} \xrightarrow{recharge} \text{HIGH}) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

45

Example: Recycling Robot

$$\mathcal{R}^{search} = \begin{bmatrix} R(\text{LOW} \xrightarrow{search} \text{LOW}) & R(\text{LOW} \xrightarrow{search} \text{HIGH}) \\ R(\text{HIGH} \xrightarrow{search} \text{LOW}) & R(\text{HIGH} \xrightarrow{search} \text{HIGH}) \end{bmatrix} = \begin{bmatrix} r_{search} & -3 \\ r_{search} & r_{search} \end{bmatrix}$$

$$\mathcal{R}^{wait} = \begin{bmatrix} R(\text{LOW} \xrightarrow{wait} \text{LOW}) & R(\text{LOW} \xrightarrow{wait} \text{HIGH}) \\ R(\text{HIGH} \xrightarrow{wait} \text{LOW}) & R(\text{HIGH} \xrightarrow{wait} \text{HIGH}) \end{bmatrix} = \begin{bmatrix} r_{wait} & r_{wait} \\ r_{wait} & r_{wait} \end{bmatrix}$$

$$\mathcal{R}^{recharge} = \begin{bmatrix} R(\text{LOW} \xrightarrow{recharge} \text{LOW}) & R(\text{LOW} \xrightarrow{recharge} \text{HIGH}) \\ R(\text{HIGH} \xrightarrow{recharge} \text{LOW}) & R(\text{HIGH} \xrightarrow{recharge} \text{HIGH}) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

46

Example: Recycling Robot

$$\mathcal{R}^{search} = \begin{bmatrix} R(\text{LOW} \xrightarrow{search} \text{LOW}) & R(\text{LOW} \xrightarrow{search} \text{HIGH}) \\ R(\text{HIGH} \xrightarrow{search} \text{LOW}) & R(\text{HIGH} \xrightarrow{search} \text{HIGH}) \end{bmatrix} = \begin{bmatrix} 5 & -3 \\ 5 & 5 \end{bmatrix}$$

$$\mathcal{R}^{wait} = \begin{bmatrix} R(\text{LOW} \xrightarrow{wait} \text{LOW}) & R(\text{LOW} \xrightarrow{wait} \text{HIGH}) \\ R(\text{HIGH} \xrightarrow{wait} \text{LOW}) & R(\text{HIGH} \xrightarrow{wait} \text{HIGH}) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$\mathcal{R}^{recharge} = \begin{bmatrix} R(\text{LOW} \xrightarrow{recharge} \text{LOW}) & R(\text{LOW} \xrightarrow{recharge} \text{HIGH}) \\ R(\text{HIGH} \xrightarrow{recharge} \text{LOW}) & R(\text{HIGH} \xrightarrow{recharge} \text{HIGH}) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

47

Example: Recycling Robot

s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	r_{search}
high	search	low	$1-\alpha$	r_{search}
low	search	high	$1-\beta$	-3
low	search	low	β	r_{search}
high	wait	high	1	r_{wait}
high	wait	low	0	r_{wait}
low	wait	high	0	r_{wait}
low	wait	low	1	r_{wait}
low	recharge	high	1	0
low	recharge	low	0	0.

"Reinforcement Learning An Introduction", 2nd Edition, Richard S. Sutton and Andrew G. Barto, MIT Press, 2018.
www.incompleteideas.net/book/the-book-2nd.html

48

BELLMAN EQUATIONS

49

Bellman Equations

An MDP does not tells us anything about what actions the agent should take to be most successful.

However, the **action-value function** can be expressed in terms of the components of an MDP to do just this.

50

Action Value Function

An **action-value function** returns the cumulative reward (or **expected return**) that an agent can expect to earn if it takes an action a_t in state s_t and then continues to select actions using policy, π, to the end of an episode.

$$Q_\pi(s_t, a_t) = E_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \ldots + \gamma^{e-t} r_e \mid s_t, a_t]$$

51

Bellman Equations

Before presenting the action value function in its new MDP form formally we can state the intuition behind it:

- We can calculate the value of taking a particular action in a given state as the **expected reward** for taking the action plus the value of the state that the agent arrives in after taking that action

52

Bellman Equations

This states the same thing formally

$$Q_\pi(s_t, a_t) = \sum_{s_{t+1}} P(s_t \xrightarrow{a_t} s_{t+1}) \left[R(s_t \xrightarrow{a_t} s_{t+1}) + \gamma \sum_{a_{t+1}} \pi(s_{t+1}, a_{t+1}) Q_\pi(s_{t+1}, a_{t+1}) \right]$$

This is one of the **Bellman Equations** first introduced by Richard Bellman in the 1950s as part of very early work on reinforcement learning.

Let's look at where it came from!

53

Bellman Equations

To restate the action value function in terms of the components of an MDP we start by reorganising it a little

$$Q_\pi(s_t, a_t) = E_\pi \left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots + \gamma^3 r_\infty \mid s_t, a_t \right]$$

54

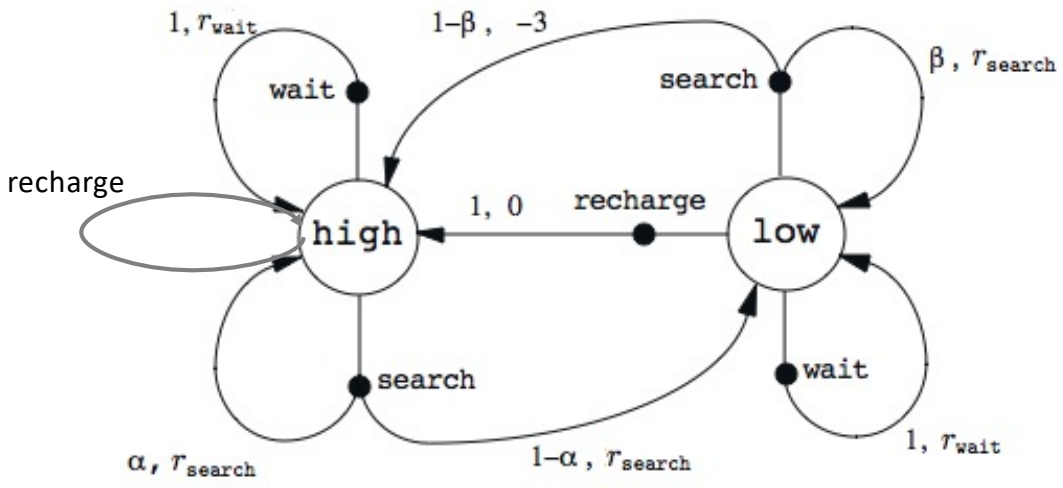
Bellman Equations

To restate the action value function in terms of the components of an MDP we start by reorganising it a little

$$
\begin{aligned}
Q_\pi(s_t, a_t) &= E_\pi \left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots + \gamma^3 r_\infty \mid s_t, a_t \right] \\
&= E_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t, a_t \right] \\
&= E_\pi \left[r_t + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t, a_t \right]
\end{aligned}
$$

55

Example: Recycling Robot



"Reinforcement Learning An Introduction", 2nd Edition, Richard S. Sutton and Andrew G. Barto, MIT Press, 2018.
www.incompleteideas.net/book/the-book-2nd.html

56

Bellman Equations

To add the components of an MDP we explicitly represent the uncertainty associated with state transitions.

$$Q_\pi(s_t, a_t) = \sum_{s_{t+1}} P(s_t \xrightarrow{a_t} s_{t+1}) \left[R(s_t \xrightarrow{a_t} s_{t+1}) + \gamma E_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_{t+1} \right] \right]$$

57

Bellman Equations

$$Q_\pi(s_t, a_t) = \sum_{s_{t+1}} P(s_t \xrightarrow{a_t} s_{t+1}) \left[R(s_t \xrightarrow{a_t} s_{t+1}) + \gamma E_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_{t+1} \right] \right]$$

To calculate the expected return of taking action a_t from state s_t we can calculate a weighted sum across the expected returns that the agent could receive in every state, s_{t+1}, that the agent could reach after taking action a_t in state s_t.

The weights in this weighted sum are the probabilities of the state transitions

58

Bellman Equations

This equation still includes the expected return essentially as it was stated before.

By adding an explicit reference to action a_{t+1} we can arrive at a very elegant recursive definition of the action-value function.

First the expected return can be written as a sum across the expected returns of all possible actions a_{t+1} that could be taken in state s_{t+1}

$$Q_\pi(s_t, a_t) = \sum_{s_{t+1}} P(s_t \xrightarrow{a_t} s_{t+1}) \left[R(s_t \xrightarrow{a_t} s_{t+1}) + \gamma \sum_{a_{t+1}} E_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_{t+1}, a_{t+1} \right] \right]$$

59

Bellman Equations

Remember that we can define the policy π as a function that returns a probability distribution over the set of possible actions that can be taken from a state

$$Q_\pi(s_t, a_t) = \sum_{s_{t+1}} P(s_t \xrightarrow{a_t} s_{t+1}) \left[R(s_t \xrightarrow{a_t} s_{t+1}) + \gamma \sum_{a_{t+1}} \pi(s_{t+1}, a_{t+1}) Q_\pi(s_{t+1}, a_{t+1}) \right]$$

This is one of the **Bellman Equations** and states that the **expected return** from taking action a_t in the state s_t is the **expected reward** for taking that action plus the **expected return** from all of the subsequent actions if the agent continues to follow the policy π.

60

This equation allows us to compute the optimal action-value function. If we solve this equation, we can find the best possible strategy for the agent.

Bellman Equations

It can be shown that for any MDP an optimal policy, π^*, exists, where following π^* will lead to returns as good as or better than the returns that would be earned following any other policy.

Using this idea the **Bellman optimality equation** for action-value functions can be written

$$Q_*(s_t, a_t) = \sum_{s_{t+1}} P(s_t \xrightarrow{a_t} s_{t+1}) \left[R(s_t \xrightarrow{a_t} s_{t+1}) + \gamma \max_{a_{t+1}} Q_*(s_{t+1}, a_{t+1}) \right]$$

61

Bellman Equations

$$Q_*(s_t, a_t) = \sum_{s_{t+1}} P(s_t \xrightarrow{a_t} s_{t+1}) \left[R(s_t \xrightarrow{a_t} s_{t+1}) + \gamma \max_{a_{t+1}} Q_*(s_{t+1}, a_{t+1}) \right]$$

There are approaches that can be used to directly solve the Bellman optimality equation to give an optimal policy

In modern reinforcement learning iterative approaches that calculate approximate solutions are typically used.

In the next lectures we will describe one of the most important of these, **temporal-difference learning**.

62

What is this equation really saying?
The bellman equation is saying that the value of taking an action right now is equal to(how good it is):The reward we get immediately.Plus the discounted future rewards we expect to receive if we keep following the policy.

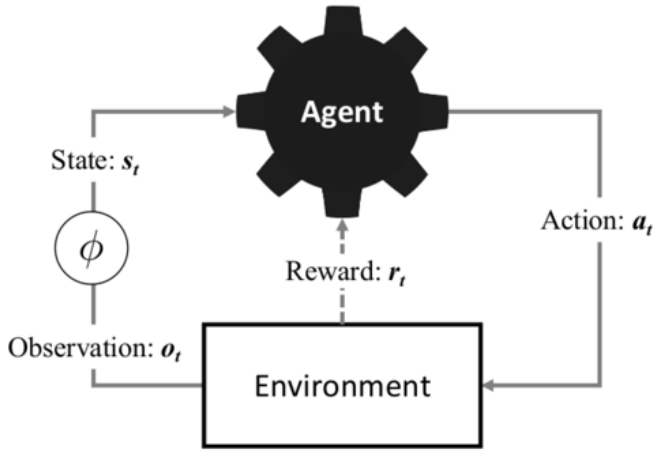
This is like making decisions in life:Imagine you are deciding whether to study for an exam or play video games.If you study now, you might not get an immediate reward (it's boring), but in the future, you will pass your exam and get a better job.The Bellman Equation helps balance short-term pleasure vs long-term success.

Qπ(s_t, a_t) = P(s_t s_{t+1} | a_t) [R(s_t s_{t+1}) + γ π(s_{t+1}, a_{t+1}) Qπ(s_{t+1}, a_{t+1})]
Let's break it down into simple parts:Qπ(s_t, a_t) is the value of taking action a_t in state s_t while following policy π.This tells us how good it is to take action a_t in s_t.The sum over s_{t+1} means we consider all possible future states that could result from our action.In other words, we are averaging over all possible situations we might end up in.P(s_t s_{t+1} | a_t) is the probability that taking action a_t in s_t leads to state s_{t+1}.This accounts for randomness in the environment.R(s_t s_{t+1}) is the immediate reward for going from s_t to s_{t+1}.This tells us how much reward we get right away.γ is the discount factor.This determines how much we care about future rewards.If γ is close to 1, we care a lot about future rewards.If γ is close to 0, we only care about immediate rewards.
The sum over a_{t+1} means we consider all possible actions in the future state s_{t+1}.This accounts for the fact that our future choices will also impact our total reward.

SUMMARY

63

Summary



$$(s_1, a_1, r_1), (s_2, a_2, r_2), (s_3, a_3, r_3), \ldots, (s_e, a_e, r_e)$$

Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples and Case Studies , Second edition
John D. Kelleher, Brian Mac Namee, Aoife D'Arcy MIT Press, 2020

64

Summary

In reinforcement learning an **intelligent agent** inhabiting an **environment** attempts to achieve a **goal** by taking a sequence of **actions** to move it between **states**

On completion of each action the agent receives an immediate scalar **reward** indicating whether the outcome of the action was positive or negative, and to what degree.

The degree to which an agent has achieved a goal is measured only by the **cumulative rewards** it has received from each action taken in pursuit of that goal.

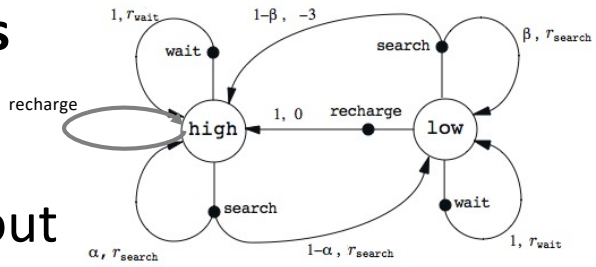
To choose which action to take in a given state the agent uses a **policy**.

Policies rely on being able to assess the **expected return** of taking an action in a particular state - an **action-value function** can be used to calculate this.

65

Markov Decision Processes

Markov decision processes (MDPs) are an attractive mathematical framework within which to reason about decision making scenarios in which outcomes are partly under the control of a decision maker, but also partly random.



Ideal for reasoning about reinforcement learning.

66

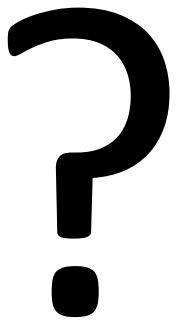
Bellman Equations

The **Bellman Equation** we use states that the **expected return** from taking action a_t in the state s_t is the **expected reward** for taking that action plus the **expected return** from all of the subsequent actions if the agent continues to follow the policy π.

$$Q_\pi(s_t, a_t) = \sum_{s_{t+1}} P(s_t \xrightarrow{a_t} s_{t+1}) \left[R(s_t \xrightarrow{a_t} s_{t+1}) + \gamma \sum_{a_{t+1}} \pi(s_{t+1}, a_{t+1}) Q_\pi(s_{t+1}, a_{t+1}) \right]$$

67

Questions



68