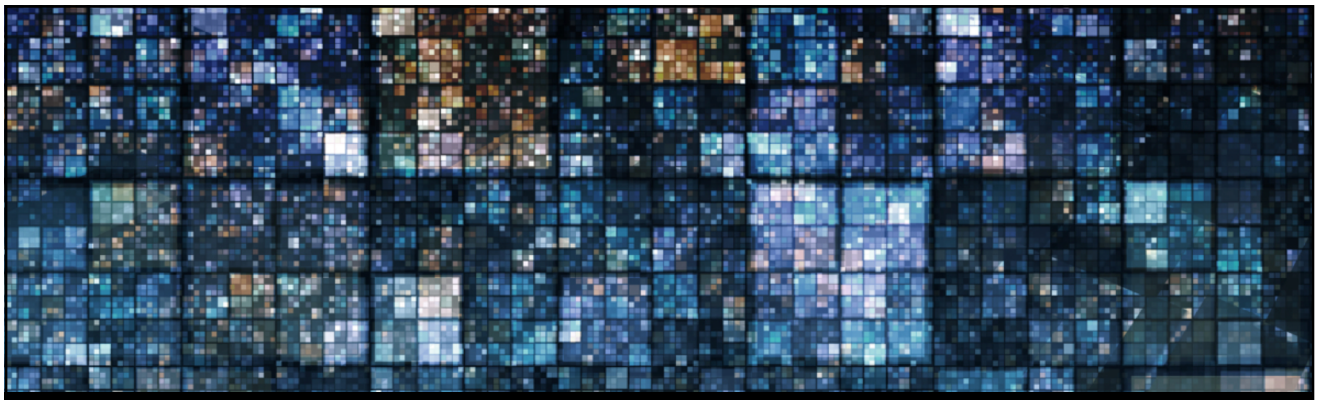
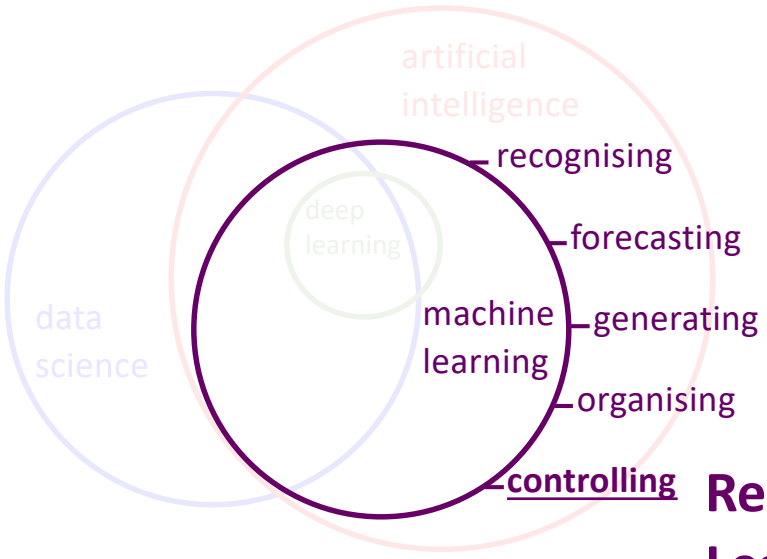


COMP47590
Advanced Machine Learning
RL - Fundamentals

Dr. Brian Mac Namee

Information

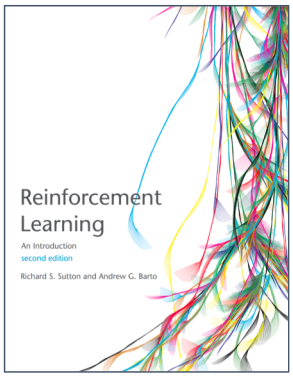
Email:	Brian.MacNamee@ucd.ie
Course Materials:	All material posted on UCD CS moodlehttps://csmoodle.ucd.ie/moodle/course/view.php?id=663 Enrolment key **UCDAdvML2019**



Reinforcement Learning

Reference Book

Much of the content in this section will follow "Reinforcement Learning An Introduction", 2nd Edition, Richard S. Sutton and Andrew G. Barto, MIT Press, 2018 (to appear)



A legitimate online version of this book is available at:

www.incompleteideas.net/book/the-book-2nd.html

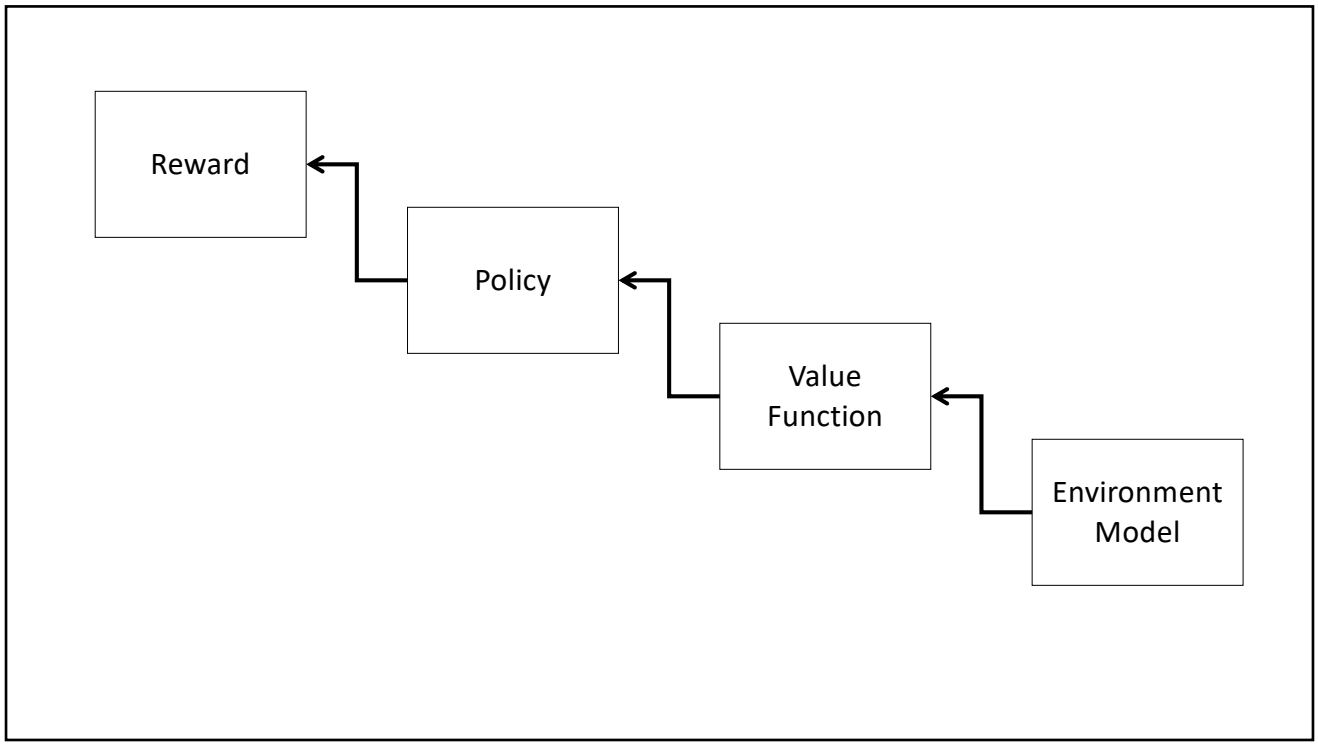
Reinforcement Learning

Reinforcement Learning: An Introduction , Second edition, in progress
Richard S. Sutton and Andrew G. Barto, MIT Press, 2017
www.incompleteideas.net/book/the-book-2nd.html

Reinforcement Learning

We have introduced the key ideas in reinforcement learning

- an agent
- an environment
- a policy
- a reward signal
- a value function
- a model of the environment (optional)



RL Fundamentals

Today we will look at 2 RL fundamentals

- k-armed bandit problems
- Finite markov decision processes (MDPs)

k-ARMED BANDIT PROBLEM

One-armed bandit slot machine pays out a prize at a fixed probability

***k*-armed bandit** slot machine pays out a prize at different fixed probabilities for each arm

Multi-armed Bandit Problem

You are faced repeatedly with a choice among k different arms on the sot machine

After each choice you receive a numerical reward (0 or some positive value) chosen from a stationary probability distribution that depends on the arm you selected

Your objective is to maximize the expected total reward over some time period, for example, over 1000 arm pulls

k-armed Bandit Problem

On each of an infinite sequence of *time steps*, *t*=1, 2, 3, …, you choose an action A_t from *k* possibilities, and receive a real-valued *reward* R_t

The reward depends only on the action taken; it is identically, independently distributed (i.i.d.)

$$q_*(a) \doteq \mathbb{E}[R_t \mid A_t = a]$$

k-armed Bandit Problem

We denote the estimated value of action a at time step t as $Q_t(a)$

We would like $Q_t(a)$ to be close to $q*(a)$ as possible

Sample average method is a simple way to estimate $Q_t(a)$

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t}$$

k-armed Bandit Problem

We denote the estimated value of action a at time step t as $Q_t(a)$

We would like $Q_t(a)$ to be close to $q*(a)$ as possible

Sample average method is a simple way to estimate $Q_t(a)$

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t}$$

k-armed Bandit Problem

The simplest action selection rule is to always select the actions with the highest estimated value

$$A_t \doteq \arg\max_a Q_t(a)$$

This is referred to as **greedy** action selection

The Exploration/Exploitation Dilemma

If $A_t = A_t^*$ then you are **exploiting**

If $A_t \neq A_t^*$ then you are **exploring**

You can't do both, but you need to do both

ε-Greedy Action Selection

In greedy action selection, you always exploit

In ε-greedy, you are usually greedy, but with probability ε you instead pick an action at random

This is perhaps the simplest way to balance exploration and exploitation

The 10-armed Testbed

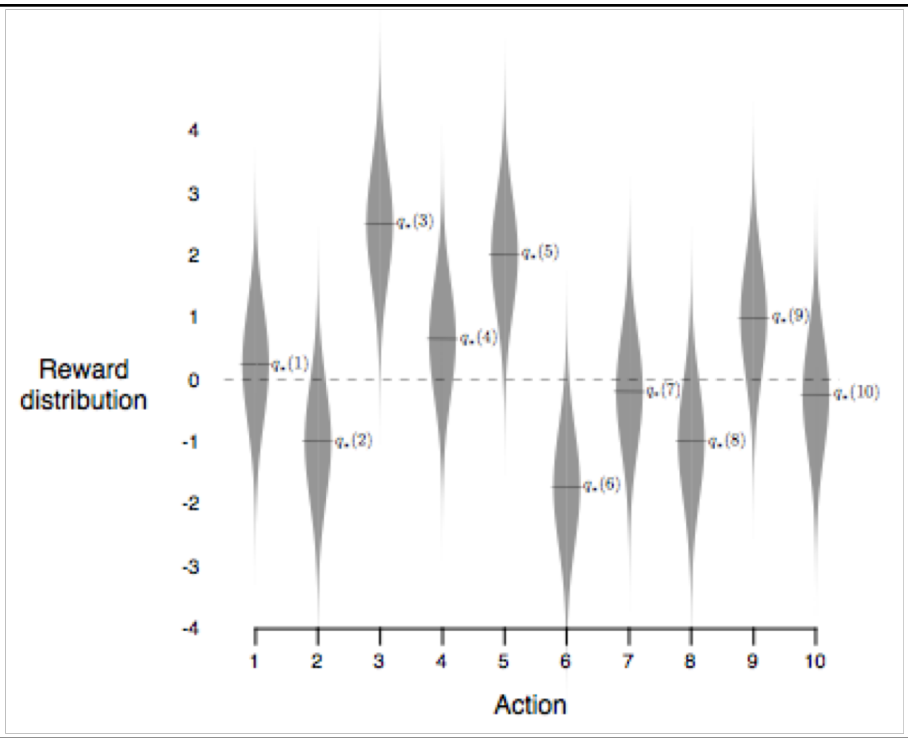
Design an artificial bandit problem

Define action values, $q_*(a)$, $a = 1, \ldots, 10$, according to a normal distribution $N(0, 1)$

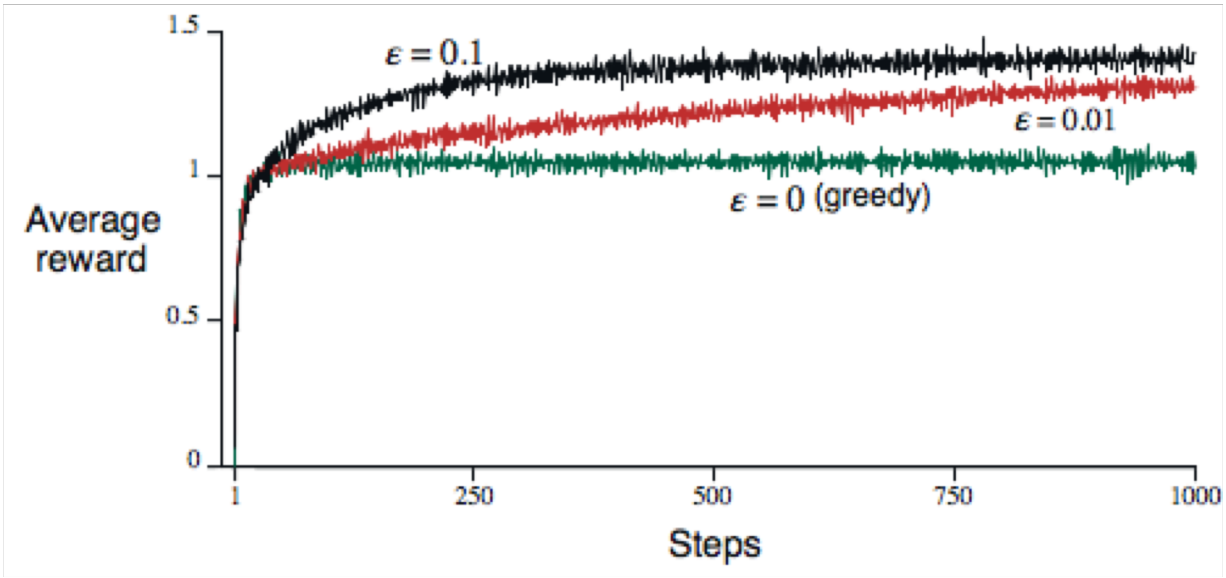
$$q_*(a) = N(0, 1)$$

At each time step t, select rewards, R_t, from a normal distribution $N(q_*(A_t), 1)$

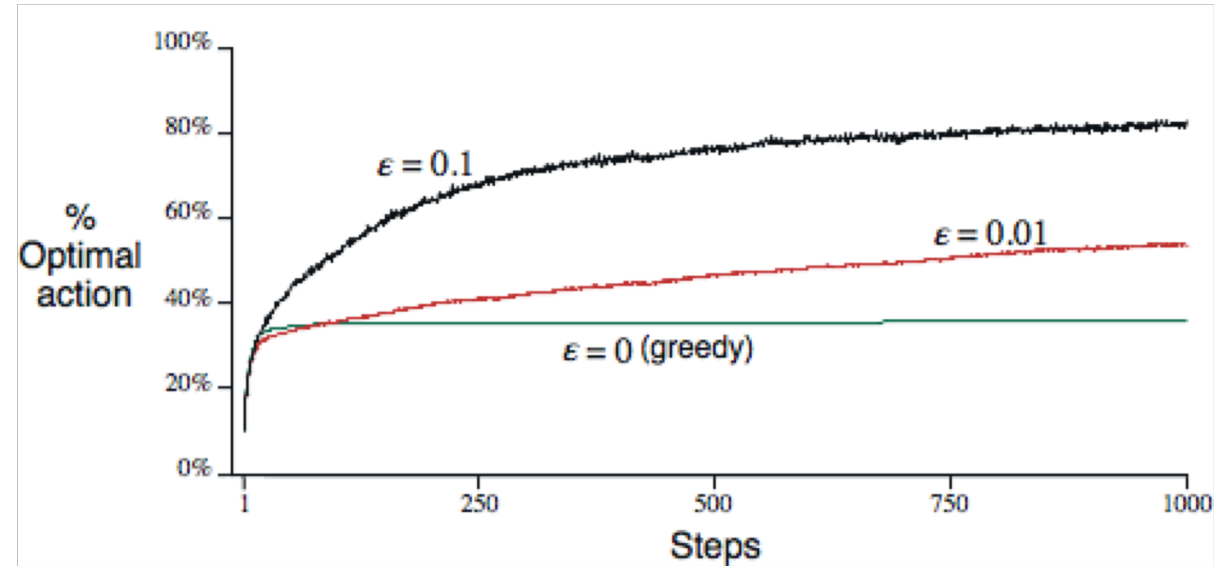
$$R_t = N(q_*(A_t), 1)$$



Reinforcement Learning: An Introduction , Second edition, in progress
Richard S. Sutton and Andrew G. Barto, MIT Press, 2017
www.incompleteideas.net/book/the-book-2nd.html



Reinforcement Learning: An Introduction , Second edition, in progress
Richard S. Sutton and Andrew G. Barto, MIT Press, 2017
www.incompleteideas.net/book/the-book-2nd.html



Reinforcement Learning: An Introduction , Second edition, in progress
Richard S. Sutton and Andrew G. Barto, MIT Press, 2017
www.incompleteideas.net/book/the-book-2nd.html

k-armed Bandit

From a reinforcement learning point of view the key components are:

- agent: the arm puller
- environment: the bandit machine
- a policy: greedy or ε-greedy action selection
- a reward signal: R_t
- a value function: $Q_t(a)$
- a model of the environment: not needed

NON-STATIONARY *k*-ARMED BANDIT PROBLEM

Non-stationary *k*-Armed Bandit Problem

In our previous discussion we assumed that the distribution of the rewards issued by each arm were stationary

There are many scenarios in which this is not the case

We can make a very simple modification to our value function

Non-stationary *k*-Armed Bandit Problem

We can rewrite the our value function to update an average Q_n of the n − 1 past rewards to be

$$Q_{n+1} \doteq Q_n + \alpha \Big[R_n - Q_n \Big]$$

where the step-size parameter $\alpha \in (0, 1]$ is constant

This is a specific form of a general update rule

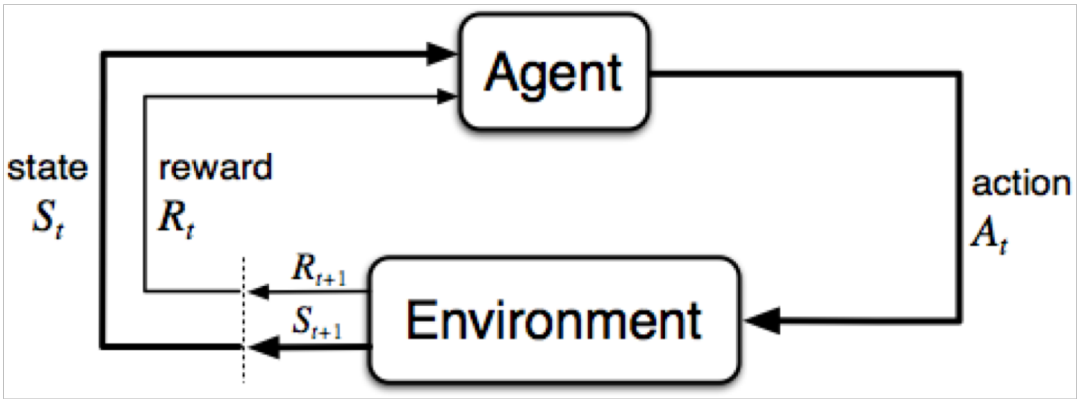
$$NewEstimate \leftarrow OldEstimate \; + \; StepSize \Big[Target - OldEstimate \Big]$$

FINITE MARKOV DECISION PROCESSES

Finite Markov Decision Processes

MDPs are a straightforward framing of the problem of learning from interaction to achieve a goal

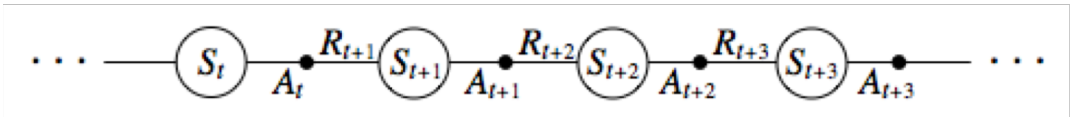
- the agent and environment interact at each of a sequence of discrete time steps, t=0,1,2,3,....2
- at each time step the agent receives some representation of the environment's state, $S_t \in S$,
- based on S_t the agent selects an action, $A_t \in A(s)$
- one time step later the agent receives a numerical reward, R_{t+1}, and finds itself in a new state, S_{t+1}



Reinforcement Learning: An Introduction , Second edition, in progress
Richard S. Sutton and Andrew G. Barto, MIT Press, 2017
www.incompleteideas.net/book/the-book-2nd.html

Finite Markov Decision Processes

The MDP and agent together give rise to a sequence or trajectory that begins like this:



Reinforcement Learning: An Introduction , Second edition, in progress
Richard S. Sutton and Andrew G. Barto, MIT Press, 2017
www.incompleteideas.net/book/the-book-2nd.html

Finite Markov Decision Processes

In a finite MDP, the sets of states, actions, and rewards ($\mathcal{S}$, $\mathcal{A}$, and $\mathcal{R}$) all have a finite number of elements

- The random variables R_t and S_t have well defined discrete probability distributions dependent only on the preceding state and action

The Markov Property

By "the state" at step *t*, we means whatever information is available to the agent at step *t* about its environment

- The state can include immediate "sensations," highly processed sensations, and structures built up over time from sequences of sensations
- State representation design is an important part of building reinforcement learning systems

The Markov Property

Ideally, a state should summarize past sensations so as to retain all "essential" information, this is the **Markov Property**:

$$\begin{aligned}
&\mathbf{Pr}\{R_{t+1} = r, S_{t+1} = s' \mid S_0, A_0, R_1, \ldots, S_{t-1}, A_{t-1}, R_t, S_t, A_t\} \\
&\quad = \mathbf{Pr}\{R_{t+1} = r, S_{t+1} = s' \mid S_t, A_t\} \\
&\quad = \; p(s', r|s, a)
\end{aligned}$$

Finite Markov Decision Processes

For particular values of these random variables, $s' \in \mathcal{S}$ and $r \in \mathcal{R}$, there is a probability of those values occurring at time t, given particular values of the preceding state and action:

$$p(s', r|s, a) \;\doteq\; \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s)$$

Finite Markov Decision Processes

The probabilities given by the four-argument function p completely characterize the dynamics of a finite MDP

From it, we can compute anything else one might want to know about the environment

Finite Markov Decision Processes

$$r(s,a) \doteq \mathbb{E}[R_t \mid S_{t-1}=s, A_{t-1}=a] = \sum_{r\in\mathcal{R}} r \sum_{s'\in\mathcal{S}} p(s',r\,|\,s,a)$$

$$p(s'\,|\,s,a) \doteq \Pr\{S_t=s' \mid S_{t-1}=s, A_{t-1}=a\} = \sum_{r\in\mathcal{R}} p(s',r\,|\,s,a)$$

$$r(s,a,s') \doteq \mathbb{E}[R_t \mid S_{t-1}=s, A_{t-1}=a, S_t=s'] = \sum_{r\in\mathcal{R}} r\,\frac{p(s',r\,|\,s,a)}{p(s'\,|\,s,a)}$$

Example: Recycling Robot

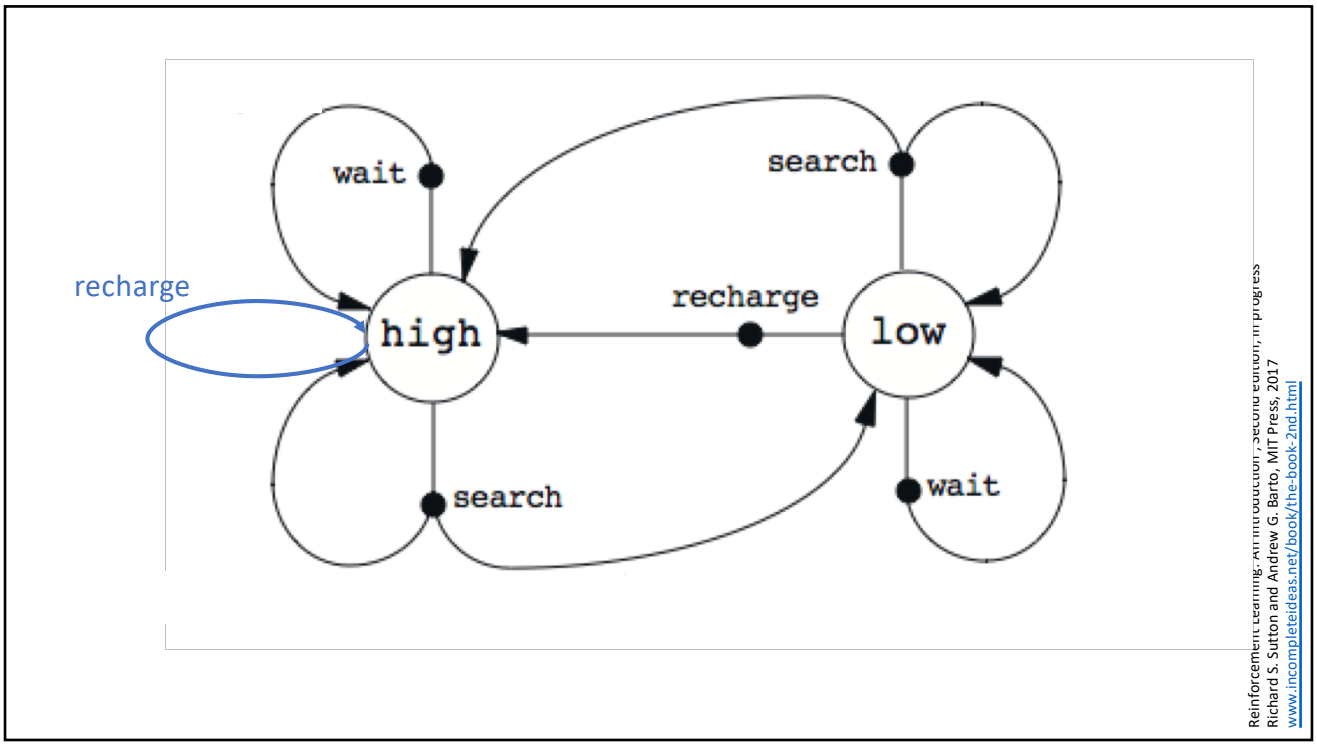
A mobile robot has the job of collecting empty cans in an office environment

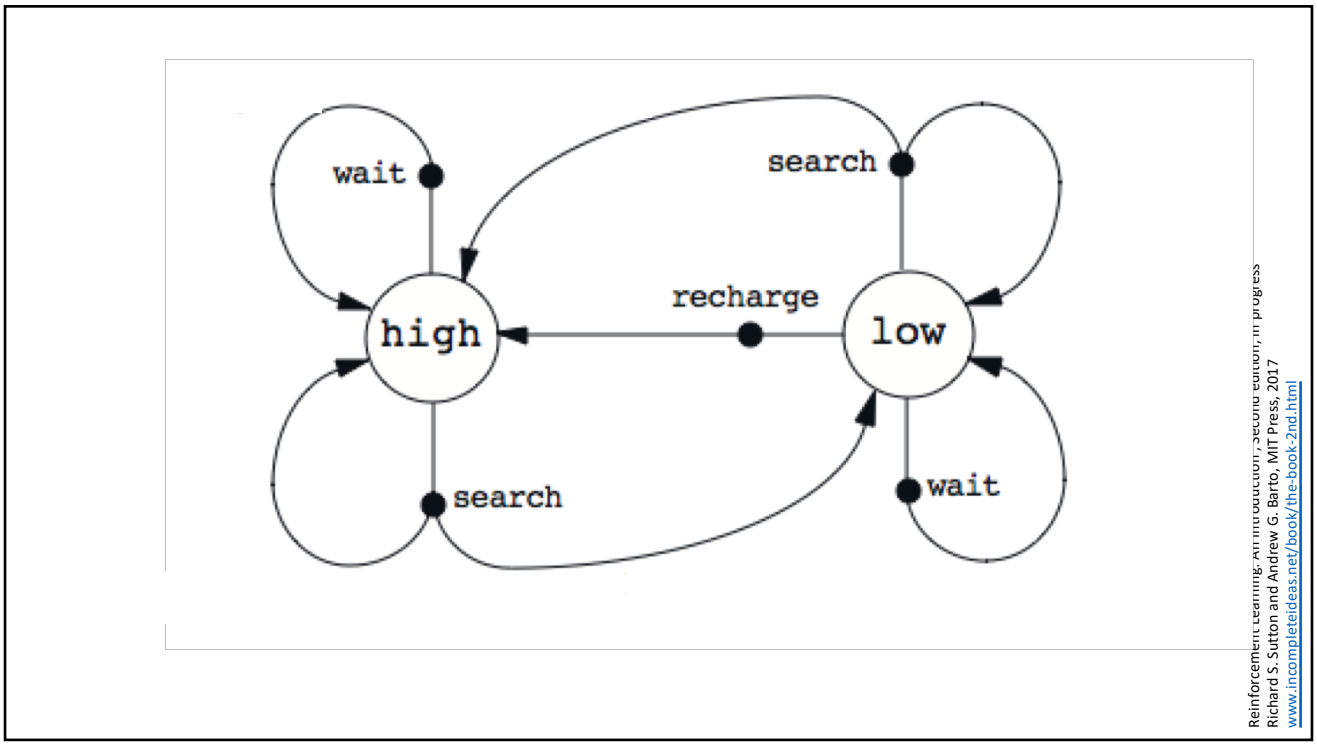
- The robot has sensors for detecting cans, and an arm and gripper that can pick them up and place them in an onboard bin
- The robot runs on a rechargeable battery
- The robot's control system has components for interpreting sensory information, for navigating, and for controlling the arm and gripper
- 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

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


wait
search
recharge
recharge
high
low
search
wait


wait
search
recharge
high
low
search
wait

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 $1 - \alpha$
- A period of searching undertaken when the energy level is low leaves it low with probability β and depletes the battery with probability $1 - \beta$.
- If the robot must be rescued the battery is then recharged

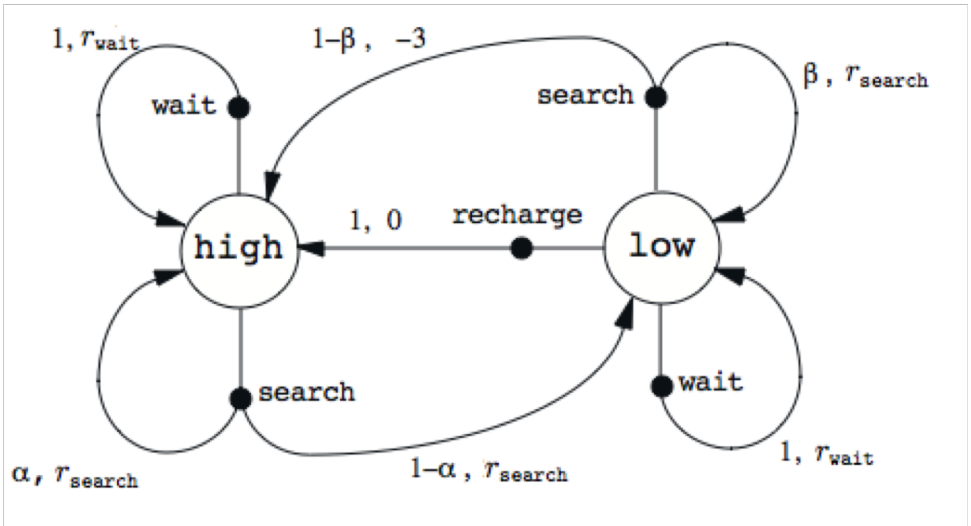
Example: Recycling Robot

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

Example: Recycling Robot



Reinforcement Learning: An Introduction , Second edition, in progress
Richard S. Sutton and Andrew G. Barto, MIT Press, 2017
www.incompleteideas.net/book/the-book-2nd.html

Example: Recycling Robot

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

Reinforcement Learning: An Introduction , Second edition, in progress
Richard S. Sutton and Andrew G. Barto, MIT Press, 2017
www.incompleteideas.net/book/the-book-2nd.html

What's Missing? A Policy

Policy at step t = π_t =

a mapping from states to action probabilities

$\pi_t(a \mid s)$ = probability that $A_t = a$ when $S_t = s$

Reinforcement learning methods specify how the agent changes its policy as a result of experience

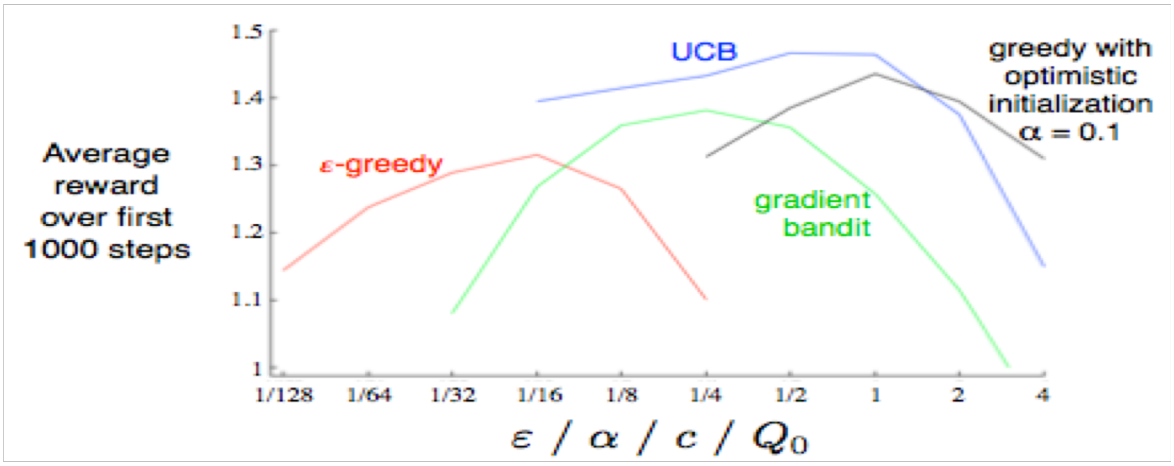
SUMMARY

Summary

We introduced multi-armed bandit problems as a simplified example of reinforcement learning

- agent: the arm puller
- environment: the bandit machine
- a policy: greedy or ε-greedy action selection
- a reward signal: R_t
- a value function: $Q_t(a)$
- a model of the environment: not needed

Other Bandit Algorithms



Summary

We introduced finite Markov decision processes as a fundamental idea in reinforcement learning

- the agent and environment interact at each of a sequence of discrete time steps, t=0,1,2,3,....2
- at each time step the agent receives some representation of the environment's state, $S_t \in S$,
- based on S_t the agent selects an action, $A_t \in A(s)$
- One time step later the agent receives a numerical reward, R_{t+1}, and finds itself in a new state, S_{t+1}

Questions