1

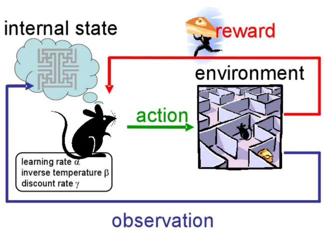
Outline

- What is Reinforcement Learning (RL)
- Examples
- Defining an RL problem
  - Markov Decision Processes
- Solving an RL problem
  - Dynamic Programming
  - Monte Carlo methods
  - Temporal-Difference learning

2

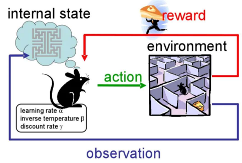
Reinforcement Learning (RL)

Learning (by operating in and interacting with the world [aka. environment]) to make good sequence of decisions under uncertainty



3

Reinforcement Learning (RL)

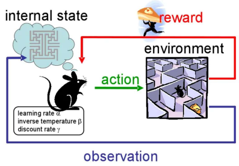


Learning (by operating in and interacting with the world [aka. environment]) to make good sequence of decisions under uncertainty

Repeat interactions with the world

4

Reinforcement Learning (RL)



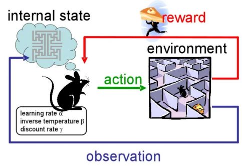
Learning (by operating in and interacting with the world [aka. environment]) to make good sequence of decisions under uncertainty

Rewards/Penalties for sequences of actions

5

Reinforcement Learning (RL)

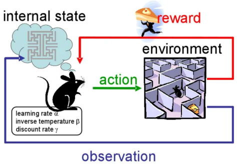
Don't know in advance how the world works



Learning (by operating in and interacting with the world [aka. environment]) to make good sequence of decisions under uncertainty

6

Reinforcement Learning (RL)



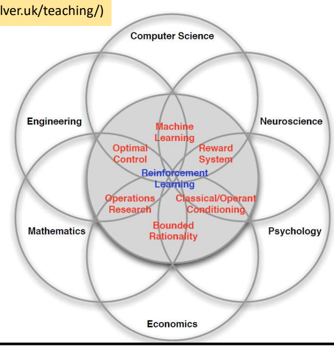
Learning (by operating in and interacting with the world [aka. environment]) to make good sequence of decisions under uncertainty

- The world can be dynamic
- Agent's actions are typically nondeterministic
- There can be other agents whose actions can be unpredictable

7

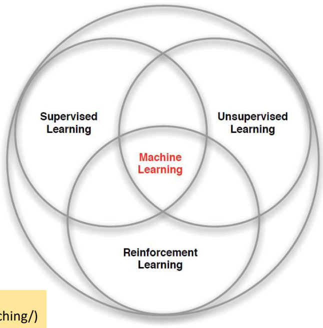
Source: David Silver (https://www.davidsilver.uk/teaching/)

Many faces of Reinforcement Learning



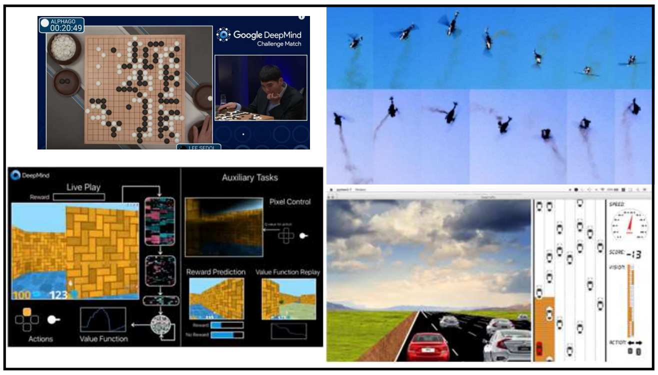
8

Branches of Machine Learning



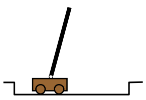
Source: David Silver (https://www.davidsilver.uk/teaching/)

9

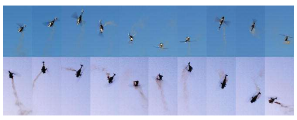


10

Other examples

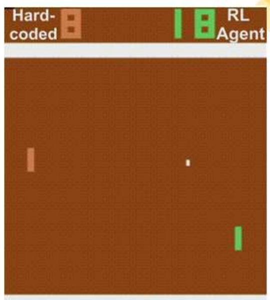


- pole-balancing
- helicopter [Andrew Ng]
- most teachers would not say "good" or "bad"
  - is reward "10" good or bad?
  - rewards could be delayed
- explore the environment and learn from experience
  - not just blind search, try to be smart about it



11

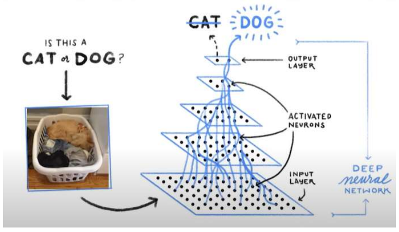
RL Agent playing atari Pong



12

Is RL really necessary?

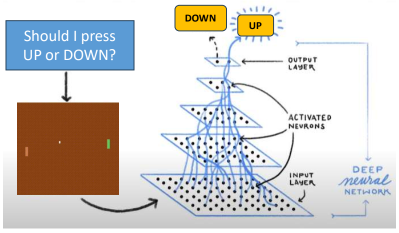
Would supervised learning do the job?



13

Is RL really necessary?

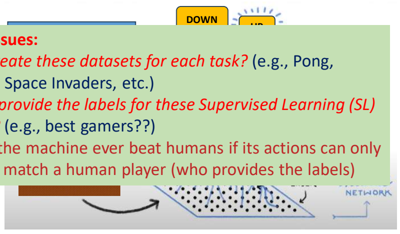
How about we turn it into a supervised learning problem?



14

Is RL really necessary?

How about w… into a superv… learning prob…



Two major issues:

1. *How to create these datasets for each task?* (e.g., Pong, Breakout, Space Invaders, etc.)
2. *Who will provide the labels for these Supervised Learning (SL) datasets?* (e.g., best gamers??)
   2a. Will the machine ever beat humans if its actions can only (at best) match a human player (who provides the labels)

15

Characteristics of Reinforcement Learning

What makes reinforcement learning different from other machine learning paradigms?

- There is no supervisor, only a *reward* signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d data)
- Agent's actions affect the subsequent data it receives

Source: David Silver (https://www.davidsilver.uk/teaching/)

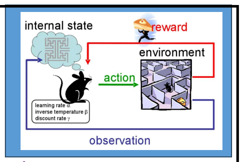
16

Outline

- Examples
- Defining an RL problem
  - Markov Decision Processes
- Solving an RL problem
  - Dynamic Programming
  - Monte Carlo methods
  - Temporal-Difference learning

17

What is Reinforcement Learning



Learning (by operating in and interacting with the world [aka. environment]) to make good sequence of decisions under uncertainty

- Reward/Penalty
  - Reward function
  - Value function
- Sequential decision making
  - Policy
- Environment
  - States
  - Uncertainty
- Learning
  - Observation
  - Algorithms
  - Exploration vs Exploitation

18

What is Reinforcement Learning - Reward

- Reward (at time *t*) R_t is a numerical feedback signal
- Indicates how well agent is doing at step *t*
- The agent's job is to maximise cumulative reward
  - Sequence of decisions → sequence of actions
- Rewards can be delayed
  - Delayed rewards can be "discounted"
  - See next slide on "Sequential decision making"

19

What is Reinforcement Learning – Sequential decision making

- **Goal**: select actions to maximise total future reward
- Actions may have long term (i.e., delayed) consequences
- It may be better to sacrifice immediate reward to gain more long-term reward
  - Study (vs party) now
  - Contribute to your superannuation (less money to spend now vs retire comfortably)
  - Self-driving EV stops now to recharge (vs stranded in the middle of nowhere)

20

Example - Robot in a room

			+100
			-100
START			

actions: UP, DOWN, LEFT, RIGHT

Deterministic vs Stochastic

- **reward** +100 at [4,3], -100 at [4,2] (and they are also terminal states)

21

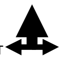
Example - Robot in a room

			+100
			-100
START			

actions: UP, DOWN, LEFT, RIGHT

UP

80% move UP
10% move LEFT
10% move RIGHT



- **reward** +100 at [4,3], -100 at [4,2] (and they are also terminal states)

22

Example - Robot in a room

			+100
			-100
START			

actions: UP, DOWN, LEFT, RIGHT

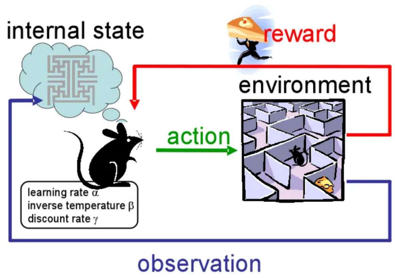
UP

80% move UP
10% move LEFT
10% move RIGHT

- **reward** +100 at [4,3], -100 at [4,2] (and they are also terminal states)
- **reward** -4 for each step

23

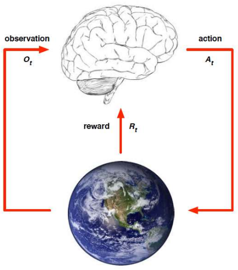
RL agent & Environment



24

Agent and Environment

Source: David Silver (https://www.davidsilver.uk/teaching/)



- At each step t the agent:
  - Executes action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at env. step

25

History and State

Source: David Silver (https://www.davidsilver.uk/teaching/)

- The history is the sequence of observations, actions, rewards

$$H_t = O_1, R_1, A_1; \dots, A_{t-1}; O_t, R_t$$

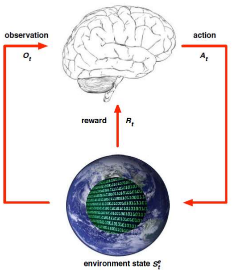
- i.e. all observable variables up to the current time t
- What happens next depends on the history:
  - The agent selects actions
  - The environment selects observations/rewards
- State is the information used to determine what happens next
- i.e., state is a function of the history:

$$S_t = f(H_t)$$

26

Environment State

Source: David Silver (https://www.davidsilver.uk/teaching/)

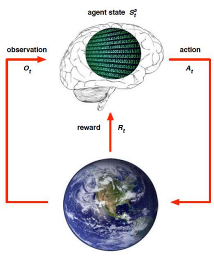


- The environment state S^e_t is the environment's private representation
- The environment state is not usually visible to the agent
- Even if S^e_t is visible, it may contain irrelevant information

27

Agent State

Source: David Silver (https://www.davidsilver.uk/teaching/)



- The agent state S^a_t is the agent's internal representation (of the environment)
- i.e. whatever information the agent uses to pick the next action
- It can be any function of the history:

$$S^a_t = f(H_t)$$

28

Information State

Source: David Silver (https://www.davidsilver.uk/teaching/)

- An information state (a.k.a. Markov state) contains all useful information from the history.

Definition: A state S_t is Markov if and only if

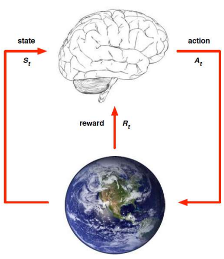
$$P[S_{t+1} \mid S_t] = P[S_{t+1} \mid S_1, \dots, S_t]$$

- i.e., "*The future is independent of the past given the present*"
- Existing RL algorithms assumes that the agent state S^a_t is Markov.

29

Fully Observable Environment

Source: David Silver (https://www.davidsilver.uk/teaching/)



- Full observability: agent directly observes environment state

$$O_t = S^a_t = S^e_t$$

- Agent state = environment state = information state
- Formally, this is a Markov Decision Process (MDP)

30

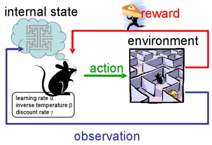
Partially Observable Environment

Source: David Silver (https://www.davidsilver.uk/teaching/)

- Partial observability: agent indirectly observes environment with its sensors:
  - Its sensors can be incomplete
  - Its sensors can be noisy
  - There are other agents who have private information not available to the RL agent (e.g., agent playing card games, warfare, etc.)
- Now agent state ≠ environment state
- Formally this is a partially observable Markov decision process (POMDP)
- Agent must construct its own state representation S^a_t:
  - e.g. Complete (observable) history: $S^a_t = H_t$

31

RL Agents – Making decisions and Learning



- Major Components of an RL Agent:
  - **Policy**: (Mandatory) agent's behaviour function
  - **Value function**: (Optional) how good is each state and/or action
  - **Model**: (Optional) agent's representation of the environment

32

Robot in a room (or, Mouse in a maze)

			+100
			-100
START			

actions: UP, DOWN, LEFT, RIGHT

UP

80%	move UP
10%	move LEFT
10%	move RIGHT

- **reward** +100 at [4,3] (the **cheese**), -100 at [4,2] (the **trap**)
- **reward** -4 for each step
- For an MDP, the states consist of the whole maze (i.e., agent having a bird's-eye view of the maze)
- what's the strategy to achieve max reward?

33

Is this a solution?

→	→	→	+100
↑			-100
↑			

- only if actions are deterministic
  - not in this case (actions are stochastic)
- Solution is a policy
  - **Policy**: mapping from each state to an action

34

Which policy?

→	→	→	+100
↑		↑	-100
↑	→	↑	←

→	→	→	+100
↑		→	-100
→	→	→	↑

→	→	→	+100
↑		↑	-100
↑	←	←	←

- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a \mid s) = P[A_t = a \mid S_t = s]$

Note: Deterministic policy ≠ Deterministic environment

35

RL algorithms - Value function

- Assume for a moment that the environment is DETERMINISTIC, **optimal policy**:

→	→	→	+100
↑		↑	-100
↑	→	↑	←

- We can calculate (recursively from the terminal states) the following **value function**:

88	92	96	+100
84		92	-100
80	84	88	84

36

RL algorithms - Value function

- Value function is a prediction of future reward
- Used to evaluate the goodness/badness of states
- And therefore to select between actions
- Example:

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... \mid S_t = s \right]$$

37

What about STOCHASTIC environments!

Optimal policy: - **WHY?**

→	→	→	+100
↑		↑	-100
↑	←	←	←

UP

80% move UP
10% move LEFT
10% move RIGHT

Can you calculate the value function?

HINT: It is the *expected value* based on the probabilistic outcomes of the action taken in each cell!

38

Reward for each step: -120

→	→	→	+100
↑		→	-100
→	→	→	↑

39

Reward for each step: -10

→	→	→	+100
↑		↑	-100
↑	→	↑	←

40

Reward for each step: -4

→	→	→	+100
↑		↑	-100
↑	←	←	←

41

Reward for each step: -1

→	→	→	+100
↑		←	-100
↑	←	←	↓

42

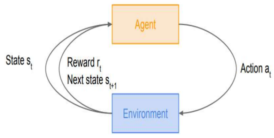
Reward for each step: +1

↓	←	←	+100
↓		←	-100
←	←	←	↓

43

Markov Decision Process (MDP) - Model

- set of states S, set of actions A, initial state s_0
- transition model P(s'|s,a)
  - P([1,2] | [1,1], up) = 0.8
- reward function r(s)
  - r([4,3]) = +100



44

Categorizing RL agents (1)

Source: David Silver (https://www.davidsilver.uk/teaching/)

- Value Based
  - No Policy (Implicit)
  - Value Function
- Policy Based
  - Policy
  - No Value Function
- Actor Critic
  - Policy
  - Value Function

45

Categorizing RL agents (2)

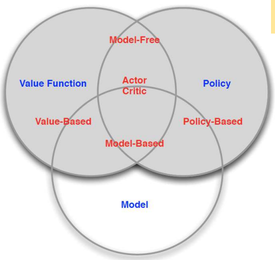
Source: David Silver (https://www.davidsilver.uk/teaching/)

- Model Free
  - Policy and/or Value Function
  - No Model
- Model Based
  - Policy and/or Value Function
  - Model

46

RL Agent Taxonomy

Source: David Silver (https://www.davidsilver.uk/teaching/)



47

Computing return from rewards

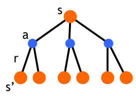
- episodic (vs. continuing) tasks
  - "game over" after N steps
  - optimal policy depends on N; harder to analyze
- additive rewards
  - $V(s_0, s_1, \ldots) = r(s_0) + r(s_1) + r(s_2) + \ldots$
  - infinite value for continuing tasks
- discounted rewards (γ < 1)
  - $V(s_0, s_1, \ldots) = r(s_0) + \gamma^* r(s_1) + \gamma^{2*} r(s_2) + \ldots$
  - value bounded if rewards bounded

48

Value functions

- state value function: $V^\pi(s)$
  - expected return when starting in *s* and following π
- state-action value function: $Q^\pi(s,a)$
  - expected return when starting in *s*, performing *a*, and following π
- For a fixed policy π, the utility function obey the **Bellman equation**:

$V^\pi(s) = r(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) * V^\pi(s')$



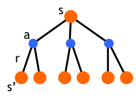
SWIN BUR NE SWINBURNE UNIVERSITY OF TECHNOLOGY

49

Value functions

- state value function: $V^\pi(s)$
  - expected return when starting in *s* and following π
- state-action value function: $Q^\pi(s,a)$
  - expected return when starting in *s*, performing *a*, and following π
- For a fixed policy π, the utility function obey the **Bellman equation**:

$V^\pi(s) = r(s) + \gamma \boxed{\sum_{s'} P(s' \mid s, \pi(s)) * V^\pi(s')}$ $Q^\pi(s,a)$



SWIN BUR NE SWINBURNE UNIVERSITY OF TECHNOLOGY

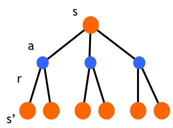
50

Optimal value functions

- there's a set of *optimal* policies
  - V^π defines the value of policy for every state $s \in S$
  - Then, there is a single optimal value for each state $s \in S$

$$V^*(s) = \max_\pi V^\pi(s)$$

- Bellman optimality equation
  - $V^*(s) = r(s) + \gamma * \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) * V^*(s')$
  - system of n non-linear equations
  - solve for $V^*(s)$
  - easy to extract the optimal policy π^*
- having $Q^*(s,a)$ makes it even simpler: $\pi^*(s) = \arg\max_a Q^*(s,a)$



SWIN BUR NE SWINBURNE UNIVERSITY OF TECHNOLOGY

51

Outline

- Examples
- Defining an RL problem
  - Markov Decision Processes
- Solving an RL problem
  - Dynamic Programming
  - Monte Carlo methods
  - Temporal-Difference learning

SWIN BUR NE SWINBURNE UNIVERSITY OF TECHNOLOGY

52

Dynamic programming (DP)

- main idea
  - If an action in a situation/state cause something (really) good/bad to happen immediately then we **"know"** the **"goodness"** of that action (given that state) and the **"value"** of that state!
  - need a **perfect** model of the environment (aka. **Model-based RL**)
- two main components
  (of "Policy iteration algorithm")
  - policy evaluation: compute V^π from π
  - policy improvement: improve π based on V^π
  - start with an arbitrary policy
  - repeat evaluation/improvement until convergence

			+100
			-100
START			

SWIN BUR NE SWINBURNE UNIVERSITY OF TECHNOLOGY

53

Policy iteration

- Initialise V and policy π' for all $s \in S$

$\pi = \pi'$

For all $s \in S$: (concurrent update ☺)

$V(s) = r(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) * V(s')$

For all $s \in S$:

$\pi'(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) * V(s')$

Repeat until $\pi = \pi'$

SWIN BUR NE SWINBURNE UNIVERSITY OF TECHNOLOGY

54

Policy iteration

- Initialise V and policy π' for all s∈S

$\pi = \pi'$

For all s∈S: (concurrent update ☺)

V(s) = r(s) + γ$\sum_{s'}$ P(s' | s, π(s))*V(s')

Question: If the state-action value function Q(s,a) is maintained, how does the algorithm look?

For all

π'(s) = arg max$_{a \in A(s)}$ $\sum_{s'}$ P(s' | s, a)*V(s')

Repeat until $\pi = \pi'$

55

Using DP

- The algorithm only works for an MDP, where:
  - The set of all states S is known
  - A transition model P(s' | s, a) is known for all s', s ∈S and all a ∈A
  - A reward function r(s) is known for all states s ∈S
- Reinforcement learning does not always have those luxuries!
  - transitions and rewards usually not available
    - Learning to walk, fly an aircraft, playing games with incomplete info such as poker
  - how to explore the environment to learn about states you may not known prior
    - Mars rover
  - how to update the policy as transition model and the reward function are being updated based on the leaner's experience

56

Outline

- examples
- defining an RL problem
  - Markov Decision Processes
- solving an RL problem
  - Dynamic Programming
  - Monte Carlo methods
  - Temporal-Difference learning

57

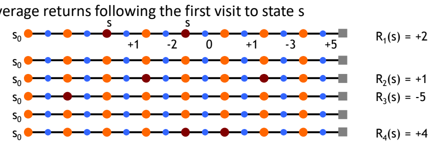
Monte Carlo methods

- don't need full knowledge of environment (aka. Model free RL)
  - just experience, or
  - simulated experience
- but similar to DP
  - policy evaluation, policy improvement
- **averaging sample returns**
  - defined only for **episodic tasks**

58

Monte Carlo policy evaluation

- want to estimate V^{π}(s)
  - = expected return starting from s and following π
  - estimate as average of observed returns in state s
- first-visit MC
  - average returns following the first visit to state s

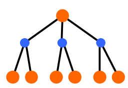


V^{π}(s) ≈ (2 + 1 - 5 + 4)/4 = 0.5

59

Monte Carlo control

- V^{π} not enough for policy improvement
  - need exact model of environment
- estimate Q^{π}(s,a)



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

- MC control
  - update after each episode $\pi_0 \rightarrow^E Q^{\pi_0} \rightarrow^I \pi_1 \rightarrow^E Q^{\pi_1} \rightarrow^I \ldots \rightarrow^I \pi^* \rightarrow^E Q^*$
- non-stationary environment

$$V(s_t) \leftarrow V(s_t) + \alpha\,[R_t - V(s_t)]$$

- a problem
  - greedy policy won't explore all actions

60

Maintaining exploration

- deterministic/greedy policy won't explore all actions
  - don't know anything about the environment at the beginning
  - need to try all actions to find the optimal one
- maintain exploration
  - use *soft* policies instead: π(s,a)>0 (for all s,a)
    - i.e., the policy does not simply return an action for each state, it returns a probability distribution over the action space for each state
- ε-greedy policy
  - with probability 1-ε perform the optimal/greedy action
  - with probability ε perform a random action
  - will keep exploring the environment
  - slowly move it towards greedy policy: ε -> 0

61

Summary of Monte Carlo

- don't need model of environment
  - averaging of sample returns
  - only for episodic tasks
- learn from sample episodes or simulated experience
- can concentrate on "important" states
  - don't need a full sweep
- need to maintain exploration
  - use soft policies

62

Outline

- examples
- defining an RL problem
  - Markov Decision Processes
- solving an RL problem
  - Dynamic Programming
  - Monte Carlo methods
  - Temporal-Difference learning

63

Temporal Difference Learning

- combines ideas from MC and DP
  - like MC: learn directly from experience (don't need a model)
  - like DP: learn from values of successors
  - works for continuous tasks, usually faster than MC
- constant-alpha MC:
  - have to wait until the end of episode to update

$$V(s_t) \leftarrow V(s_t) + \alpha \left[R_t - V(s_t)\right]$$

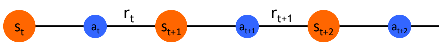
- simplest TD
  - update after every step, based on the successor

$$V(s_t) \leftarrow V(s_t) + \alpha \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right]$$

64

Sarsa

- again, need Q(s,a), not just V(s)



$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$$

- control
  - start with a random policy
  - update Q and π after each step
  - again, need ε-soft policies

65

The RL Intro book

Richard Sutton, Andrew Barto
Reinforcement Learning,
An Introduction

http://www.cs.ualberta.ca/~sutton/book/the-book.html

66

Summary

- Reinforcement learning
  - use when need to make decisions in uncertain environment
- solution methods
  - dynamic programming
    - need complete model
  - Monte Carlo
  - time-difference learning (Sarsa, Q-learning)
- most work
  - algorithms simple
  - need to design features, state representation, rewards

67

Where to start?

- You need an environment (to build your RL agent)
- The OpenAI Gym provides a good starting point with some pre-defined environments and also allow you to define your own environment:
- https://towardsdatascience.com/reinforcement-learning-with-openai-d445c2c687d2

68