Reinforcement Learning I

Types of machine learning

	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Goal	**Predict** …from examples	**Describe** ...structure in data	**Strategize** learn by trial and error
Data	(x, y)	x	delayed feedback
Types	• Classification • Regression	• Density estimation • Clustering • Dimensionality reduction • Anomaly detection	• Model-free learning • Model-based learning

Resources

This reinforcement learning series draws heavily on these resources

Sutton and Barto, 1998
(2nd edition 2018)

Reinforcement Learning: An Introduction

Draft of updated edition available free online:
http://www.incompleteideas.net/book/the-book-2nd.html

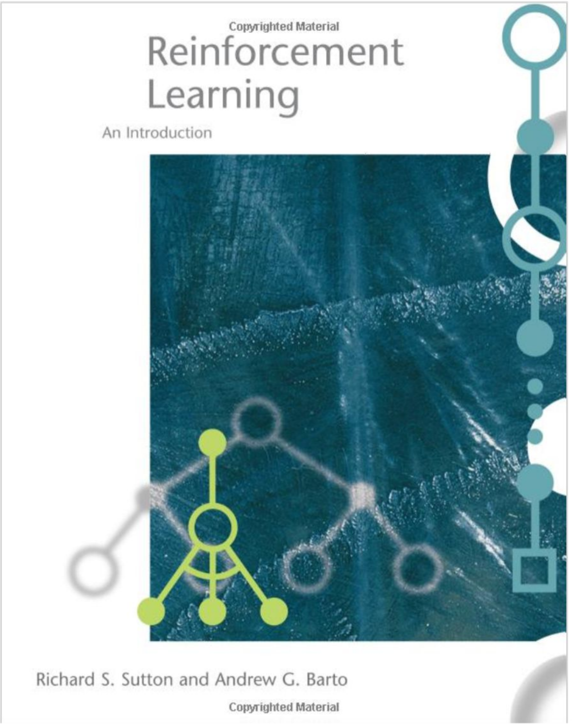


Image from Amazon.com (where the book may be purchased)

David Silver, 2015

University College London
Advanced Topics 2015 (COMPM050/COMPGI13)

Course website:
http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html

Video series:
https://www.youtube.com/watch?v=2pWv7GOvuf0&list=PL7-jPKtc4r78-wCZcQn5IqyuWhBZ8fOxT

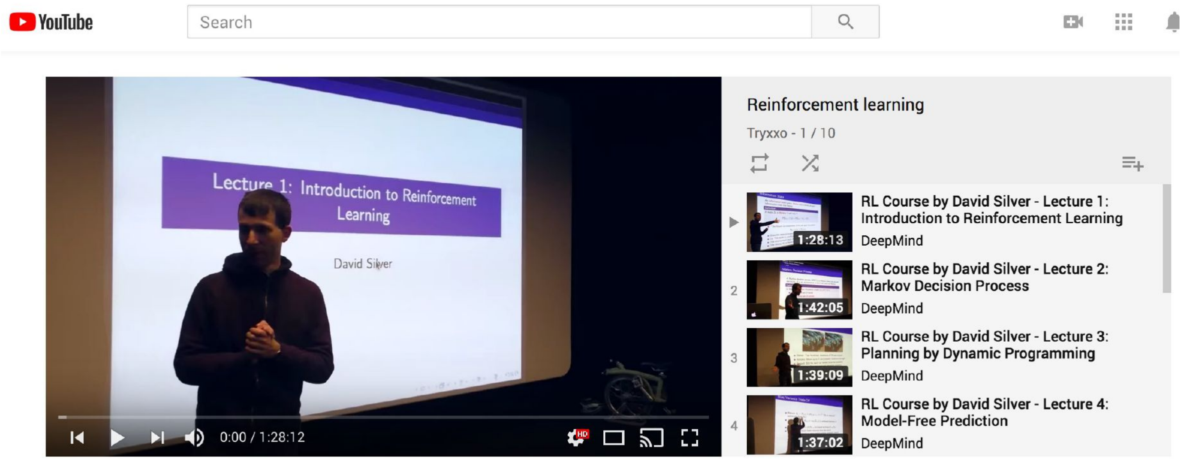


Image from Youtube.com

Reinforcement learning

Control Theory
(optimal control)

Psychology and Neuroscience

Reinforcement Learning

Machine Learning / Artificial Intelligence

Operations Research

Reinforcement Learning

Goal: select actions to maximize total long-term rewards

Sequential decision making

Challenge: an action needs to be taken at each step

Evaluation of rewards versus instruction (examples of correct actions)

Challenge: this leads to a trial-and-error approach to learning

May be better to sacrifice immediate reward for long-term gains

Challenge: exploration (of untried actions) vs exploitation (of current knowledge)

Rewards may be delayed

Challenge: credit assignment: which action(s) led to the reward(s)?

David Silver, 2015

Reinforcement Learning Applications

- Self-driving cars (link)
- Energy-efficient data center cooling control (link)
- Financial trading (link)
- Medical diagnosis and treatment (link)
- Gaming (AlphaGo, Atari, StarCraft)

Industry Leaders: Google Deepmind (link)

Reinforcement Learning Examples

Winning at Atari: https://youtu.be/V1eYniJ0Rnk

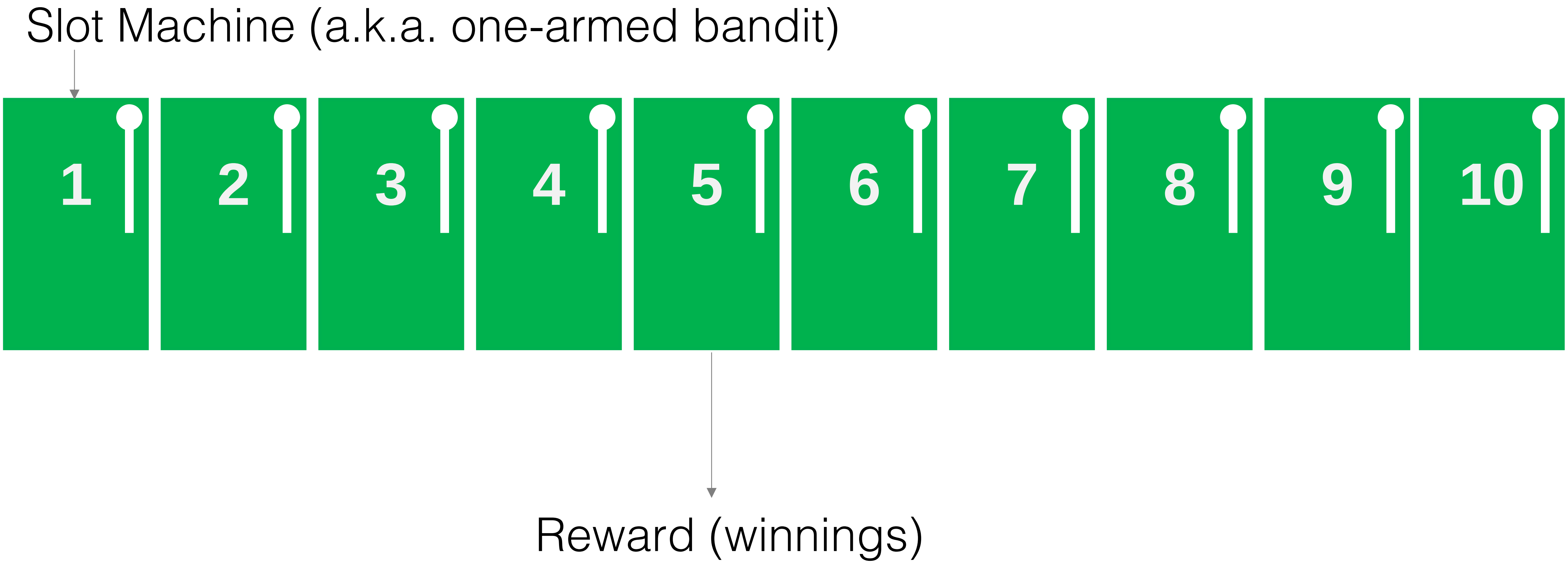
Balancing an inverted pendulum: https://youtu.be/b1c0N_Fs9wc

Flipping pancakes: https://youtu.be/W_gxLKSsSIE

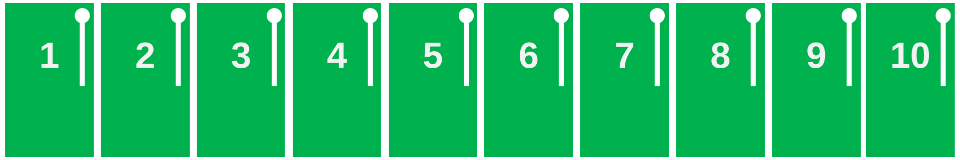
RL is a unifying framework for a wide range of problems

Multi-armed Bandit

You walk into a casino…



Multi-armed bandit problem



Trial/episode: play one machine

Action: pick one machine to play (one action per trial/episode)

Reward: how much you win or lose

- Each machine has an unknown probability of payoff/reward
- The rewards are stochastic (their distributions are unknown)

Action-Value: expected reward for taking each action

State: only 1 “state” in this problem - our environment doesn’t change

create a policy

Policy: How do we choose actions to maximize our total rewards?

- If we knew the best machine, we’d always pick it
- This is what we want to learn

Multi-armed bandit

The *true* **action-value** of an action is $q_*(a)$

Our *estimated* **action-value** at the t^{th} play is $q_t(a)$

If action a has been chosen k_a times prior to t:

$$q_t(a) = \frac{r_1 + r_2 + \cdots + r_{k_a}}{k_a}$$

As we take action a more, our action-value estimates improve

Sutton and Barto, 1998

Multi-armed bandit policies, $\pi(s)$

Greedy action:

Select a* $= \arg\max_{a} q_t(a)$

Problem: if the initial rewards are not representative, this will be suboptimal

ϵ-Greedy methods:

Select a* with probability $1 - \epsilon$, otherwise, randomly select another option

Problem: in the long run, this will waste reward once the best action is known

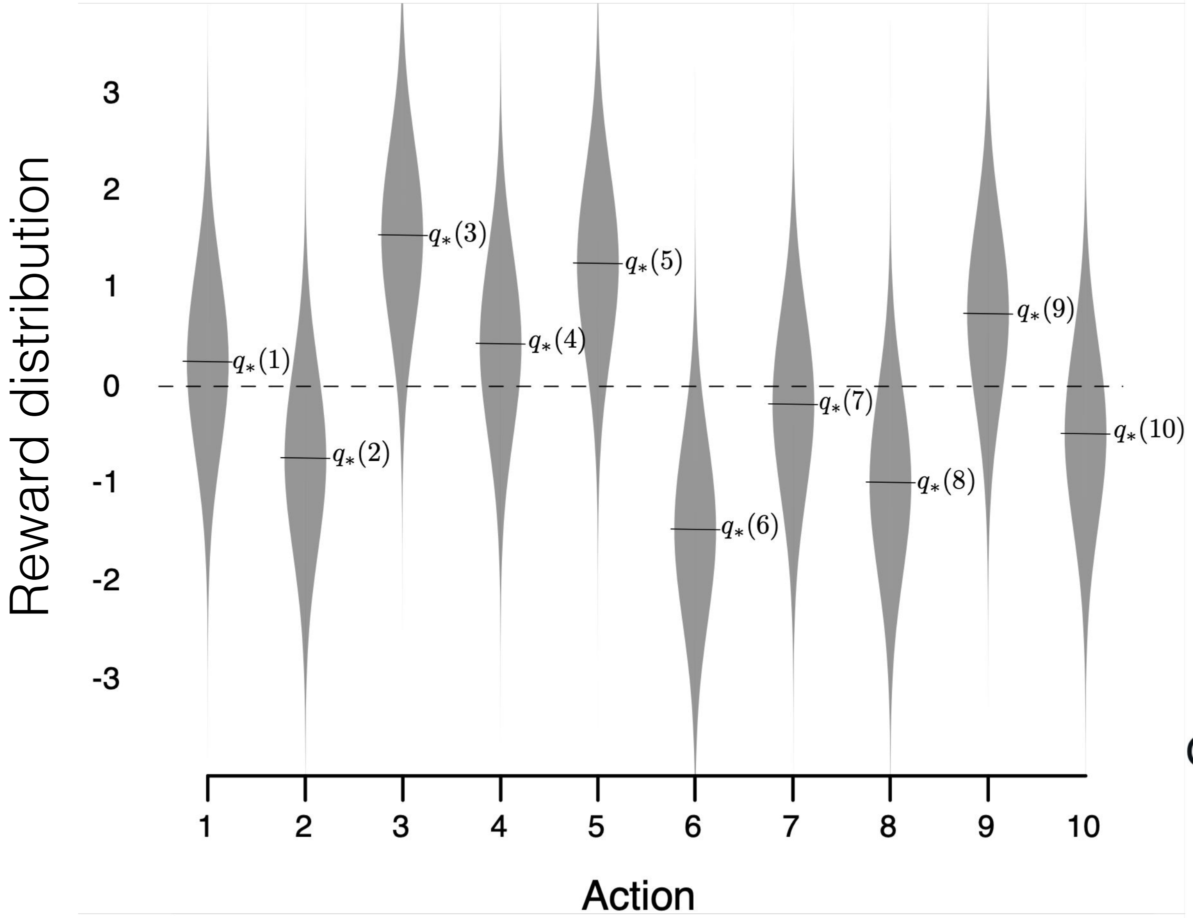
Solution: reduce ϵ over time

Alternative:

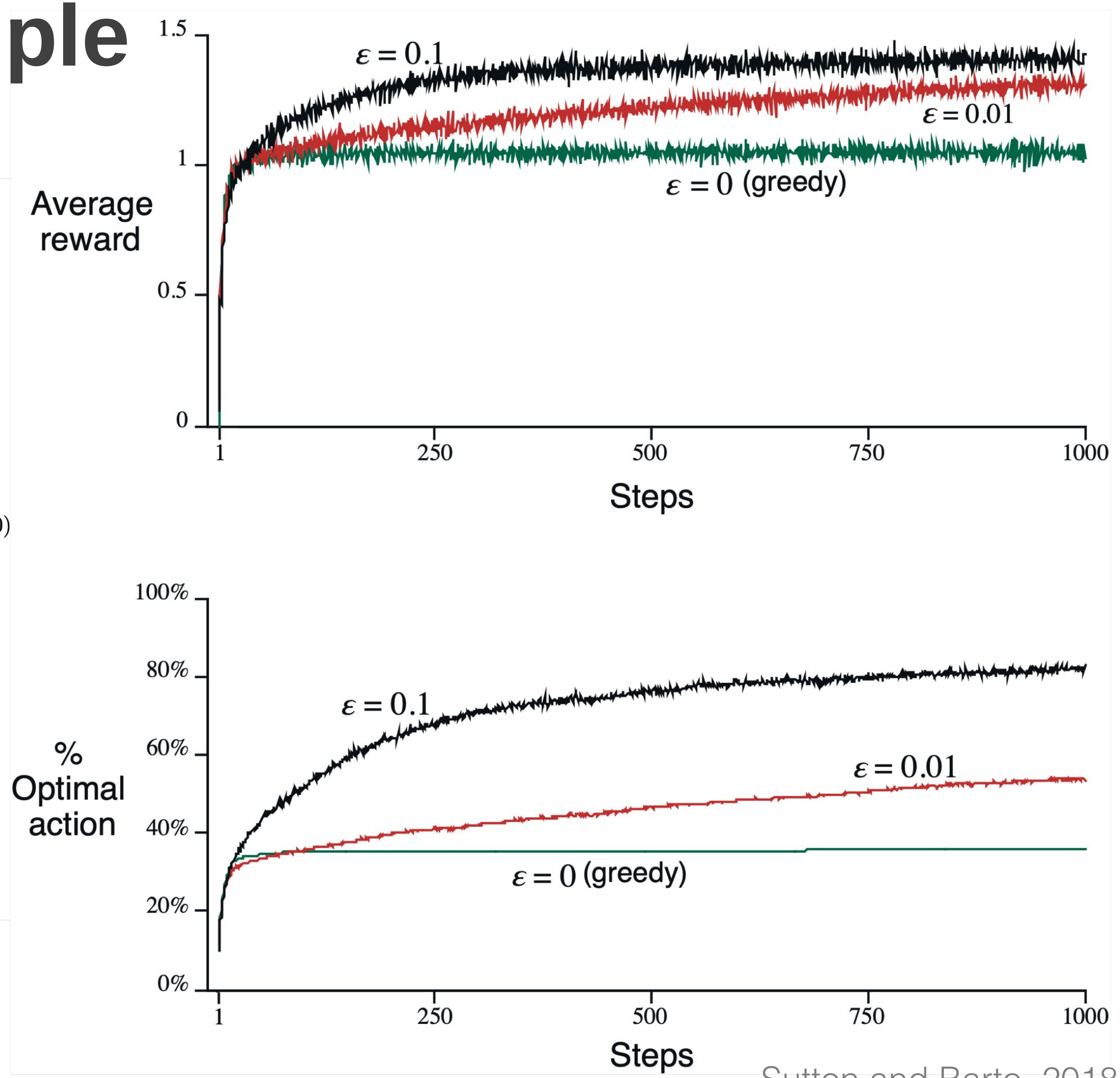
Select the action probabilities based on the expected value

Probability of selecting action $P(a) = \frac{\exp(q_t(a))}{\sum_{b=1}^{n} \exp(q_t(b))}$

10-Armed Bandit Example



Note: Each distribution has a mean $q_*(a)$ with unit variance



Sutton and Barto, 2018

Roadmap for this module

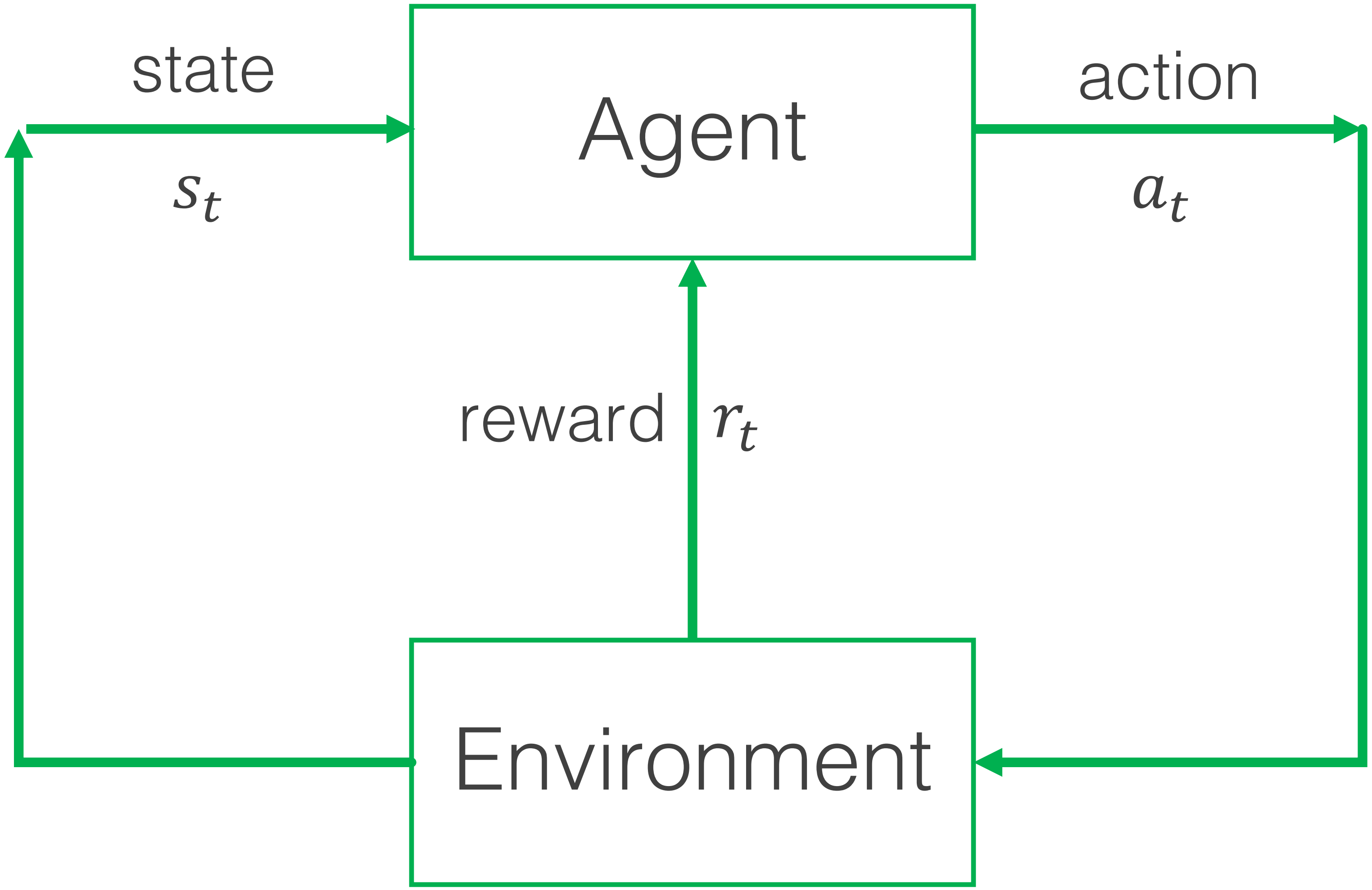
The multi-armed bandit only has 1 state, but the full RL problem learns policies when there are many states that the agent moves between

State representations and Markov decision processes (MDPs)
(with a discussion of Markov processes)

Mathematically formulating the RL problem with MDPs

Methods for solving RL problems in practice
(dynamic programming and Monte Carlo control)

Agent-environment Interaction



Agent at each step t…

Encounters state, s_t
Executes action a_t
Receives scalar reward, r_{t+1}

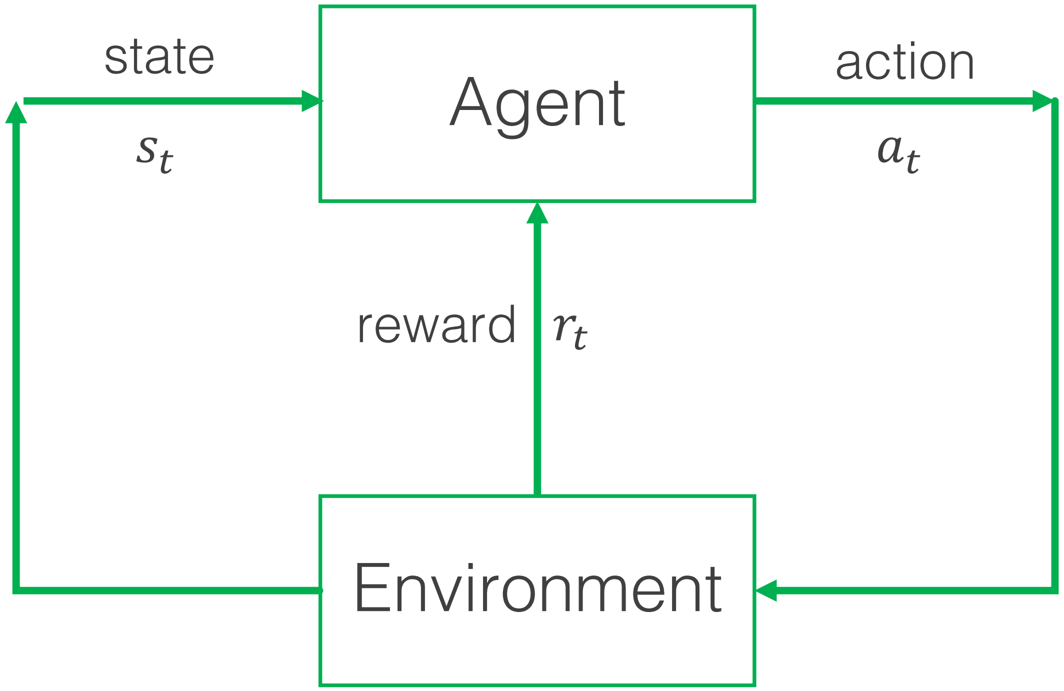
Environment at each step t…

Receives action a_t
Transitions to state, s_{t+1}
Emits scalar reward, r_{t+1}

Actions: choices made by the agent
States: basis on which choices are made
Rewards: define the agent's goals

David Silver, 2015

Reinforcement Learning Components

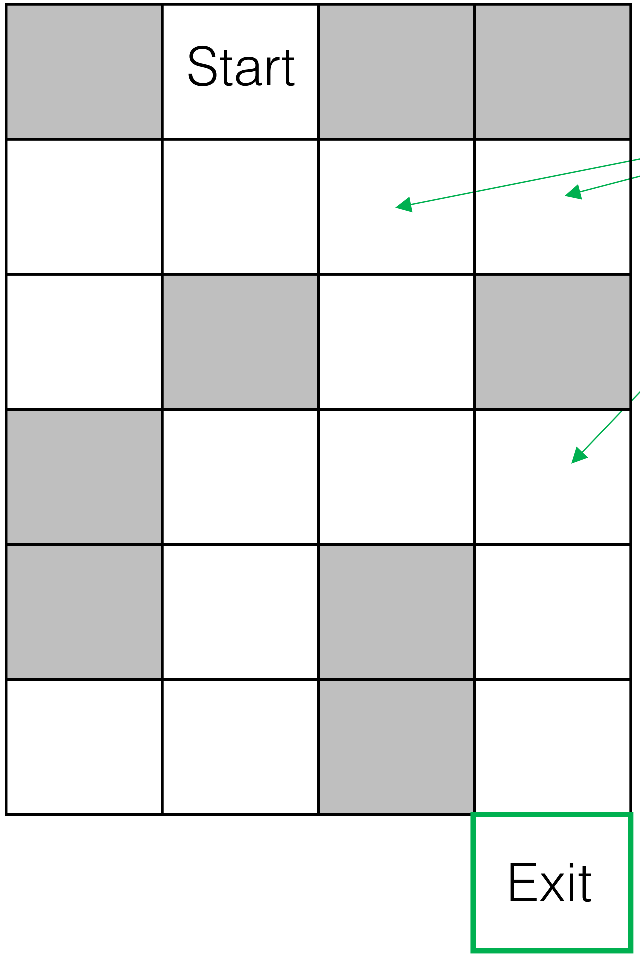


Policy (agent behavior), $\pi(s)$

Reward function (the goal), r_t

Value functions (expected returns),
$v(s)$ State value
$q(s,a)$ Action value

Maze Example: Policy, Value, and Reward



Each location in the maze represents a **state**

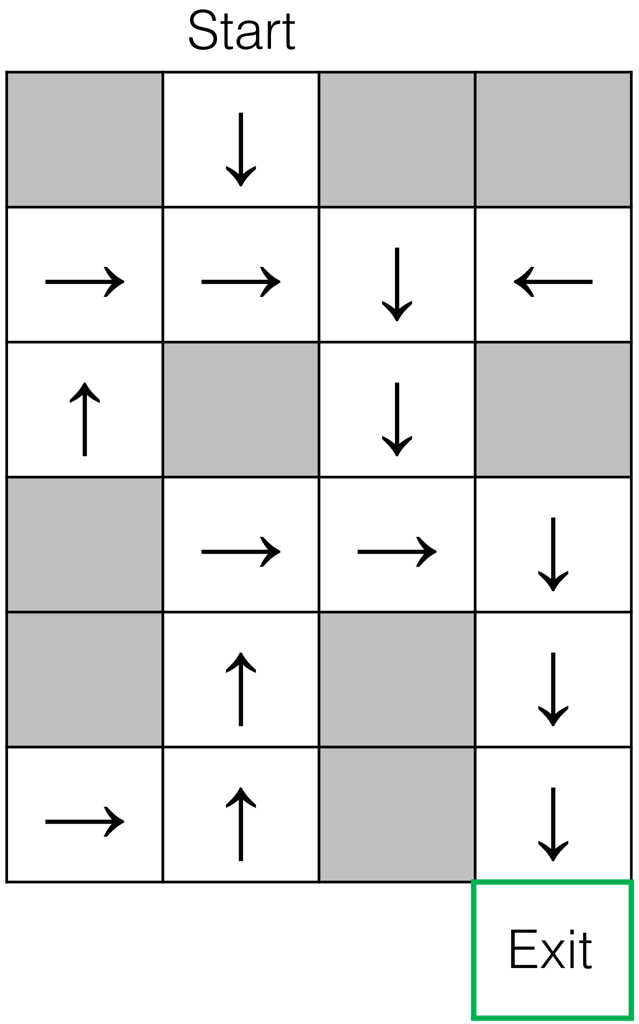
The **reward** is -1 for each step the agent is in the maze

Available **actions**: move ↑,↓,←,→ (as long as that path is not blocked)

Adapted from David Silver, 2015

Policy $\pi(s)$

(which actions to take in each state)



Adapted from David Silver, 2015

Policy $\pi(s)$

(which actions to take in each state)

Start

	↓		
→	→	↓	←
↑		↓	
	→	→	↓
	↑		↓
→	↑		↓

Exit

Reward r_t

(rewards are received after actions are taken)

Start

	-1		
-1	-1	-1	-1
-1		-1	
	-1	-1	-1
	-1		-1
-1	-1		-1

Exit

Adapted from David Silver, 2015

Policy $\pi(s)$

(which actions to take in each state)

Start

	↓		
→	→	↓	←
↑		↓	
	→	→	↓
	↑		↓
→	↑		↓
			Exit

Reward r_t

(rewards are received after actions are taken)

Start

	-1		
-1	-1	-1	-1
-1		-1	
	-1	-1	-1
	-1		-1
-1	-1		-1
			Exit

State Value $v_\pi(s)$

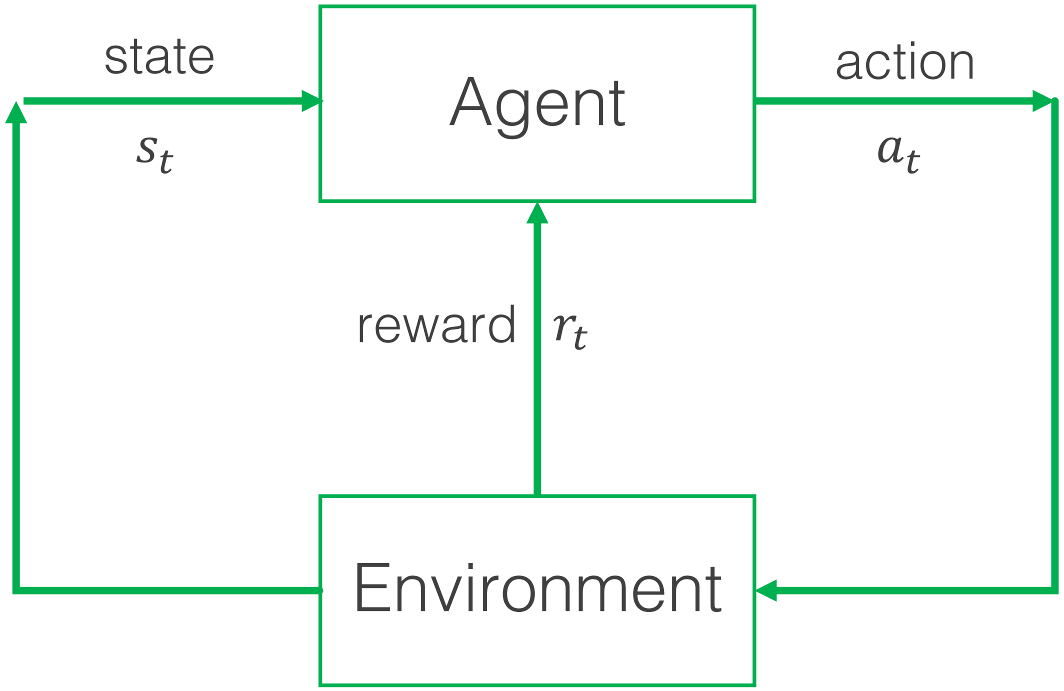
(expected cumulative rewards starting from current state **if** we follow the policy)

Start

	-8		
-8	-7	-6	-7
-9		-5	
	-5	-4	-3
	-6		-2
-8	-7		-1
			Exit

Adapted from David Silver, 2015

Policy



Policy, $\pi(s)$

- Selects an action to choose based on the state
- Determines an agent's "behavior"

Deterministic policy:

$$a = \pi(s)$$

Stochastic policy:

$$\pi(a|s) = P(a_t = a|s_t = s)$$

Helps us "explore" the state space

RL tries to learn the "best" policy

Goals and rewards

Rewards are the **only way** of communicating RL goals

Ex 1: Robot learning a maze

- 0 until it escapes, then +1 when it does
- -1 until it escapes (encourages it to escape quickly)

Ex 2: Robot collecting empty soda cans

- +1 for each empty soda can
- Negative rewards for bumping into things

Chess: what if we set +1 for capturing a piece?
(it may not win the game and still maximize rewards)

What you want achieved not **how**

Returns / cumulative reward

Episodic tasks (finite number, T, of steps, then reset)

$$G_t = r_{t+1} + r_{t+2} + \cdots + r_T$$

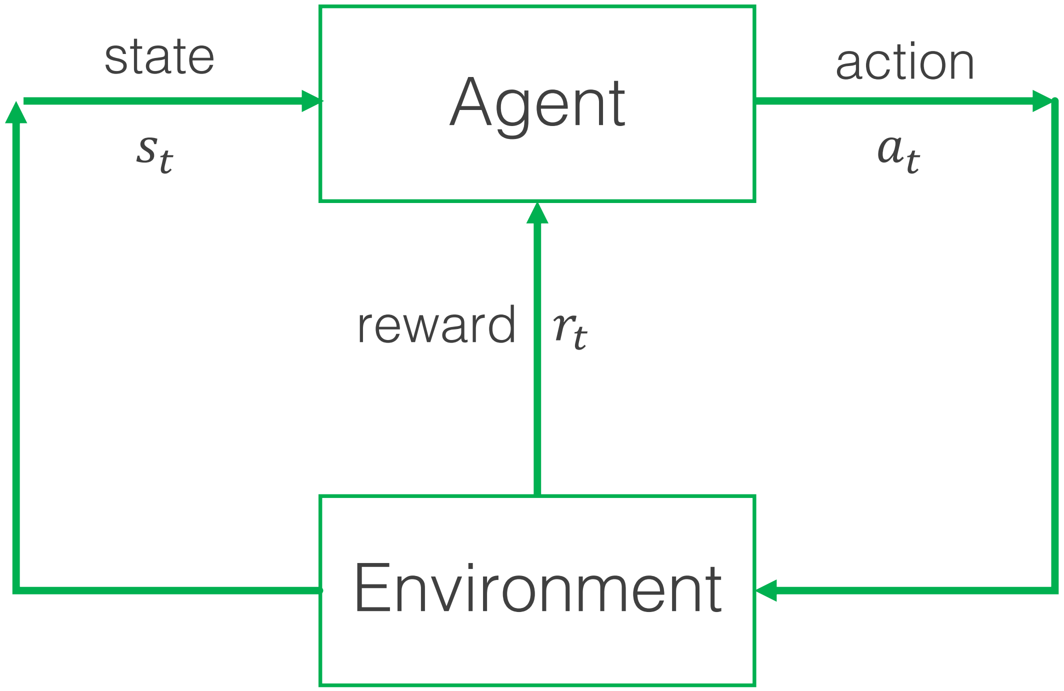
Continuing tasks with discounting ($T \to \infty$)

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

where $0 \leq \gamma \leq 1$ is the discount rate

This makes the agent care more about immediate rewards

Value functions



State Value function, $v_\pi(s)$

- How "good" is it to be in a state, s_t then follow policy π to choose actions
- Total expected rewards

$$v_\pi(s) = E_\pi[G_t | s_t = s]$$

Action Value function, $q_\pi(s,a)$

- How "good" is it to be in a state, s, take action a, then follow policy π to choose actions
- Total expected rewards

$$q_\pi(s,a) = E_\pi[G_t | s_t = s, a_t = a]$$

Where $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$

Policy $\pi(s)$
(which actions to take in each state)

Start

	↓		
→	→	↓	←
↑		↓	
	→	→	↓
	↑		↓
→	↑		↓

Exit

Reward r_t
(rewards are received after actions are taken)

Start

	-1		
-1	-1	-1	-1
-1		-1	
	-1	-1	-1
	-1		-1
-1	-1		-1

Exit

State Value $v_\pi(s)$
(expected cumulative rewards starting from current state **if** we follow the policy)

Start

	-8		
-8	-7	-6	-7
-9		-5	
	-5	-4	-3
	-6		-2
-8	-7		-1

Exit

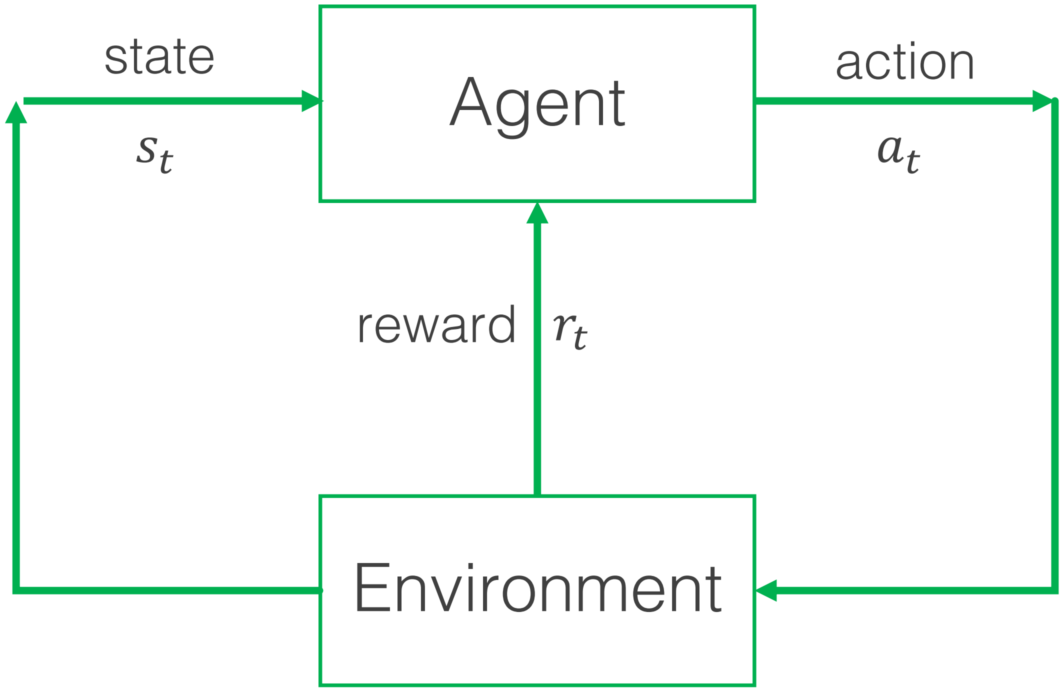
Action Value $q_\pi(s, a)$
(expected cumulative rewards starting from current state **if** we take action a then follow the policy)

↑ -9
→ -7
← -9

↑ -4
↓ -2

Adapted from David Silver, 2015

Model



Model (of the environment)

Transitions: predicts what state the environment will transition to next

$$P_{ss'}^{a} = P(s_{t+1} = s' | s_t = s, a_t = a)$$

Rewards: predicts the next reward given an action

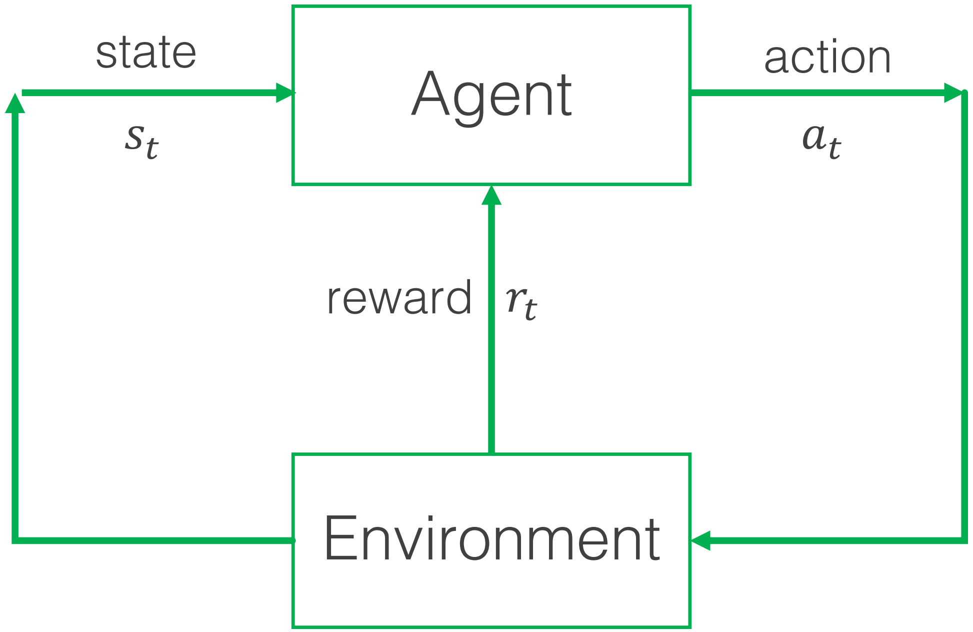
$$R_s^a = E[r_{t+1} | s_t = s, a_t = a]$$

"Planning" is the process of using a model to create or improve a policy

We don't always have a full model of the environment

Model-based RL uses a model
Model-free RL does not use a model

Reinforcement Learning Components



Policy (determines agent behavior), $\pi(s)$

- Determines action given current state
- Agent's way of behaving at a given time

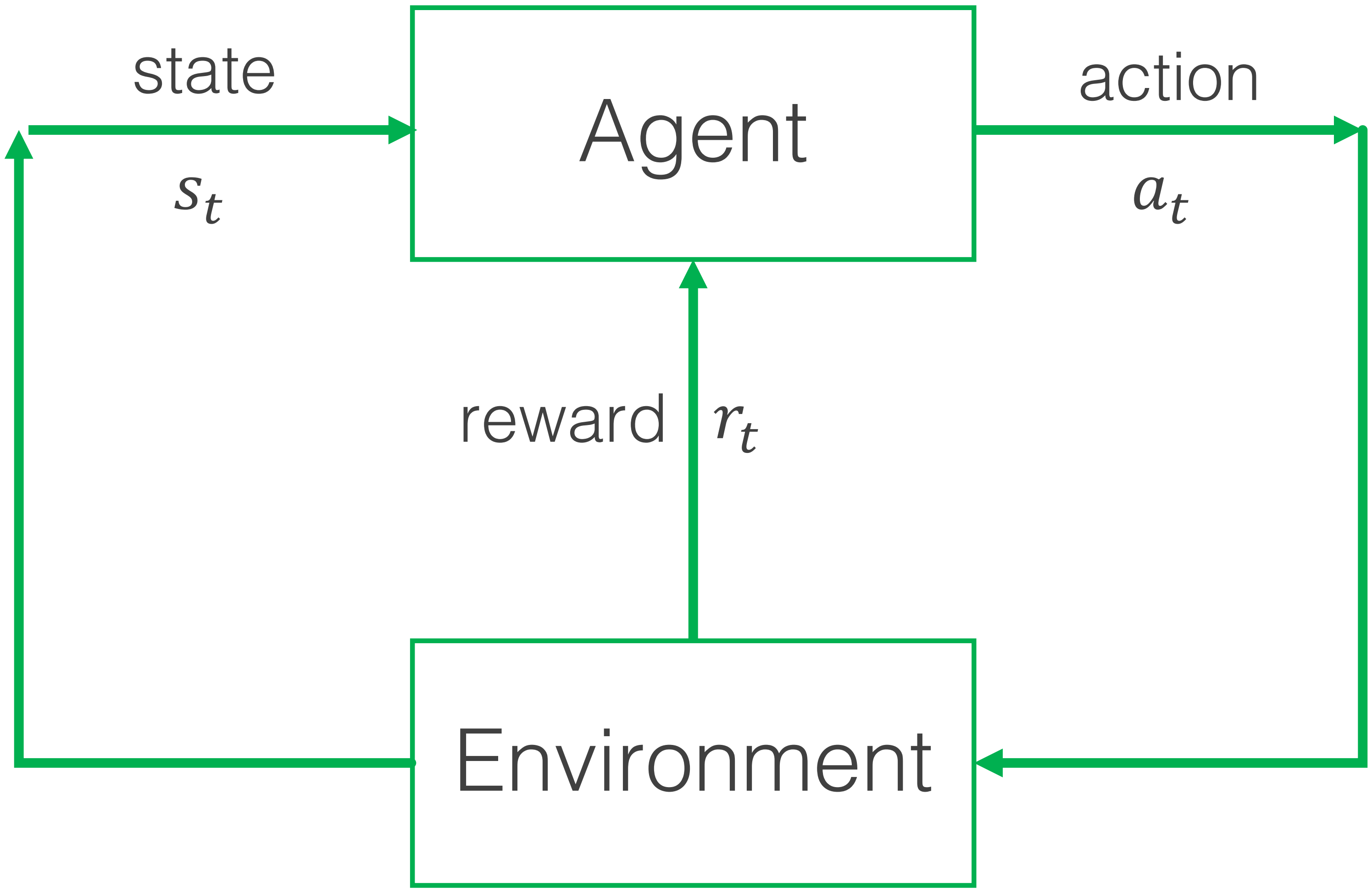
Reward function (sets the goal), r_t

- Maps state of the environment to a reward that describes the state desirability
- Objective is to **maximize total rewards**

Value (estimates expected returns), $v(s,a), q(s)$

- Expected returns from a state and following a specific policy
- How "good" is each state

Environment



Markov Decision Process
(assumed form for most RL problems)

Goal

Maximize returns (expected rewards)

Find the best policy to guide our actions in an environment

Here, environment is modeled as a Markov Decision Process