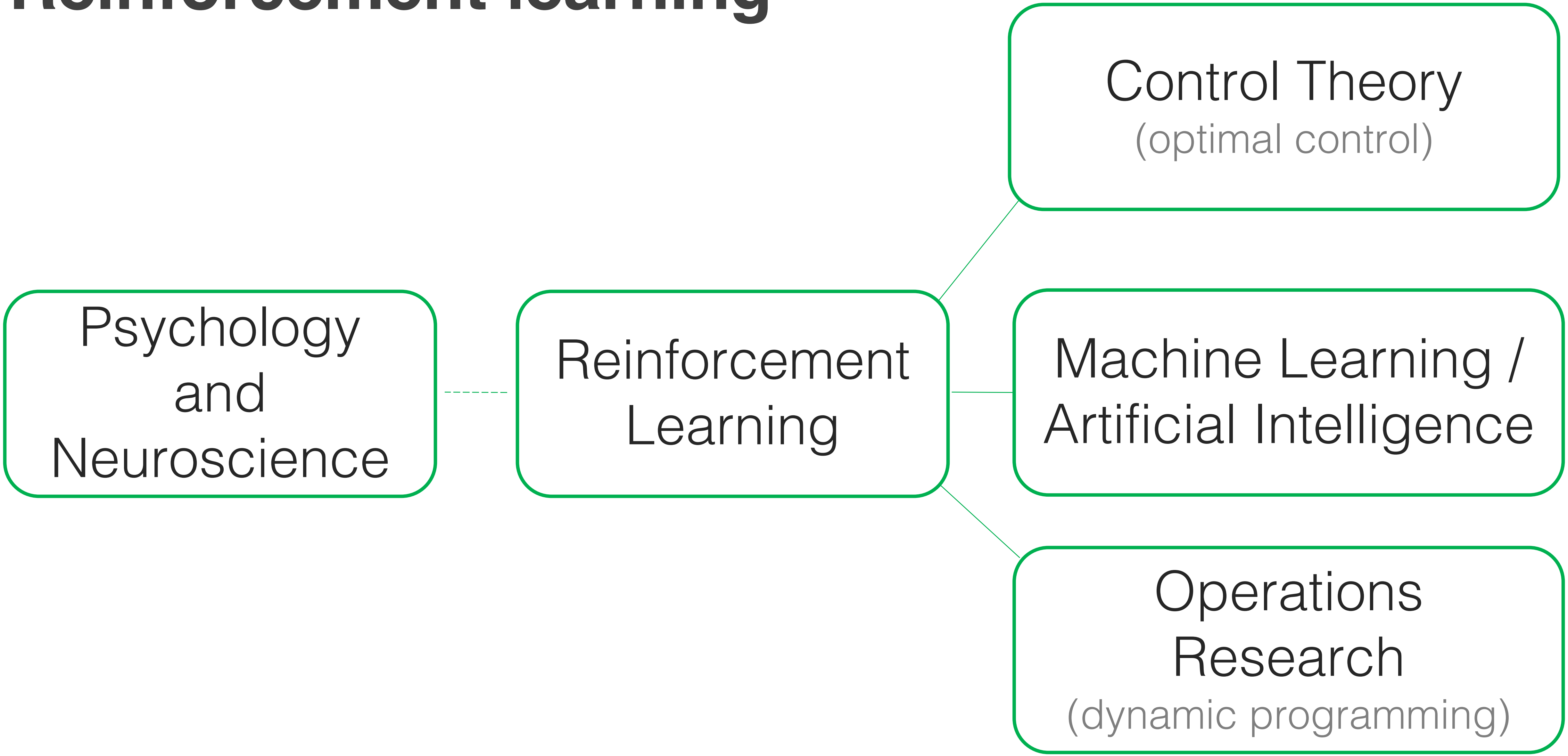
Reinforcement Learning I

Lecture 20

Types of machine learning

	Unsupervised Learning	Supervised Learning	Reinforcement Learning
Goal	**Describe** ...structure in data	**Predict** …from examples	**Strategize** learn through interaction
Data available	predictors, x	predictor and response pairs, (x, y)	actions and delayed responses (called rewards)
Examples	• Density estimation • Clustering • Dimensionality reduction • Anomaly detection	• Classification • Regression	• Model-free learning • Model-based learning

Reinforcement learning



Resources

This reinforcement learning series draws heavily on these resources

Sutton and Barto, 1998

Reinforcement Learning: An Introduction

Draft of 2018 edition available free online:
http://www.incompleteideas.net/book/the-book-2nd.html

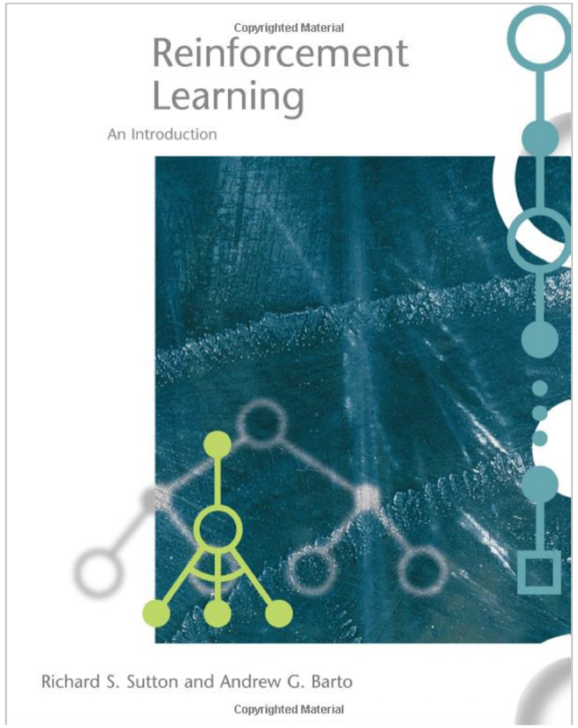


Image from Amazon.com (where the book may be purchased)

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

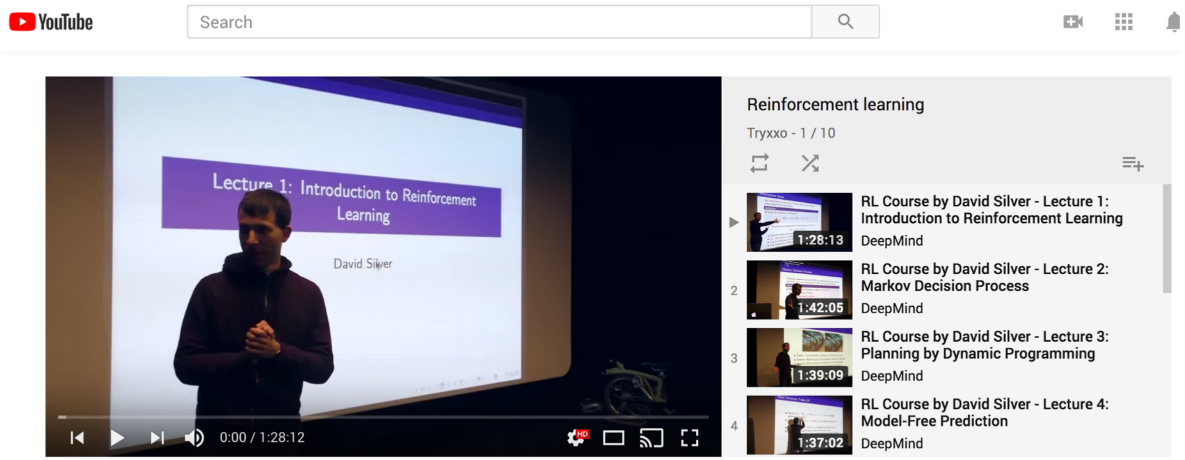


Image from Youtube.com

Reinforcement Learning

Goal: select actions to maximize total long-term rewards

Sequential decision making

An action needs to be taken at each step

Evaluation (rewards) versus instruction (examples of correct actions)

This leads to a trial-and-error approach to learning

Rewards may be delayed

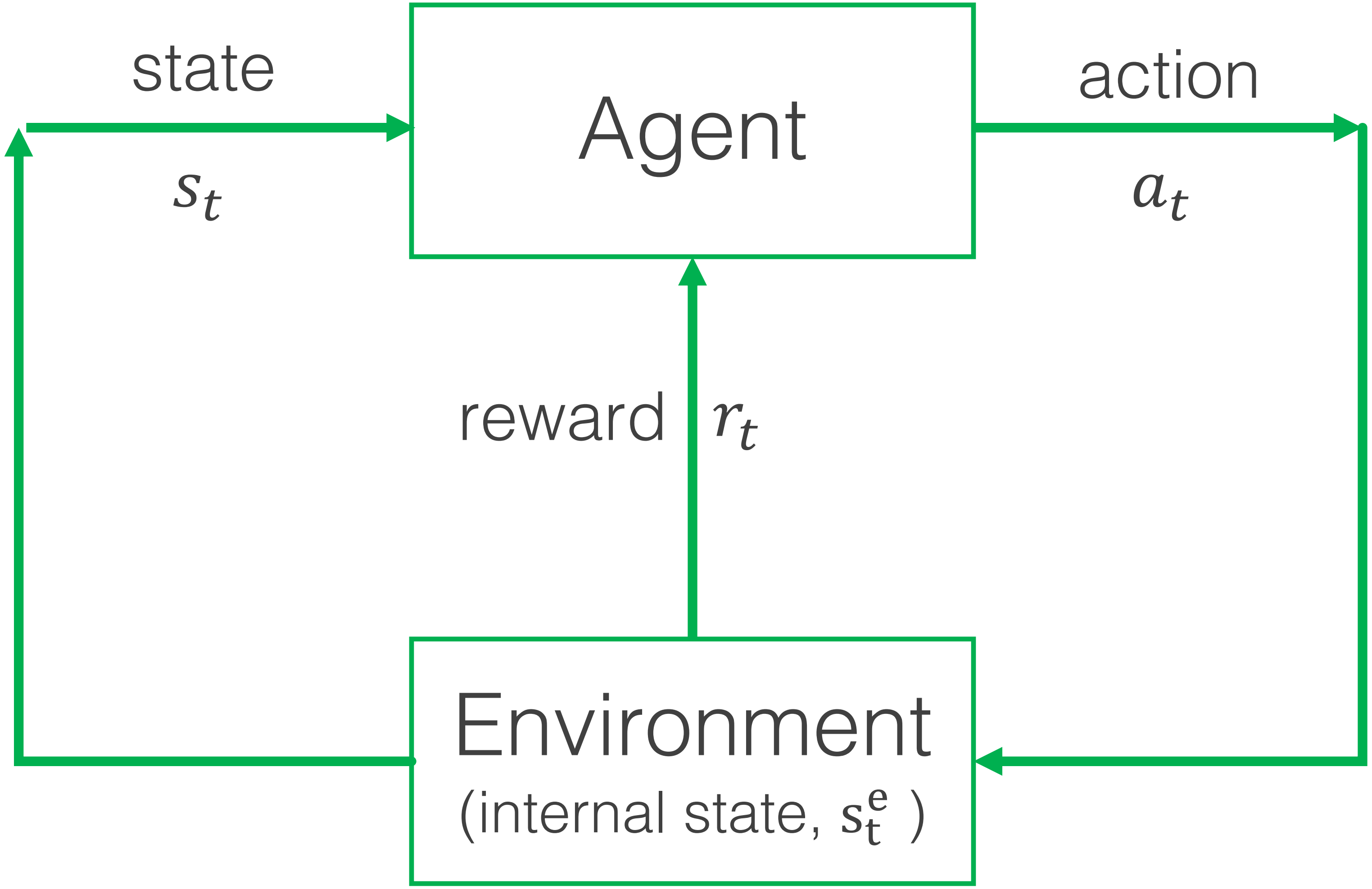
Credit assignment: which action(s) led to the reward

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

Exploration (of untried actions) vs exploitation (of current knowledge)

David Silver, 2015

Agent-environment Interaction



Agent at each step t…

Executes action a_t
Receives state, s_t
Receives scalar reward, r_t

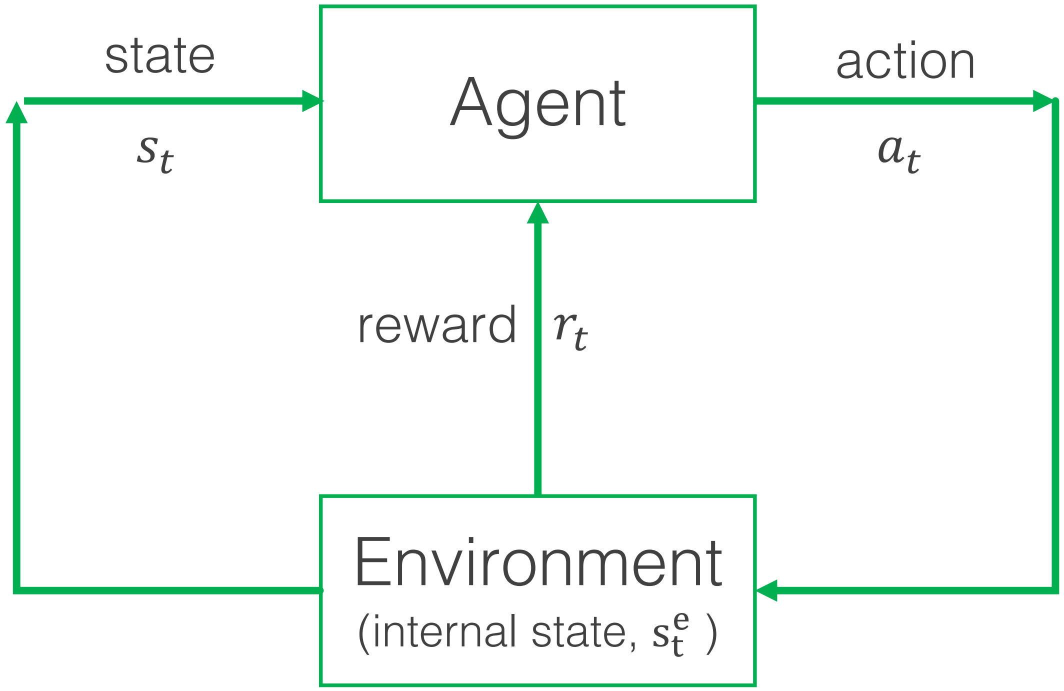
Environment at each step t…

Receives action a_t
Emits 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

RL Components



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

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

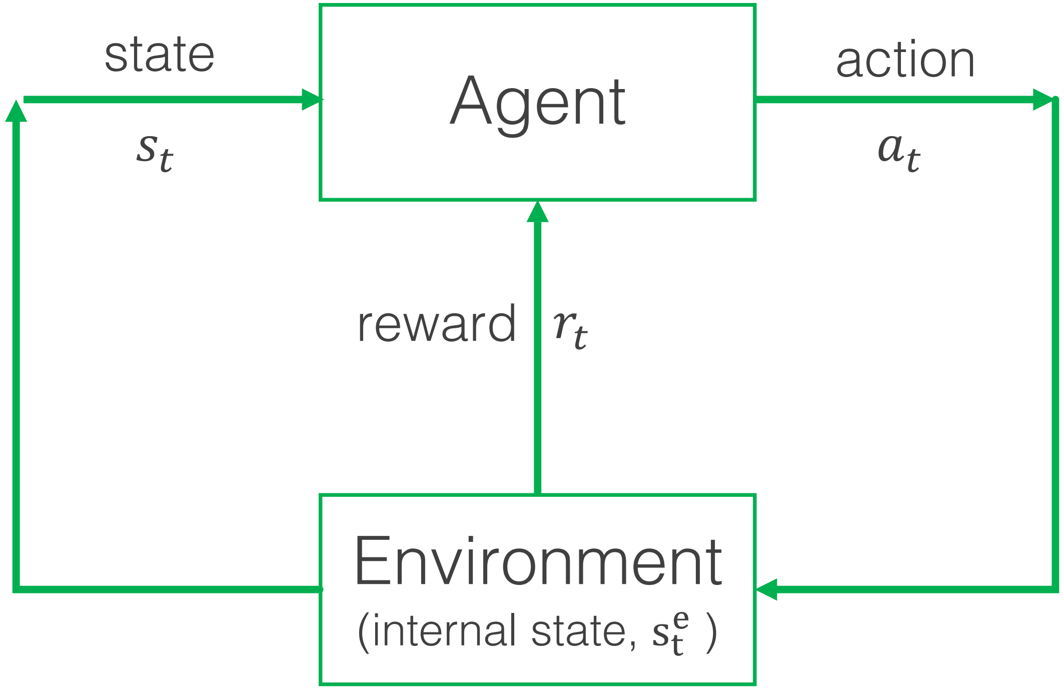
Reward function (the goal to max), r_t

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

Value function (state reward), $v(s_t)$

- Total expected reward from a state
- How good is each state

Policy



Policy, $\pi(s_t)$

- Agent's way of behaving at a given time
- Maps state to actions

Deterministic: $a = \pi(s_t)$

Stochastic: $\pi(a|s_t) = P(a_t = a|s_t = s)$
Helps us "explore" the state space

RL tries to learn the "best" policy

Goals and rewards

Rewards are the **only way** of communicating what to accomplish

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)

Returns / cumulative reward

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

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

Continuous 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

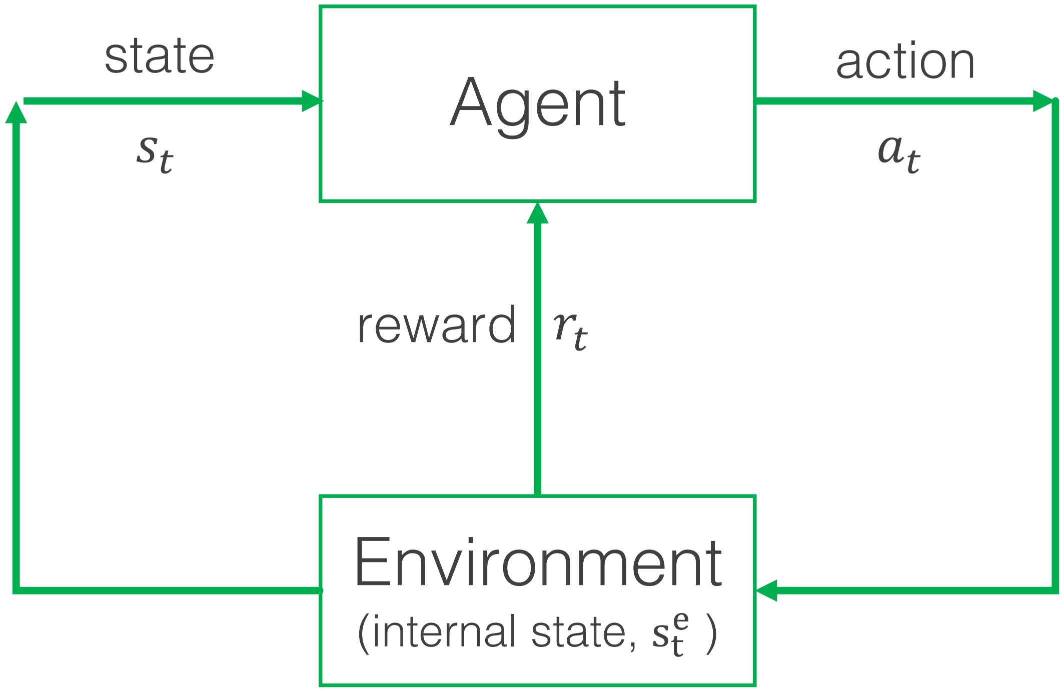
This makes the agent care more about immediate rewards

Value function

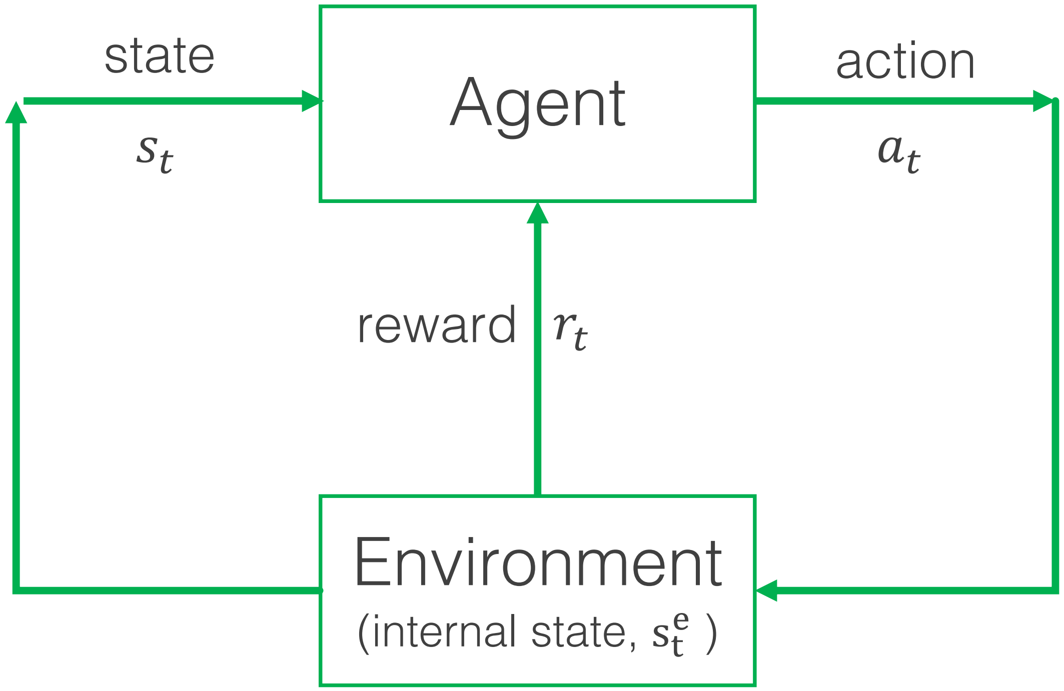
Value function, $v(s_t)$

- How good is each state / action
- Total expected reward

$$v_\pi(s) = E_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s]$$



Model



Model

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 these predictions

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

Reinforcement Learning Examples

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

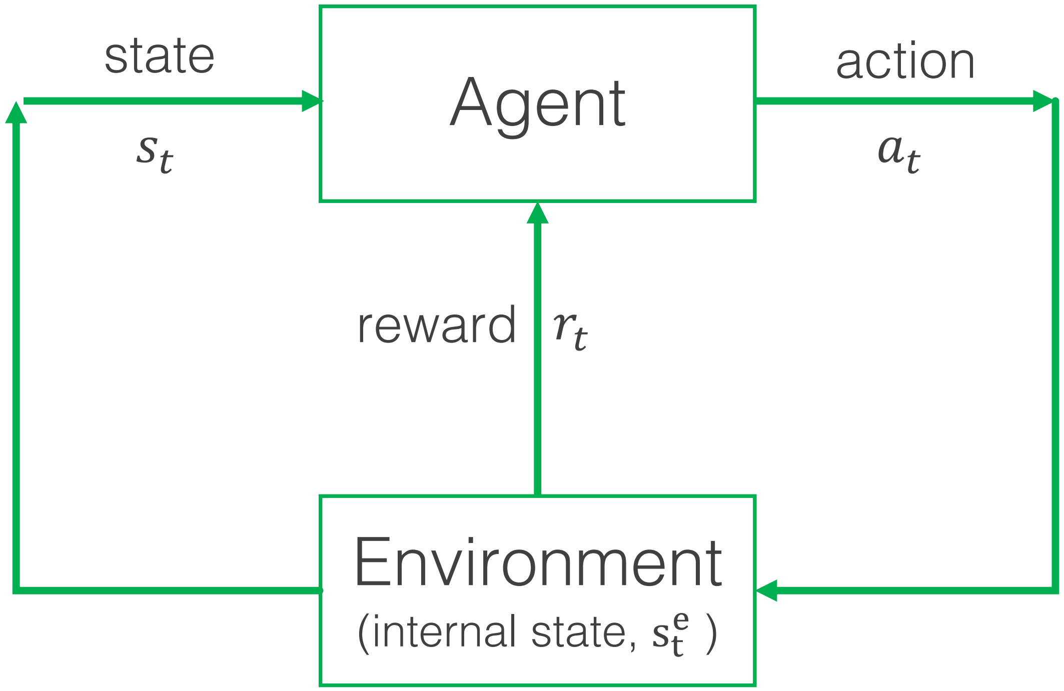
Balancing an inverted pendulum: https://youtu.be/b1c0N_Fs9wc

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

Car Drifting: https://youtu.be/opsmd5yuBF0

RL is a unifying framework for a wide range of problems

Reinforcement Learning Components



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

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

Reward function (the goal to max), r_t

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

Value function (state reward), $v(s_t)$

- Total expected reward from a state
- How good is each state

Multi-armed Bandit

Multi-armed bandit example

You have N slot machines to choose from

For each trial/episode, you take an **action**: pick one machine to play

Each machine has an unknown probability of payoff/**reward**

The reward distributions are unknown

Only 1 "state"

i.e. create a policy, $\pi(s)$

How do we choose actions to maximize our total rewards?

(if we knew the best machine, we'd always pick it, but we have to learn it)

Multi-armed Bandit Demo

https://dataorigami.net/blogs/napkin-folding/79031811-multi-armed-bandits

Multi-armed bandit

The "true" **value** of an action is $v^*(a)$

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

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

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

As we take action a more, our value estimates improve

Sutton and Barto, 1998

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

Greedy action:

Select a* $= \arg\max_a V_t(a)$

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

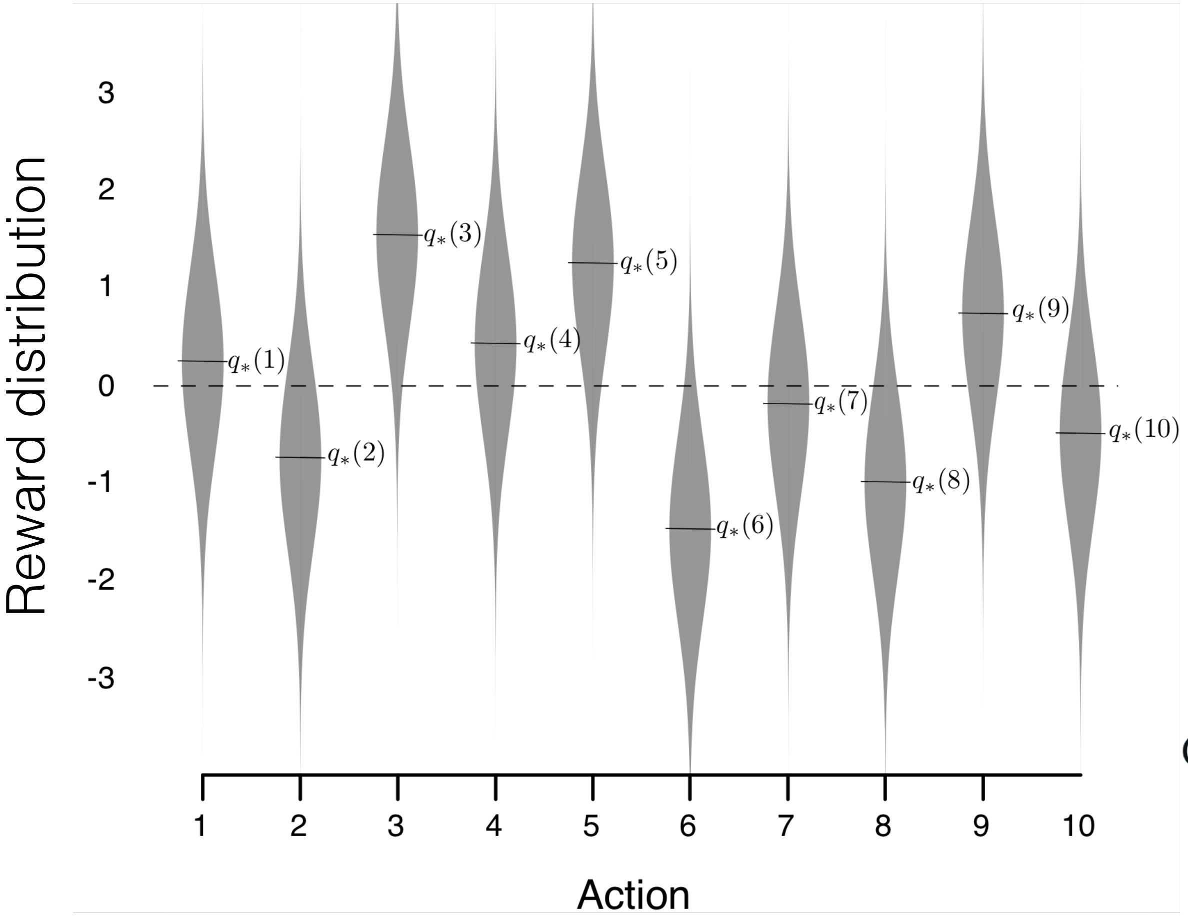
Alternative:

Select the action probabilities based on the expected value

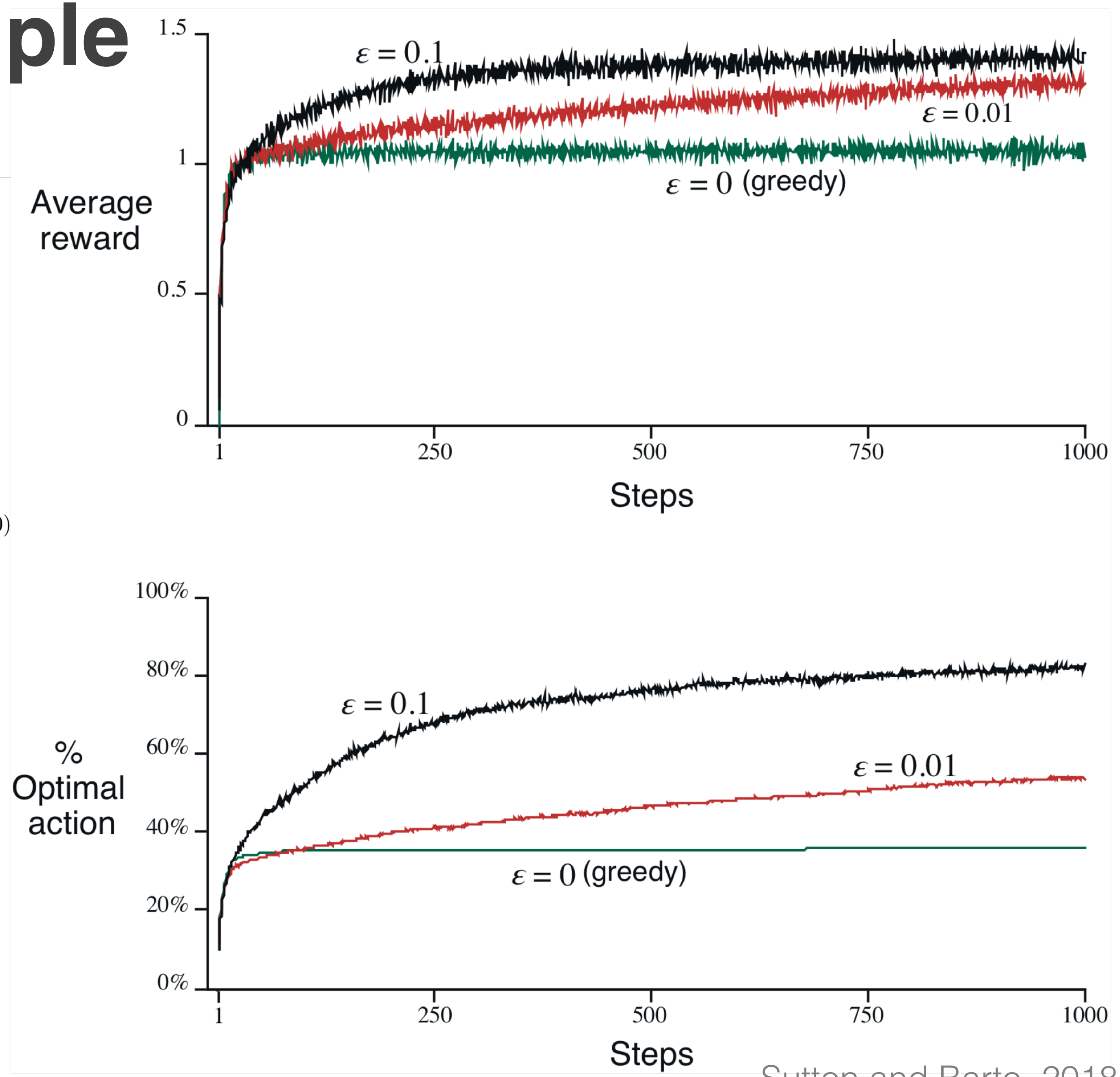
Probability of selecting action a = $\dfrac{\exp(v_t(a)/\tau)}{\sum_{b=1}^{n} \exp(v_t(b)/\tau)}$

Can reduce τ over time to reduce exploration

10-Armed Bandit Example



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



Sutton and Barto, 2018

Next steps

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

State representations and Markov decision processes (MDPs)
(with an aside on Markov processes)

Mathematically formulating the RL problem with MDPs

Methods for solving RL problems in practice