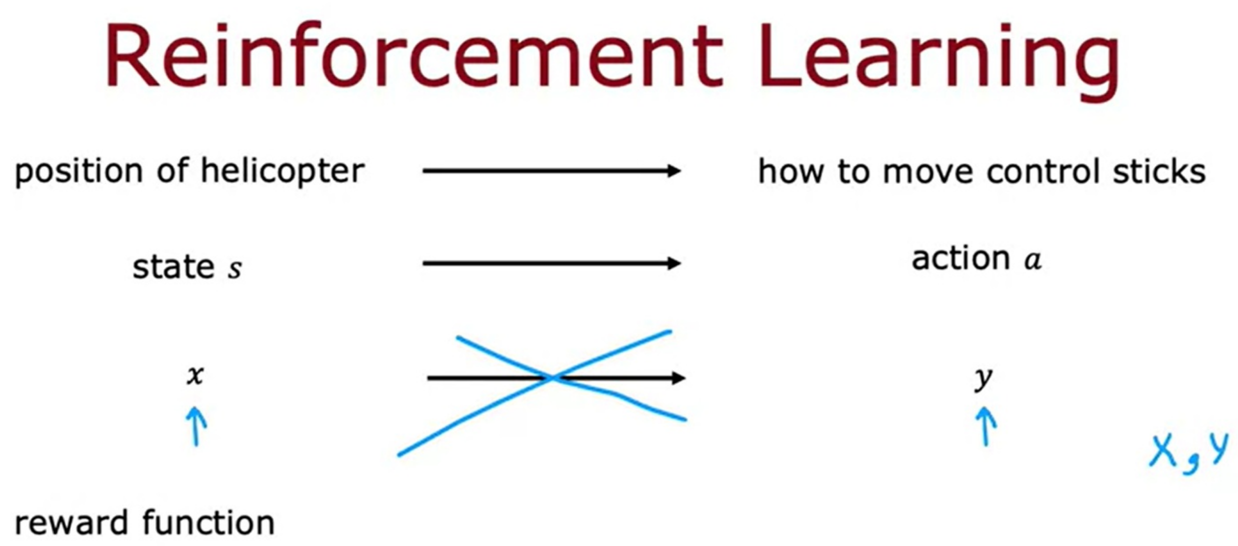
Week 3

Reinforcement Learning

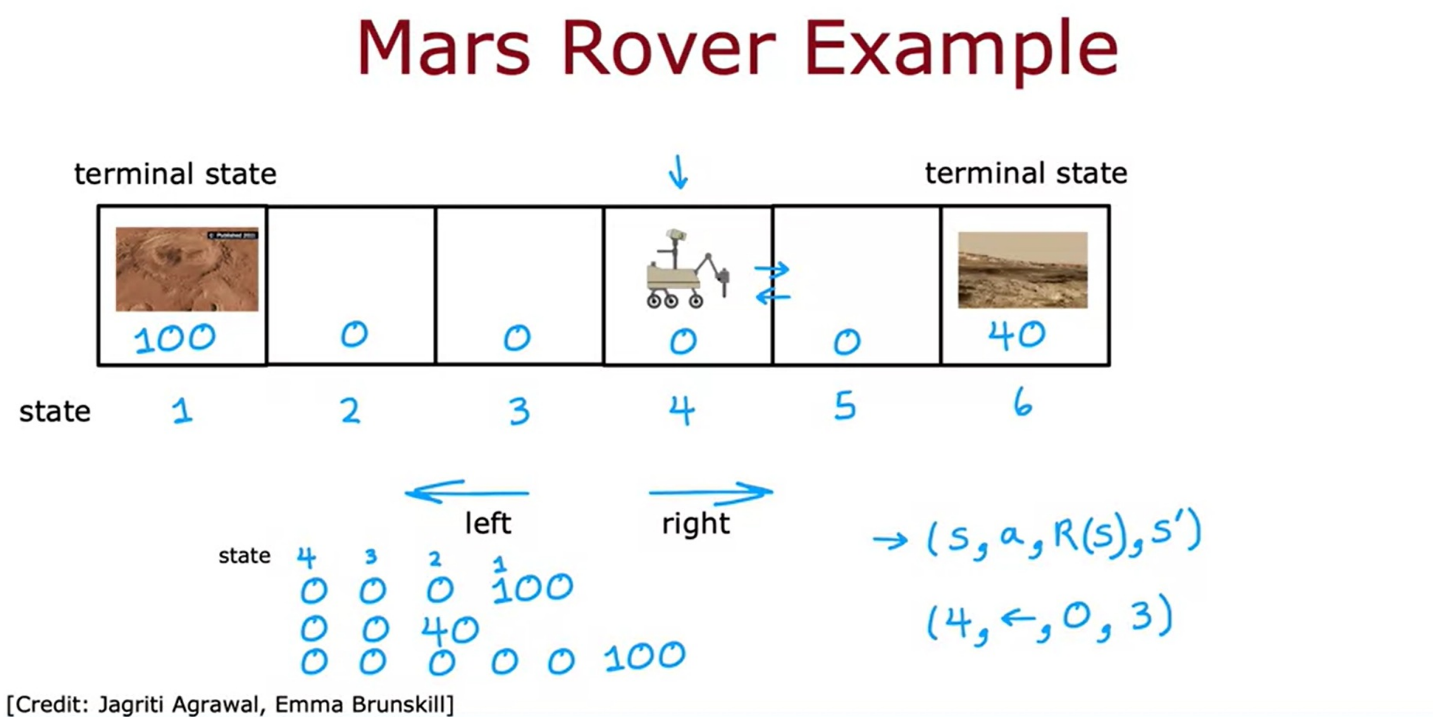
Why use Reinforcement Learning?

- Supervised learning is not a good approach → Very difficult to get use a dataset to get an ideal action → Many ambiguities/nuances on what is the right action to take
- You don't need to know the exact actions to take → Robot will automatically learn based on the reward function



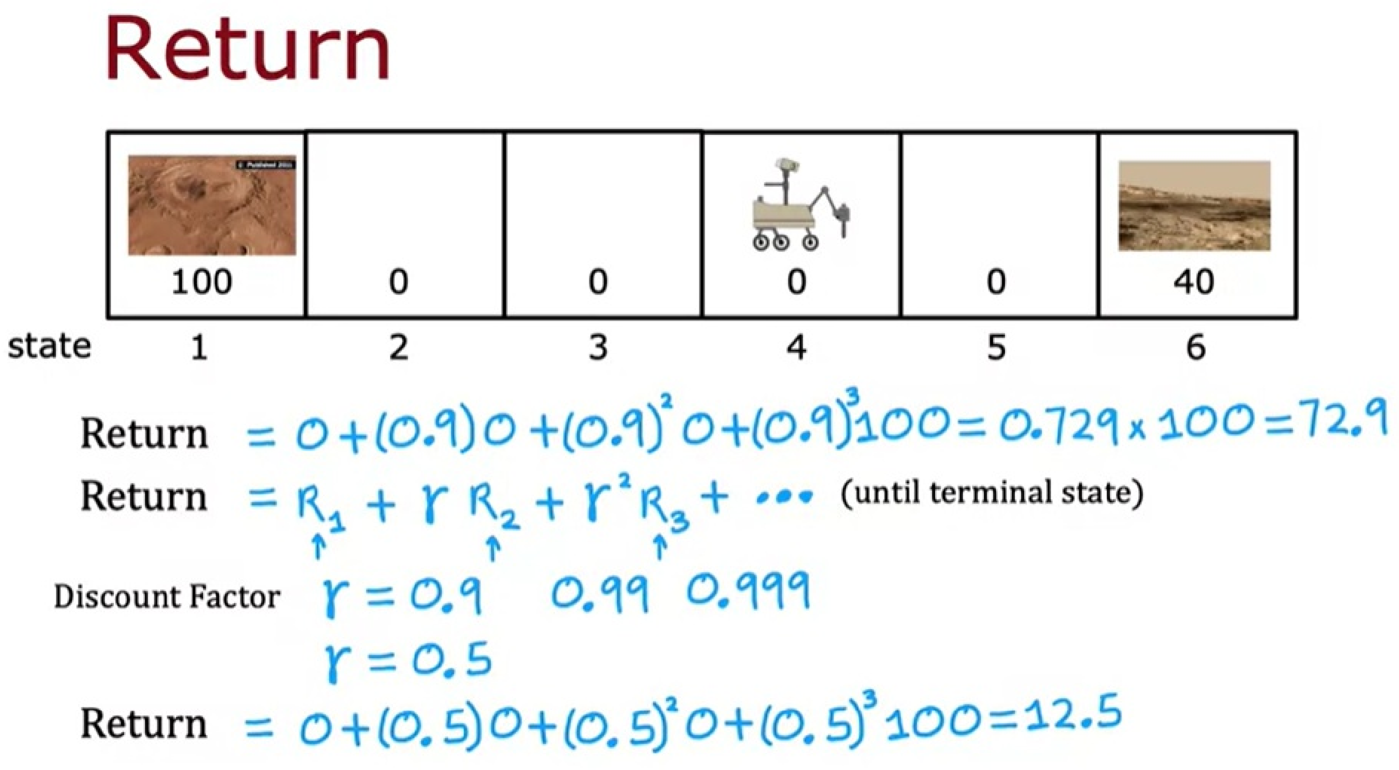
Overview of how Reinforcement Learning Works

- Robot will be aware of:
  (state, action, reward, new state)



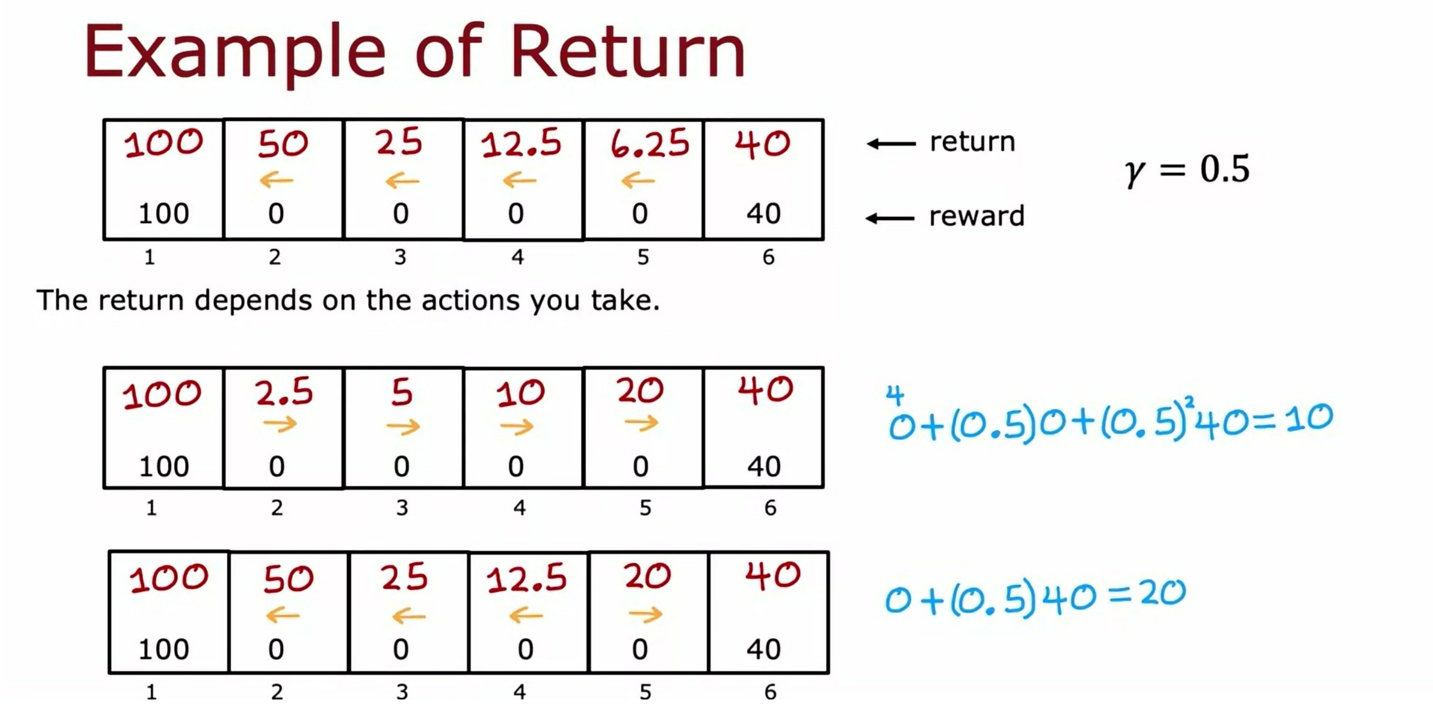
Return

- Idea: Penalize the additional states taken to reach the terminal state by adding a discount factor (γ)(γ)(γ)
- high γ = agent seeks long-term rewards



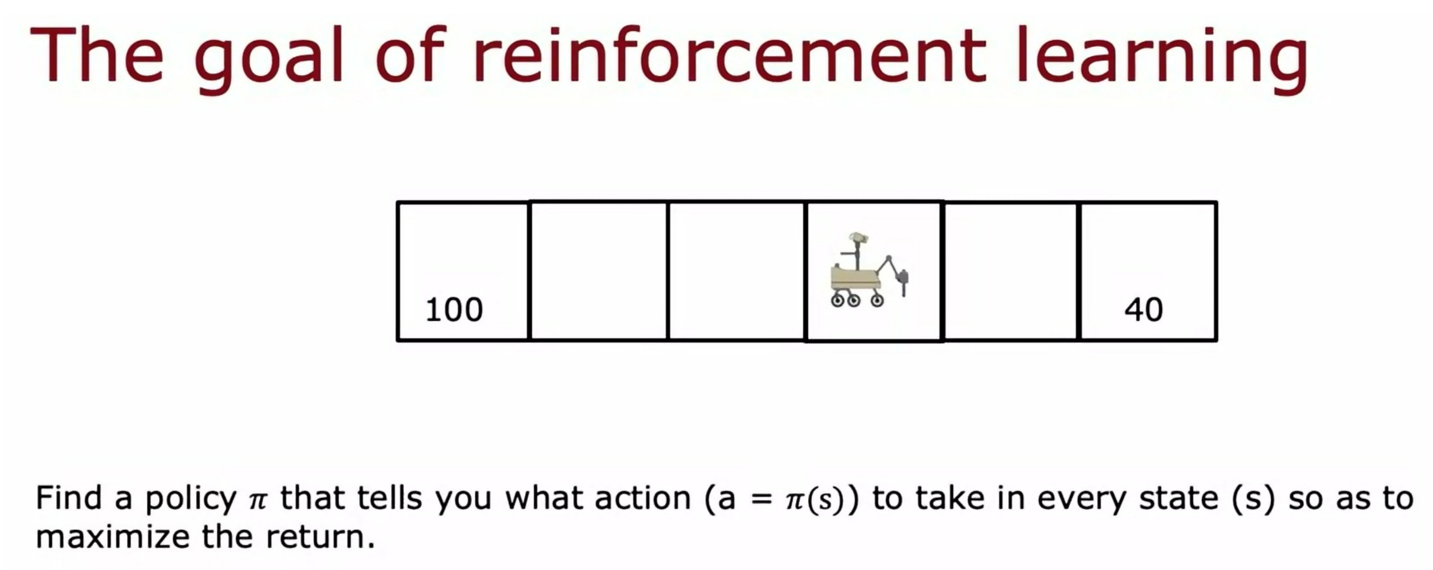
Example

- Reward at each state (depending on the action) if discount factor γγγ **is 0.5**
- Keep in mind that a state can have **negative** reward



Goal of Reinforcement Learning - Policy

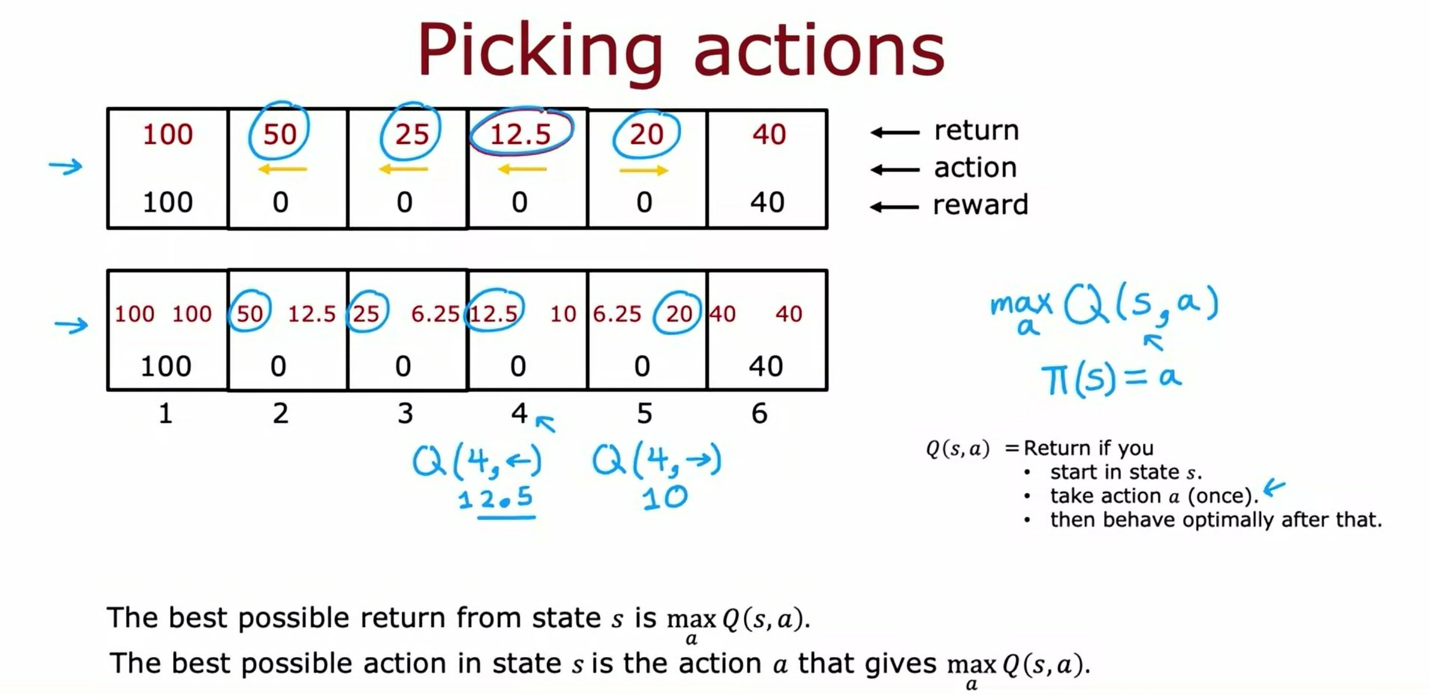
- Create a policy that maps a state to an action



State-action Value Function

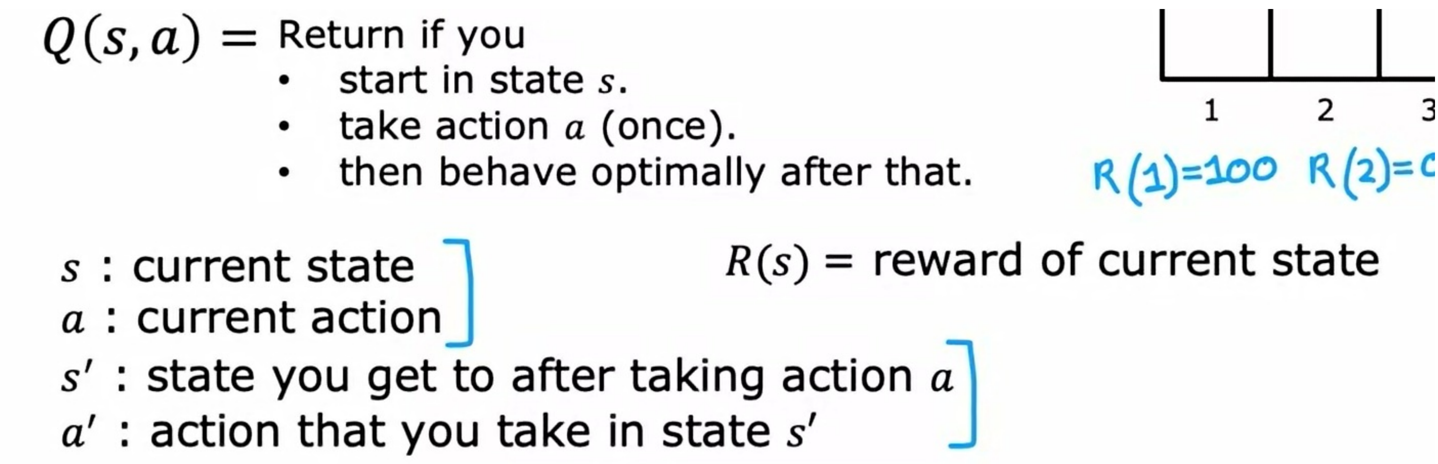
Q-learning function

- Q(s,a)Q(s,a)Q(s,a) represents the **maximum potential reward** possible if a particular action aaa is taken at state sss, assuming that steps taken after are **optimal** (which depends on the policy).



Bellman Equation

Notations:



- Bellman equation is a recursive function
- maxaQ(s′,a′)max_aQ(s',a') maxaQ(s′,a′) represents the return from optimal behavior from s′s's′

Explanation of Bellman Equation

$Q(s,a) =$ Return if you
- start in state s.
- take action a (once).
- then behave optimally after that.

$s \rightarrow s'$

→ The best possible return from state s' is $\max_{a'} Q(s', a')$

$$Q(s,a) = \underbrace{R(s)}_{\text{Reward you get right away}} + \gamma \underbrace{\max_{a'} Q(s',a')}_{\text{Return from behaving optimally starting from state } s'.}$$

$$R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \cdots$$

$$Q(s,a) = R_1 + \gamma [R_2 + \gamma R_3 + \gamma^2 R_4 + \cdots]$$

Random Stochastic Environment
- Misstep probability → The next state can be different from the optimal based on probability
- Maximize average expected return

Expected Return

Goal of Reinforcement Learning:

Choose a policy $\pi(s) = a$ that will tell us what action a to take in state s so as to maximize the expected return.

Bellman Equation:

$$Q(s,a) = R(s) + \gamma\, \underbrace{E[\max_{a'} Q(s',a')]}_{2 \text{ or } 4}$$

3 ←

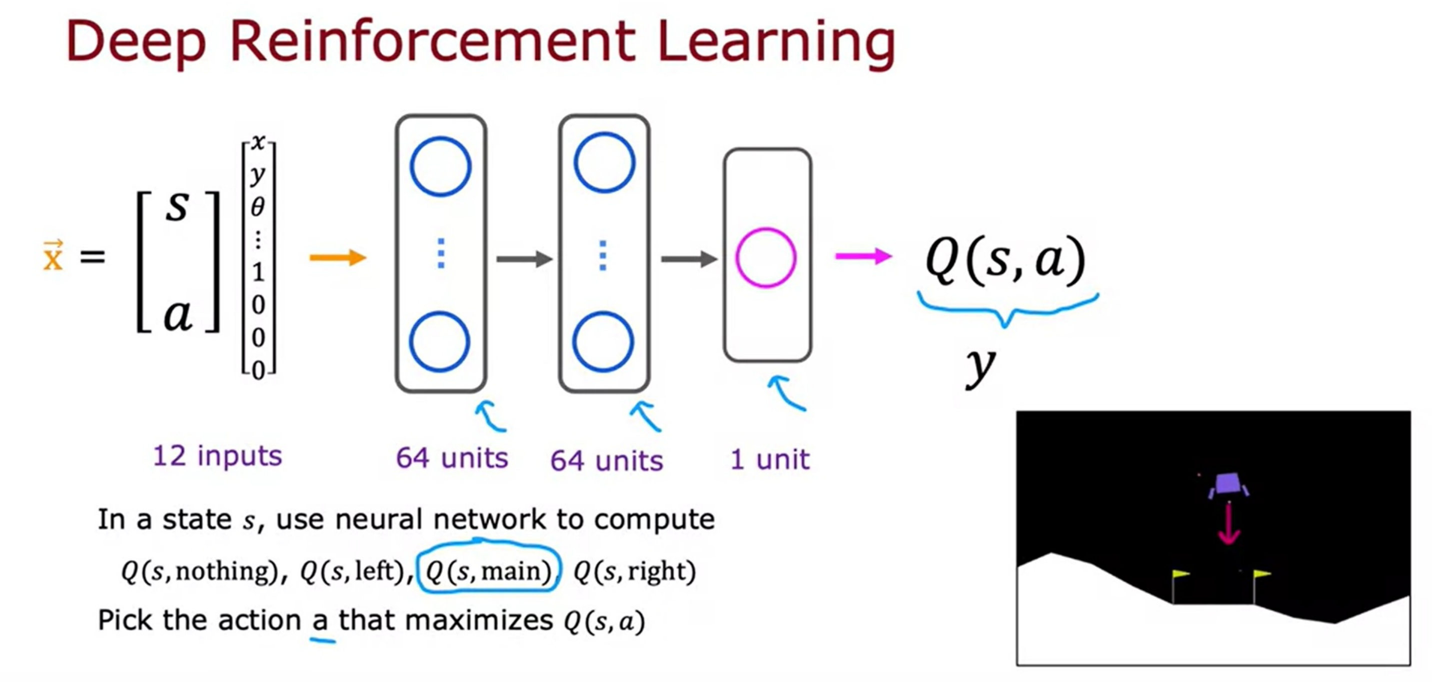
Continuous State Spaces

Discrete vs Continuous states

- Instead of simply moving from state 1 - 6, think about helicopters moving x, y z, rotation, which is continuous

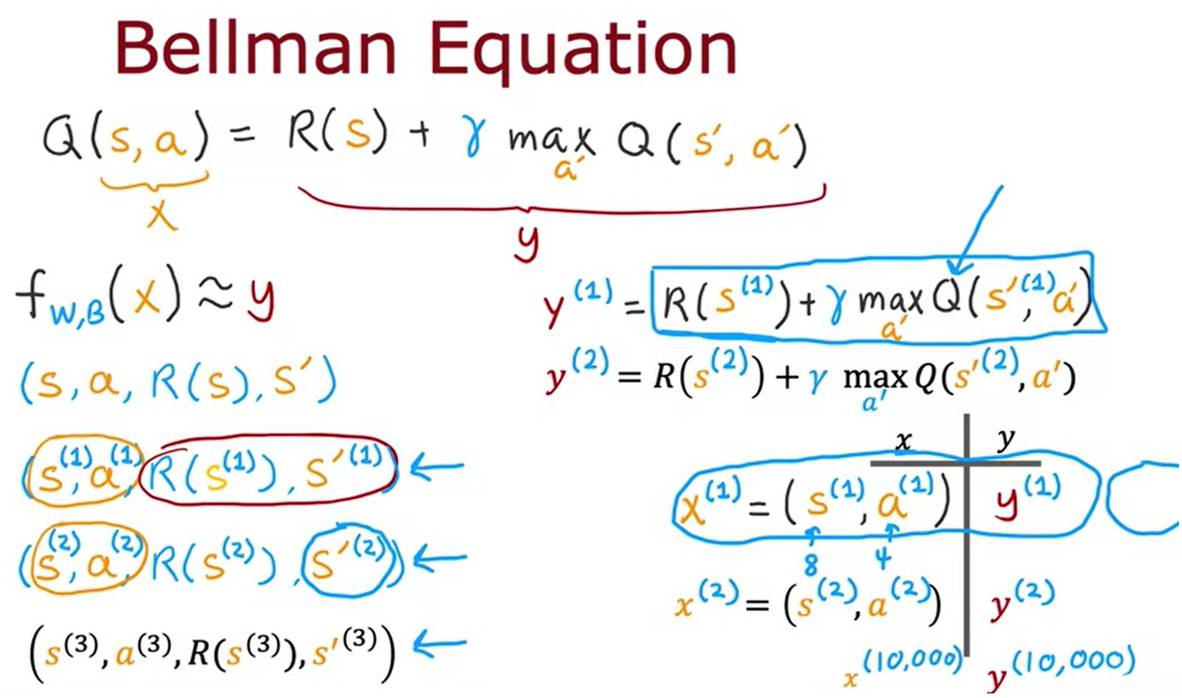
Deep Reinforcement Learning
- In the example below, the 12 inputs come from:
  - 8 states
  - 4 q-learning functions (1 if the function is used, 0 otherwise) → actions

- How do we train a neural network to output Q(s,a)Q(s,a)Q(s, a)? Use Bellman Equation to create a training set of x and y.



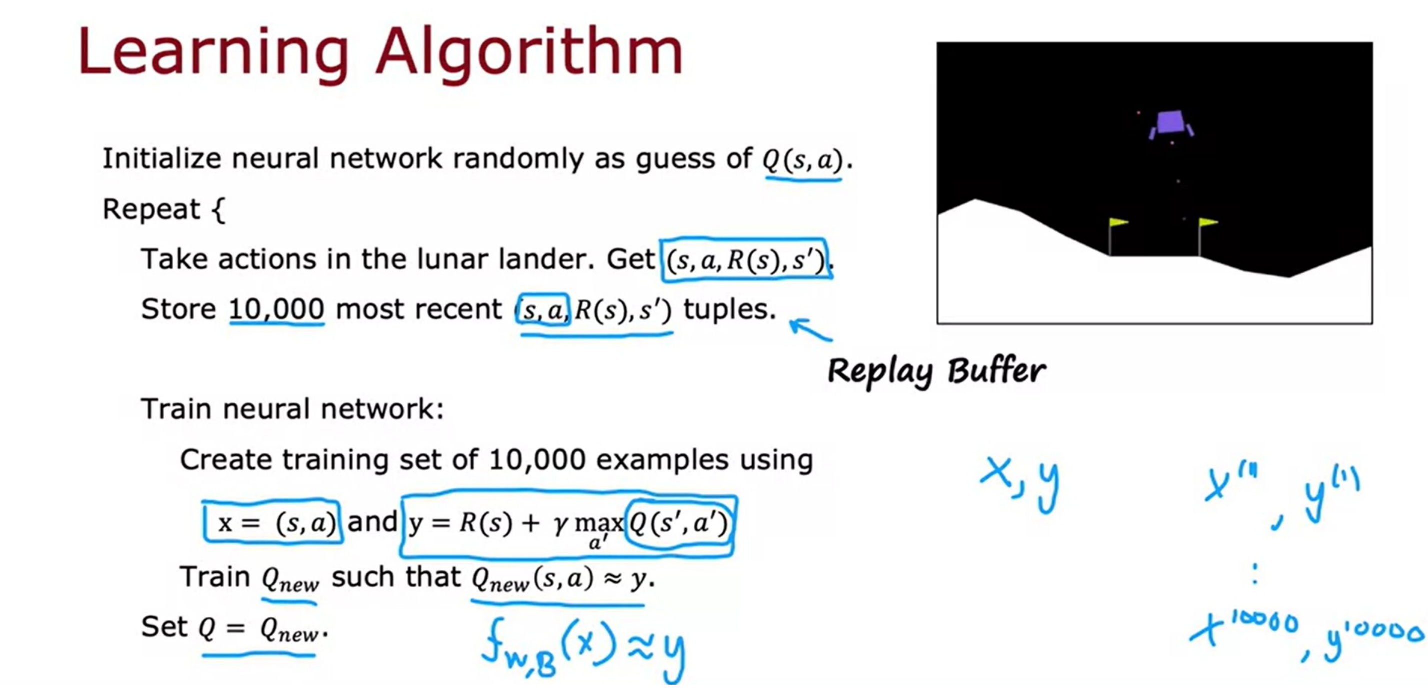
Creating training set for Reinforcement Learning

- Note that s’1s'^1s’1 is not fixed → Q is from a random guess



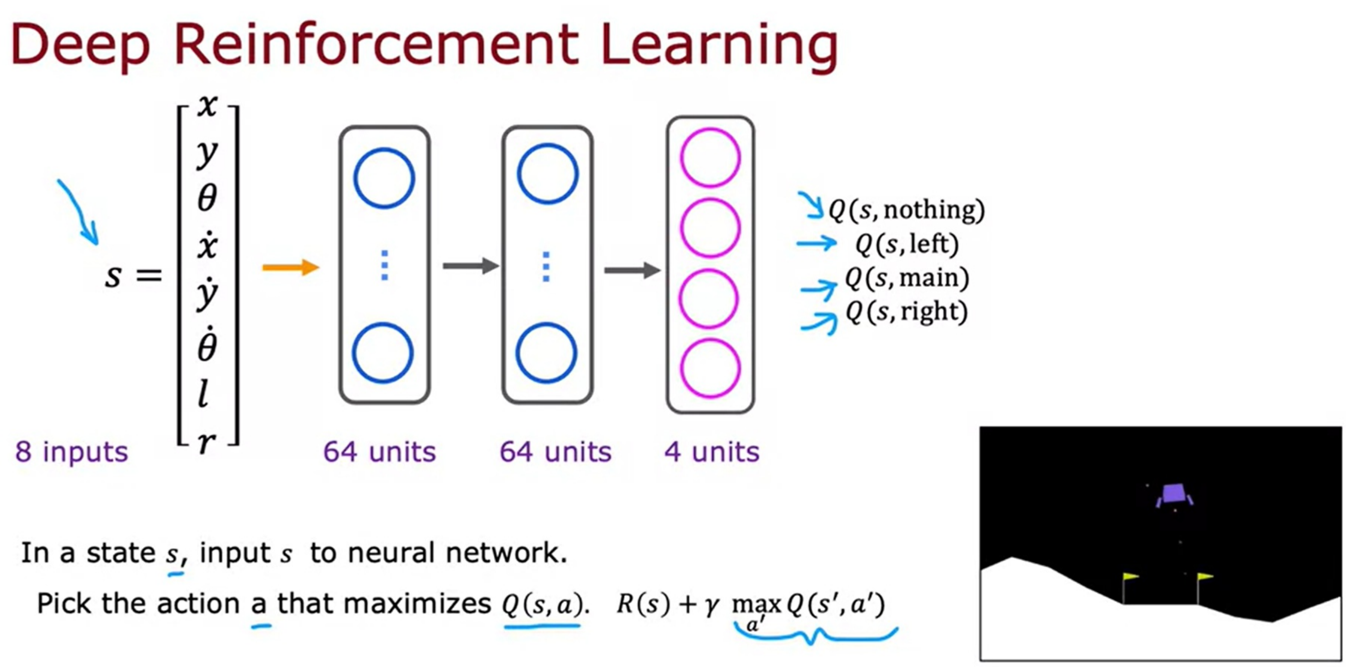
Learning Algorithm

- As you run this algorithm for many iterations, the accuracy will improve



Algorithm Refinement

- Inefficient → The original requires you to carry out inference 4 times in the neural network for some state
- Train neural network to output all 4 values simultaneously
  - 8 inputs only (for each state)
  - Given a state s, we only need to run the inference once!
  - Bellman Equation is also more efficient due to the 4 Q functions outputted in each inference



How to choose actions while still learning?

- Option 1 can be bad as it can be influenced easily by the random Q()
  - E.g Q(s,main) generated could be low, and using option 1, main thrusters will never be used
- Hence an exploration step is needed to find an alternate Q() which can sometimes be a good option to take
- **ε-greedy policy (ε = 0.05)**

How to choose actions while still learning?

In some state s

Option 1:

Pick the action a that maximizes $Q(s,a)$.

Option 2:

→ With probability 0.95, pick the action a that maximizes $Q(s,a)$. Greedy, "Exploitation"

→ With probability 0.05, pick an action a randomly. "Exploration"

ε*–greedy policy* $(\varepsilon = 0.05)$

0.95

Q(s,main) is low

↑

a

Start ε high

$1.0 \rightarrow 0.01$

Gradually decrease

Limitations of Reinforcement Learning

Limitations of Reinforcement Learning

- Much easier to get to work in a simulation than a real robot!
- Far fewer applications than supervised and unsupervised learning.
- But … exciting research direction with potential for future applications.