Reinforcement Learning

[Machine Learning - Coursera](#)

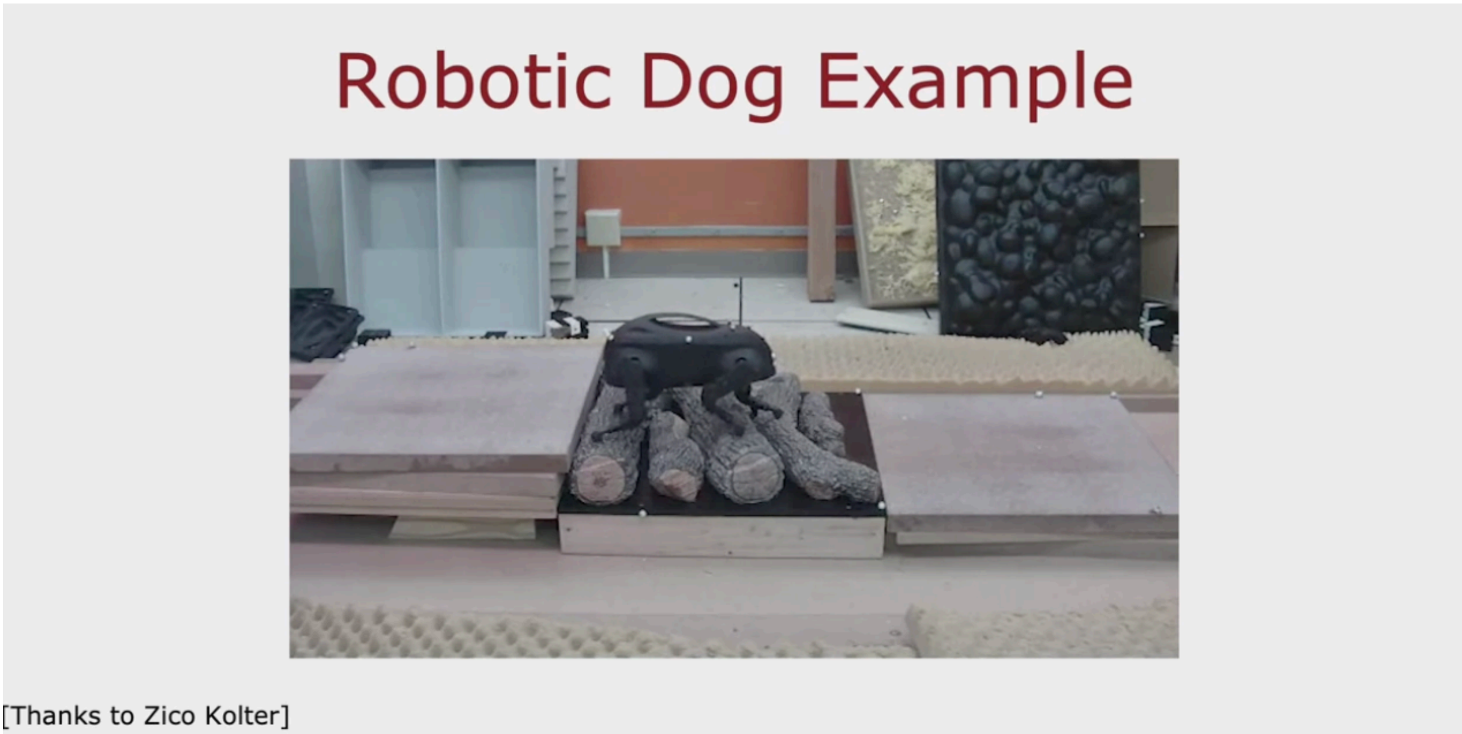
Position is given -> How to move the control sticks?
state s -> action a

Supervised Learning?
x -> y
for x's we will find y's from our knowledge?

it is too ambigious for it, flying is unpredictable.

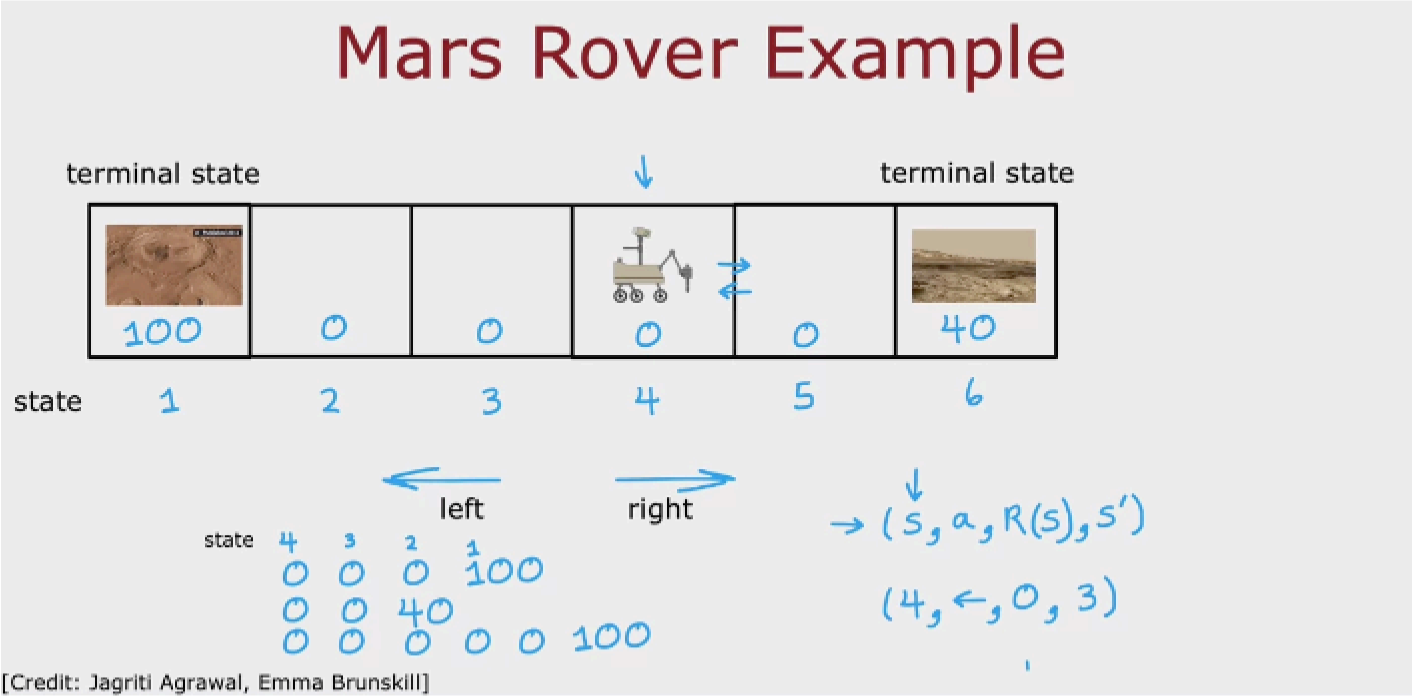
how to make a dog behave? Good dog! Bad dog!

you have to tell it what to do, rather than how to do it.
reward/punishment



Financial trading?

terminal state when gets to the one of the *states*, day ends.

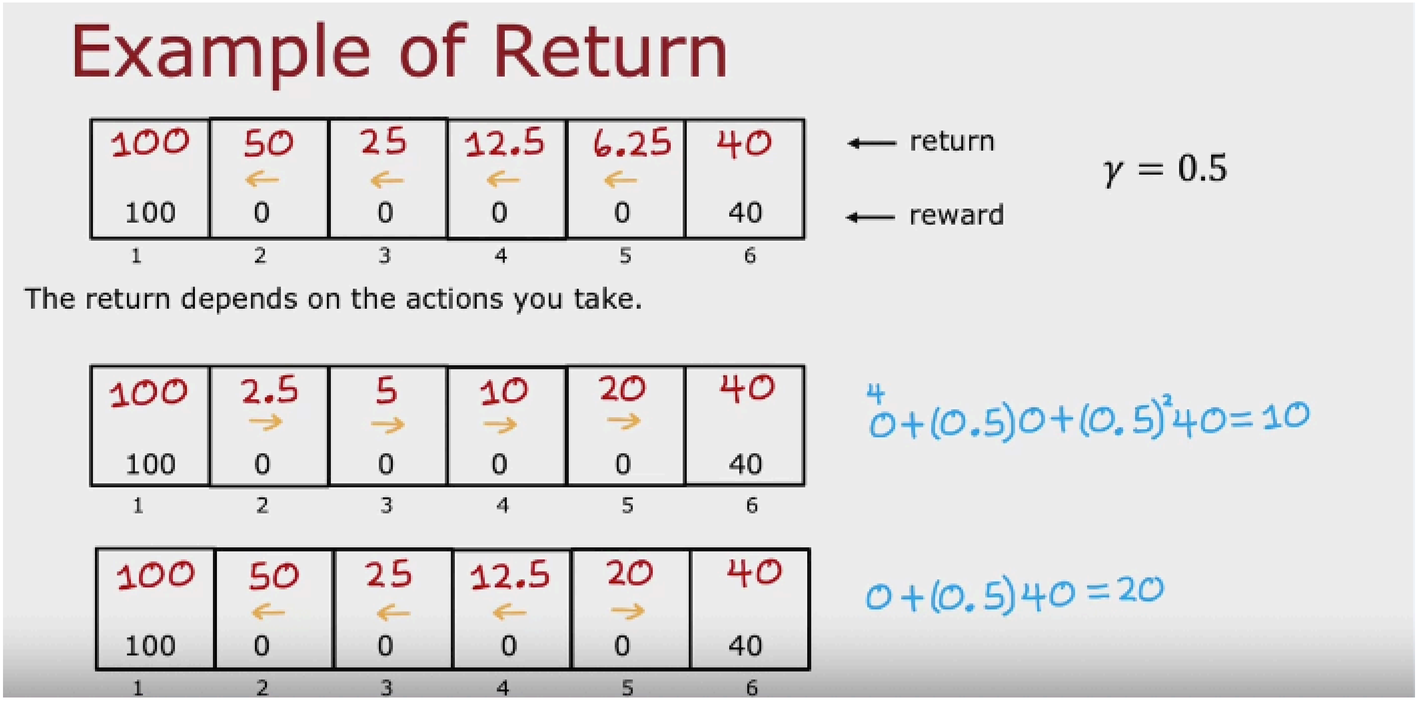


The Return in Reinforcement Learning

discount factor of r = (ex) 0.9

Return = R + r.R + r^2.R + ...

Finance: Interes rate, time value of money.

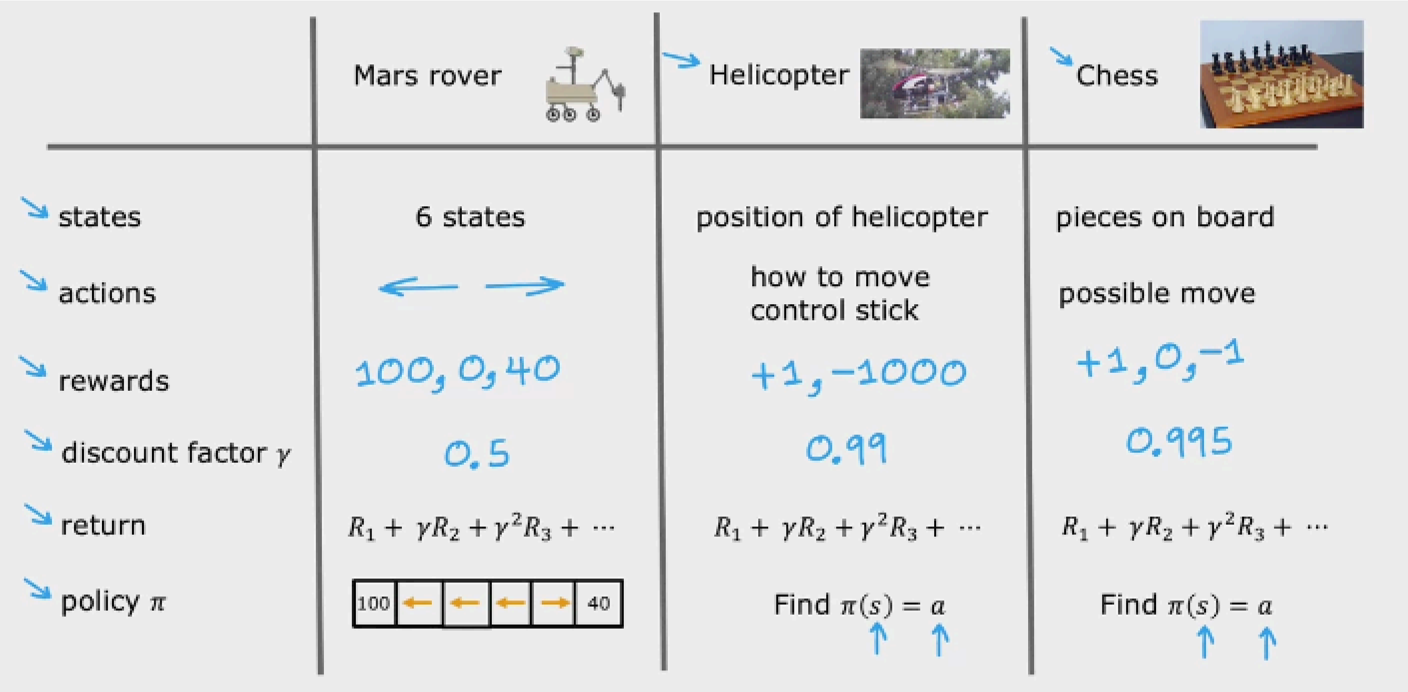


Policies in reinforcement learning
can choose always go *direction*

state -> (policy) -> action

goal of RL: find a policy (pi) that tells you what action to take in every state so as to maximize the return.

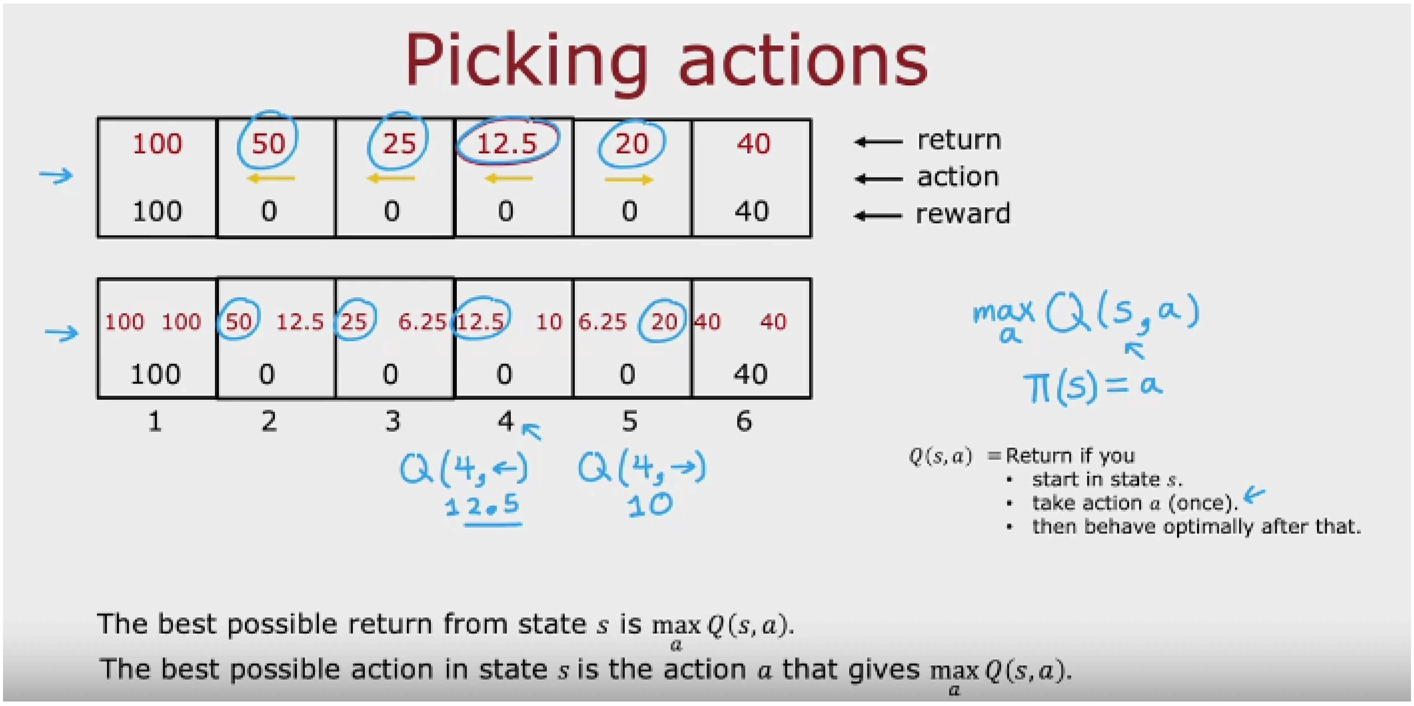
states, actions, rewards, discount factor, return, policy(pi)



Markov Decision Process (MDP)

future depends on where you are now, not on how you got here.

State Action Value Function (Q-function)



Bellman Equation:

Bellman Equation

$R(1)=100$ $R(2)=0$ ... $R(6)=40$

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

s : current state
a : current action
s' : state you get to after taking action a
a' : action that you take in state s'

$R(s)$ = reward of current state

$$Q(s,a) = R(s) + \gamma \max_{a'} Q(s',a')$$

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 $\max_{a'} Q(s',a')$

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

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

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

Discrete vs Continuous State

how would you represent the position of the helicopter/truck/...

pos, pitch, angle. with a vector:

LUNAR LANDER

actions:

do nothing

left thruster

main thruster

right thruster

s=

[

x

y

velocity

angle

angular velocity

l sitting on ground?

r sitting on ...?

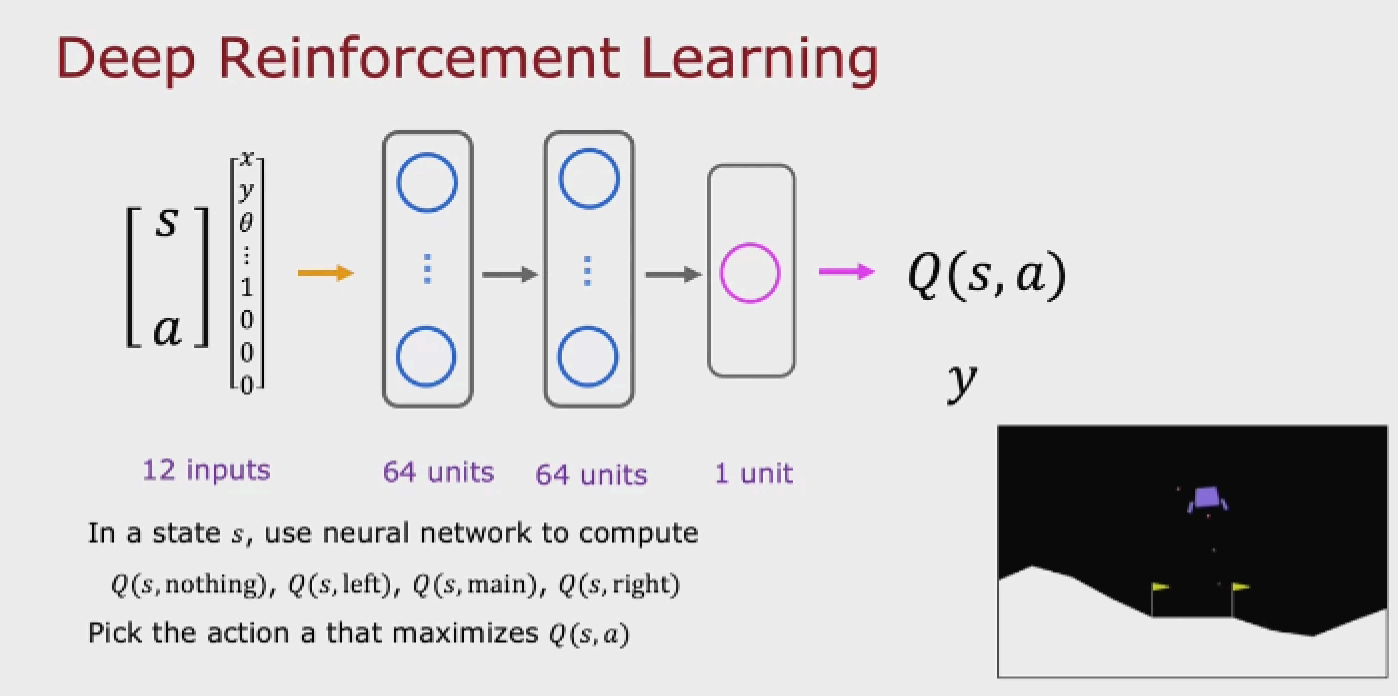
]

custom reward function:

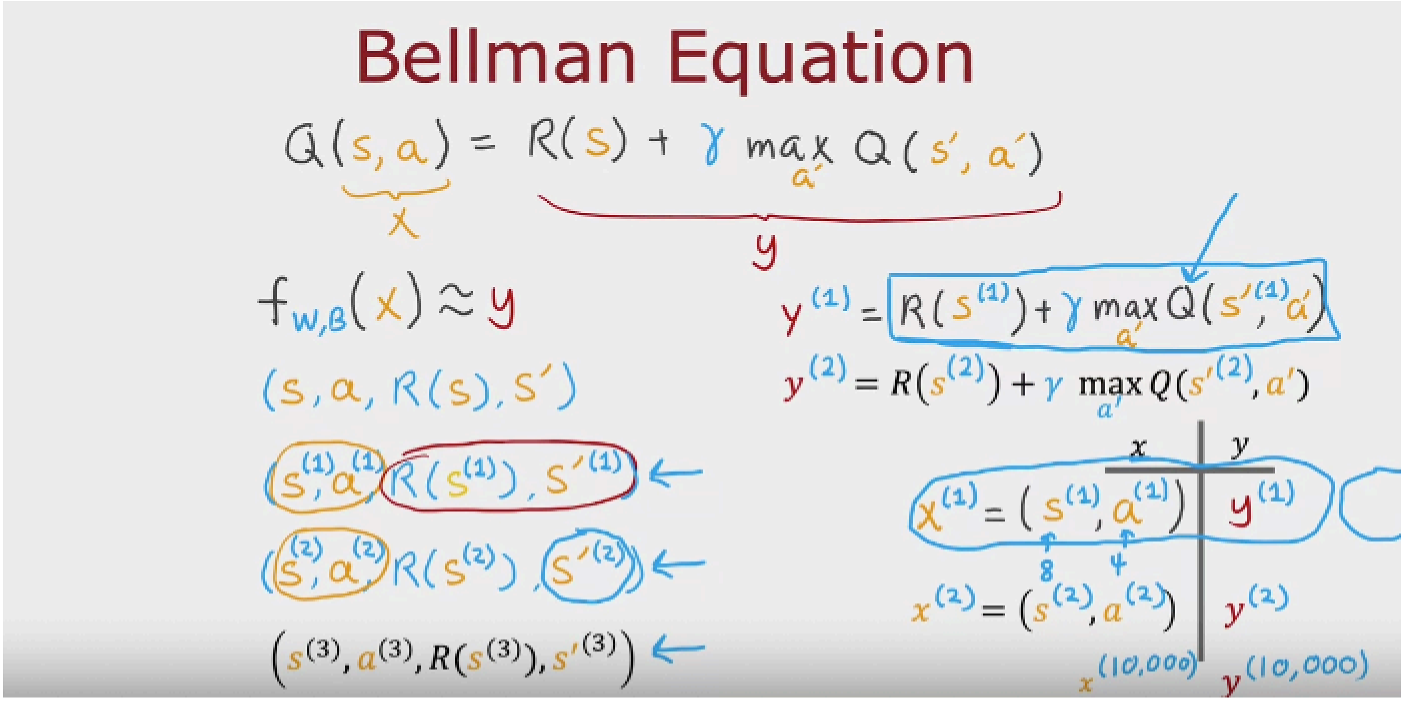
crush -100

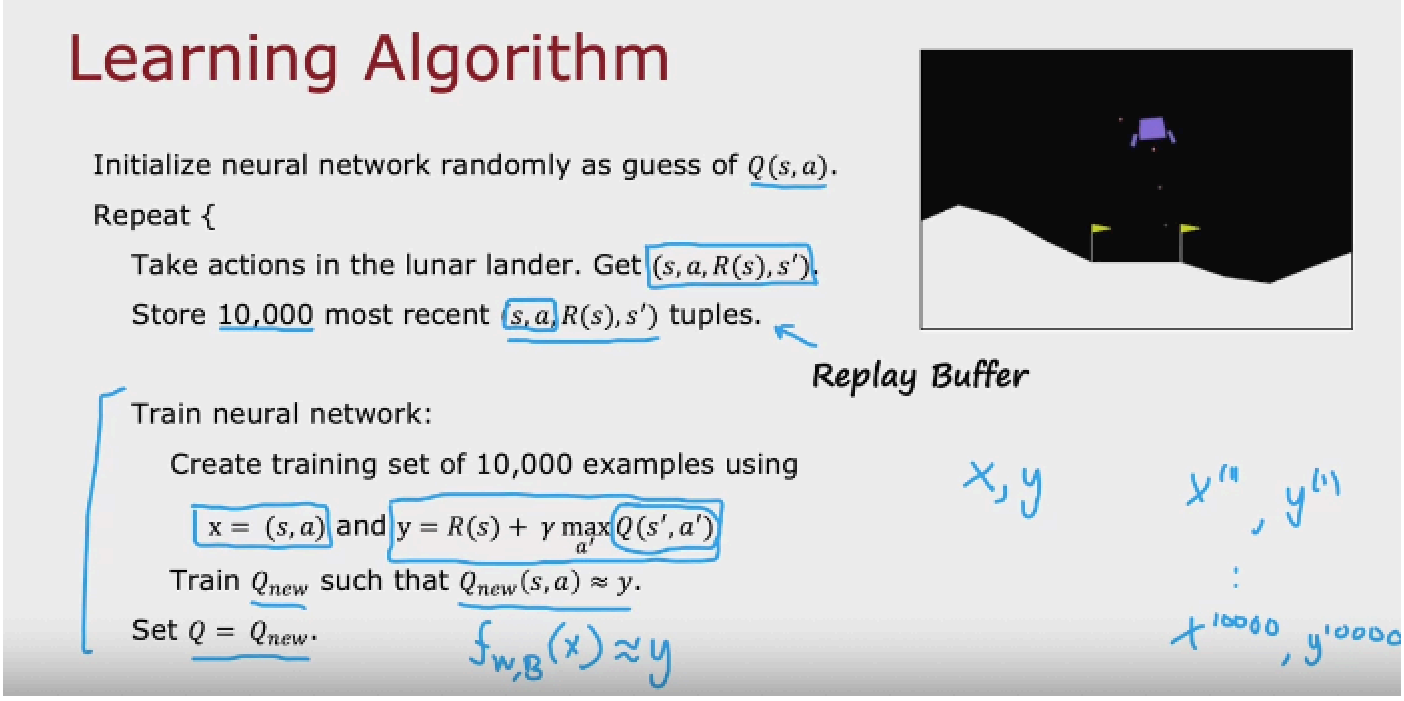
soft land +100

leg grounded +, fire main engine ---, side thrust -, get to landing pad ++

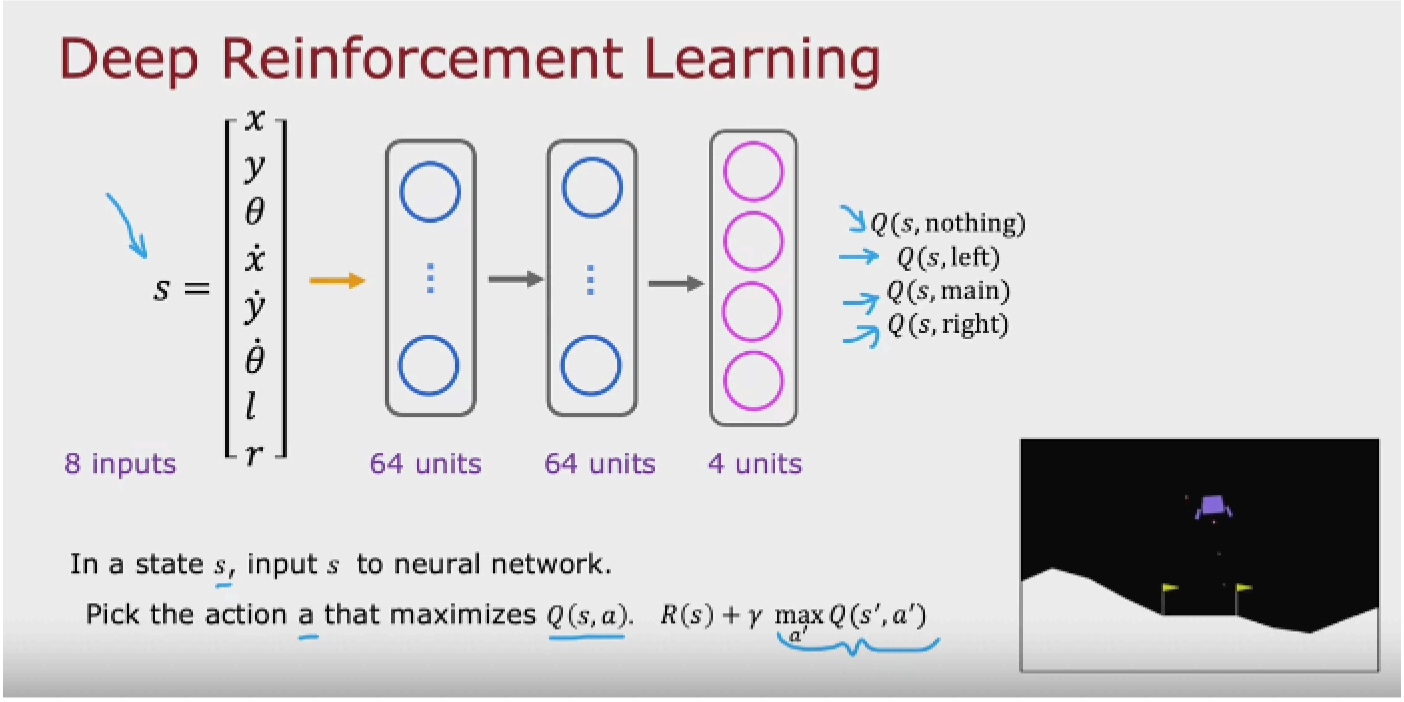


Create a supervised learning data for the NN output (Learning the Q(s, a) function)





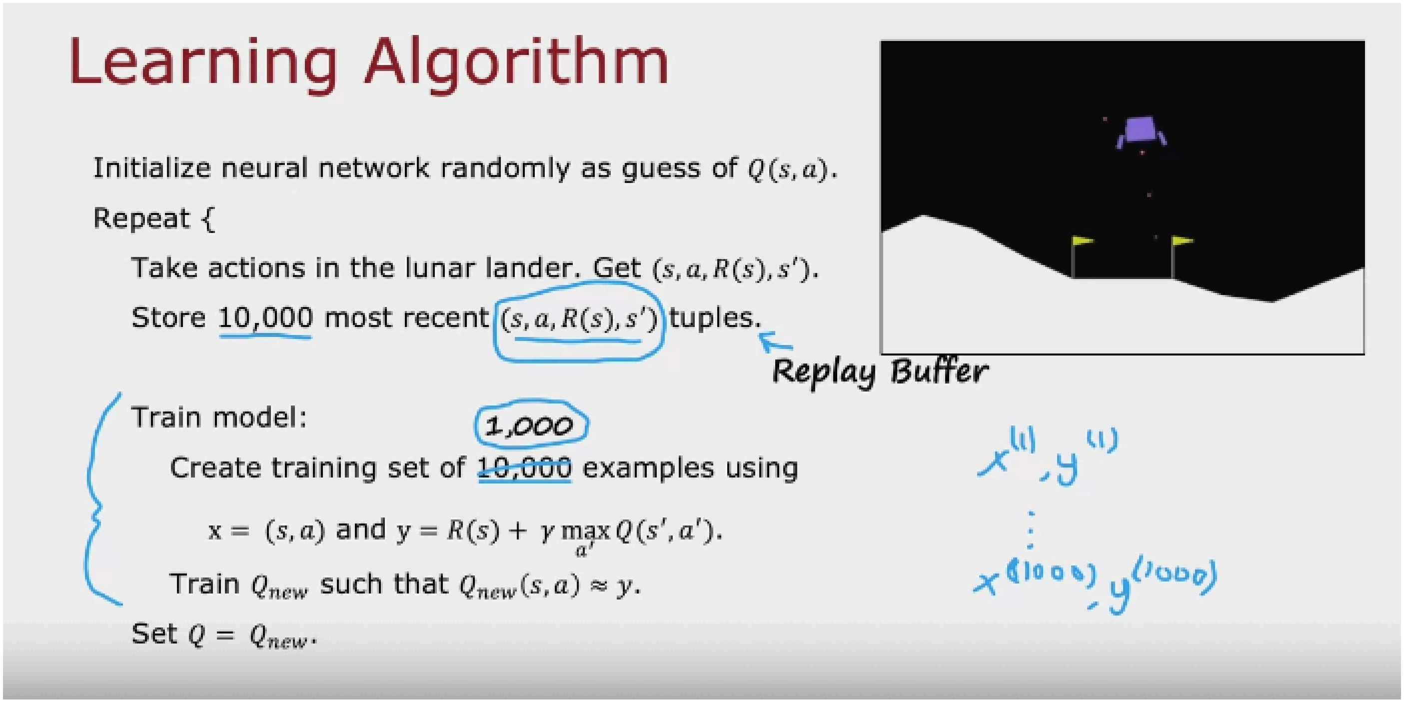
Continuous State Spaces:
Improved NN Architecture



Algorithm refinement: ϵ-greedy policy
How to choose actions while we are still learning?
at some probability, pick an action *a* randomly (%5)
epsilon greedy, epsilon = 0.05
Some strategies may never be tried by the NN in some cases, (stuck by chance)

start e high and gradually decrease (we know more, possibility of being stuck is lower.)

Algorithm refinement: Mini-batch and soft updates



Limitations

much easier in simulations rather than real world scenarios.
fewer applications compared to (un)supervised learning.

potential for future.