REINFORCEMENT LEARNING ANDREW NG

Friday, December 9, 2022 9:03 PM

Resource:- https://www.coursera.org/learn/unsupervised-learning-recommenders-reinforcement-learning/home/week/3

Reinforcement Learning

position of helicopter → how to move control sticks

state s → action a

x → y

reward function

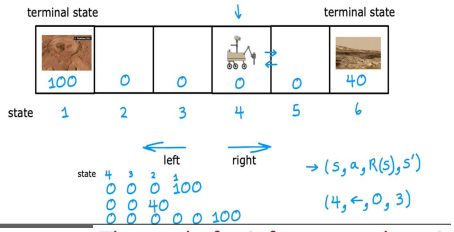
positive reward : helicopter flying well +1

negative reward : helicopter flying poorly −1000

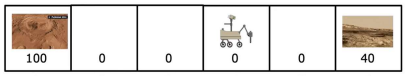
Robotic Dog Example

X, Y

Mars Rover Example



Return



Return = $0+(0.9)0+(0.9)^2 0+(0.9)^3 100 = 0.729 \times 100 = 72.9$

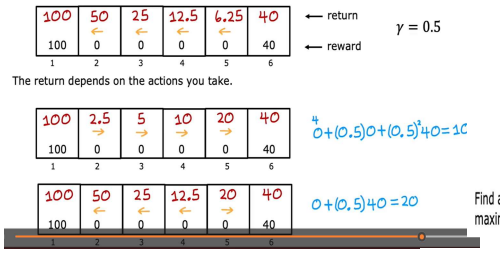
Return = $R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$ (until terminal state)

Discount Factor $\gamma = 0.9$ 0.99 0.999

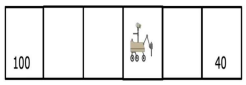
$\gamma = 0.5$

Return = $0+(0.5)0+(0.5)^2 0+(0.5)^3 100 = 12.5$

Example of Return

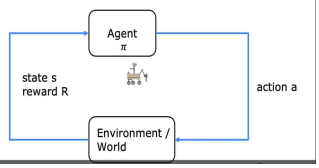


The goal of reinforcement learning



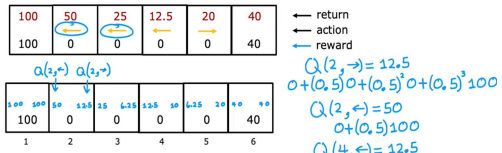
Find a policy π that tells you what action (a = $\pi(s)$) to take in every state (s) so as to maximize the return.

Markov Decision Process (MDP)

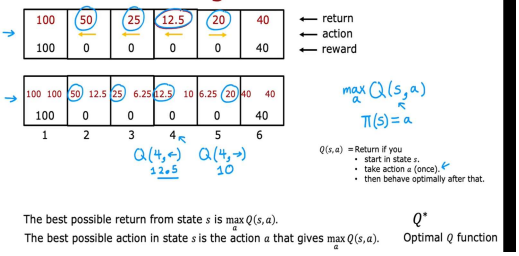


State action value function (Q-function)

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



Picking actions



The best possible return from state s is $\max_a Q(s,a)$.

The best possible action in state s is the action a that gives $\max_a Q(s,a)$.

Q^* Optimal Q function

Bellman Equation

$Q(s,a)$ = 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'

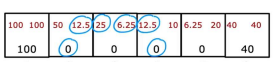
$R(s)$ = reward of current state

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

Bellman Equation

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

$Q(5,a) = R(5)$



s = 2, a = →, s' = 3

$Q(2,\rightarrow) = R(2) + 0.5 \max_{a'} Q(3,a')$
$= 0 + (0.5)25 = 12.5$

$Q(4,\leftarrow) = R(4) + 0.5 \max_{a'} Q(3,a')$
$= 0 + (0.5)25 = 12.5$

s = 4, a = ←, s' = 3

DeepLearning.AI Stanford ONLINE Andrew Ng

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

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

Reward you get right away

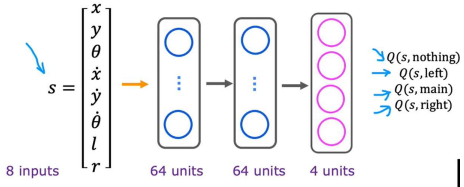
Return from behaving optimally starting from state s'.

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

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

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

Deep Reinforcement Learning



In a state s, input s to neural network.

Pick the action a that maximizes $Q(s,a)$. $R(s) + \gamma \max_{a'} Q(s',a')$

Learning Algorithm

Initialize neural network randomly as guess of $Q(s,a)$.

Repeat {

Take actions in the lunar lander. Get $(s,a,R(s),s')$.

Store 10,000 most recent $(s,a,R(s),s')$ tuples.

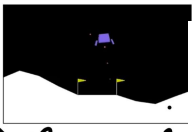
Train model:

Create training set of 10,000 examples usir g

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

Train Q_{new} such that $Q_{new}(s,a) \approx y$. $f_{w,b}(x) \approx y$

Set $Q = Q_{new}$.



Refinement

Greedy Policy

How to choose actions while still learning?

In some state s

Option 1:

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

Q(s,main) is low

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 ($\varepsilon = 0.05$)

095

Start ε high

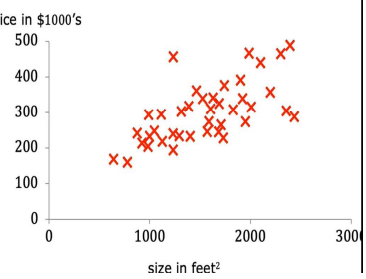
1.0 → 0.01

Gradually decrease

Mini-batch

x	y
2104	400
1416	232
1534	315
852	178
...	...
3210	870

mini-batch 1
mini-batch 2
mini-batch 3



Soft Update

Set $Q = Q_{new}$. ← $Q(s,a)$

W, B W_{new}, B_{new}

$W = 0.01 W_{new} + 0.99 W$

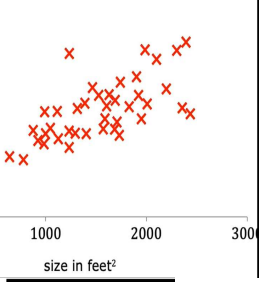
$B = 0.01 B_{new} + 0.99 B$

$w = 1 w_{new} + 0 w$

Mini-batch

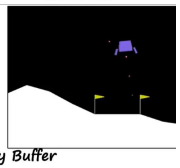
x	y
2104	400
1416	232
1534	315
852	178
...	...
3210	870

mini-batch 1
mini-batch 2
mini-batch 3



Learning Algorithm

Initialize neural network randomly as guess of $Q(s,a)$.
Repeat {
  Take actions in the lunar lander. Get $(s,a,R(s),s')$.
  Store 10,000 most recent $(s,a,R(s),s')$ tuples.



Replay Buffer

  Train model:
    Create training set of ~~10,000~~ 1,000 examples using
      $x = (s,a)$ and $y = R(s) + \gamma \max_{a'} Q(s',a')$.
    Train Q_{new} such that $Q_{new}(s,a) \approx y$.
  Set $Q = Q_{new}$.

$x^{(1)}, y^{(1)}$
$x^{(1000)}, y^{(1000)}$

Soft Update

Set $Q = Q_{new}$. ← $Q(s,a)$

W, B ← W_{new}, B_{new}

$W = 0.01\, W_{new} + 0.99\, W$
$B = 0.01\, B_{new} + 0.99\, B$

$w = 1 w_{new} + 0 w$