WEEK 3

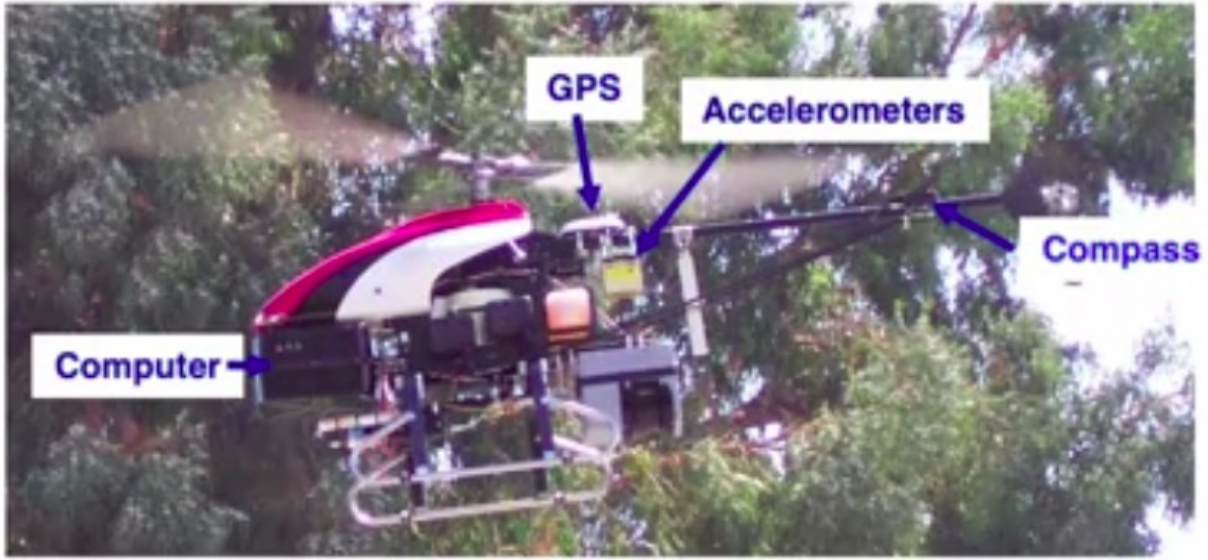
DeepLearning.AI

Stanford
ONLINE

Reinforcement Learning Introduction

What is Reinforcement Learning?

Autonomous Helicopter



How to fly it?

Autonomous Helicopter

[Thanks to Pieter Abbeel, Adam Coates and Morgan Quigley]

→ For more videos: http://heli.stanford.edu.

Reinforcement Learning

position of helicopter → how to move control sticks

state s → action a

x → y

X, Y

reward function

positive reward : helicopter flying well +1

negative reward : helicopter flying poorly −1000

Robotic Dog Example

[Thanks to Zico Kolter]

Robotic Dog Example

[Thanks to Zico Kolter]

Robotic Dog Example

[Thanks to Zico Kolter]

Robotic Dog Example

[Thanks to Zico Kolter]

Robotic Dog Example

[Thanks to Zico Kolter]

Robotic Dog Example

[Thanks to Zico Kolter]

Robotic Dog Example

[Thanks to Zico Kolter]

Applications

- Controlling robots
- Factory optimization
- Financial (stock) trading
- Playing games (including video games)

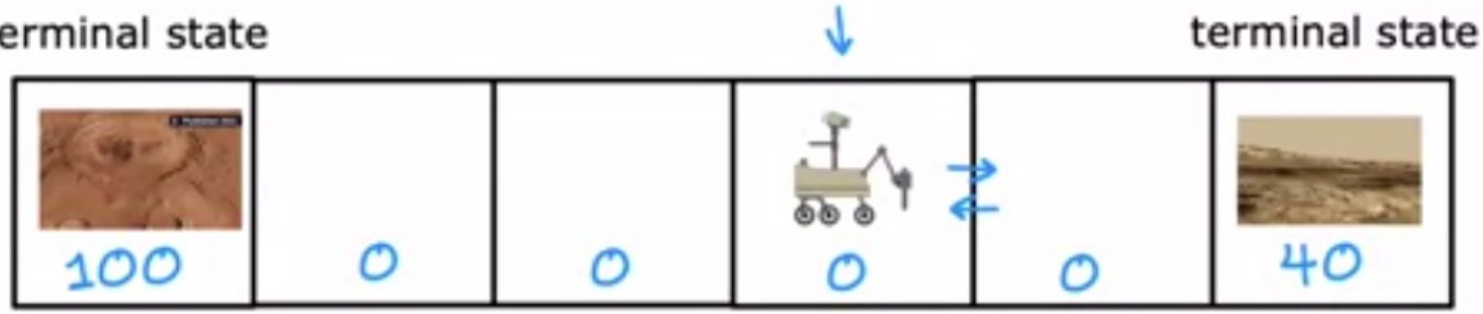
DeepLearning.AI

Stanford
ONLINE

Reinforcement Learning formalism

Mars rover example

Mars Rover Example



left right

state

4	3	2	1	
0	0	0	100	
0	0	40		
0	0	0	0	0 100

$(s, a, R(s), s')$

$(4, \leftarrow, 0, 3)$

[Credit: Jagriti Agrawal, Emma Brunskill]

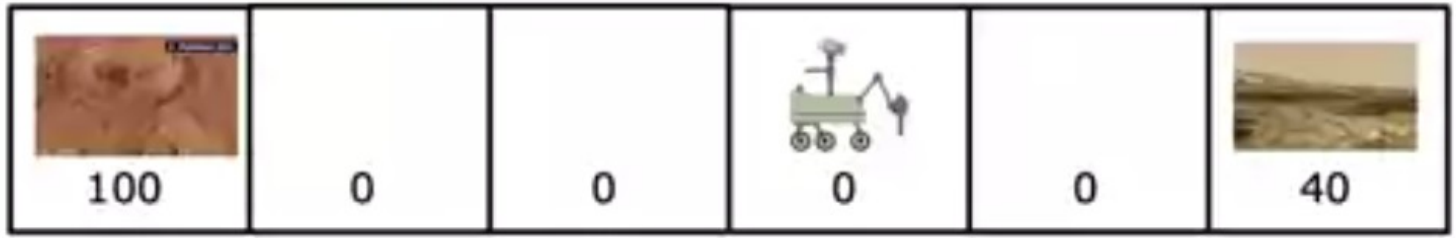
DeepLearning.AI

Stanford ONLINE

Reinforcement Learning formalism

The Return in reinforcement learning

Return



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

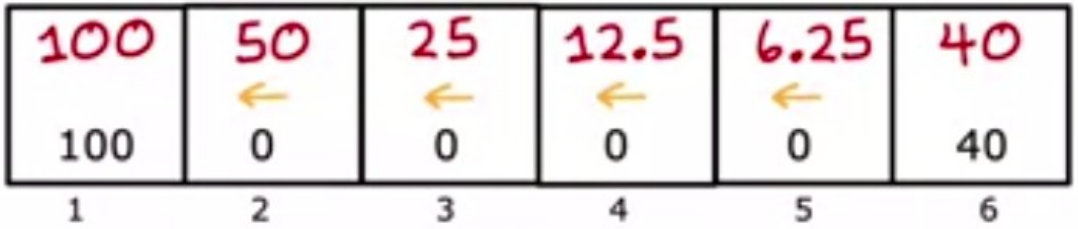
Return $= R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$ (until terminal state)

Discount Factor $\gamma = 0.9 \quad 0.99 \quad 0.999$

$\gamma = 0.5$

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

Example of Return

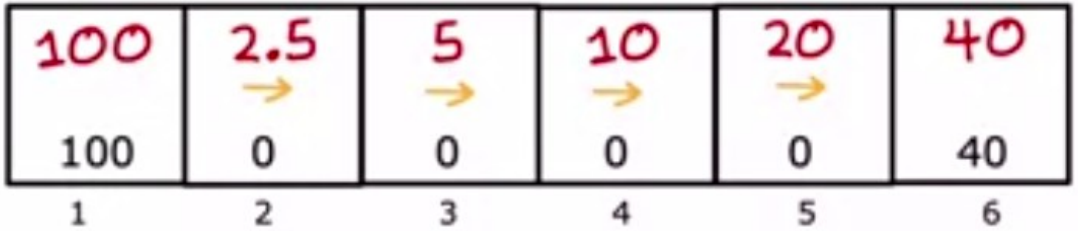


← return

← reward

$\gamma = 0.5$

The return depends on the actions you take.



$0 + (0.5)0 + (0.5)^2 40 = 10$

$0 + (0.5)40 = 20$

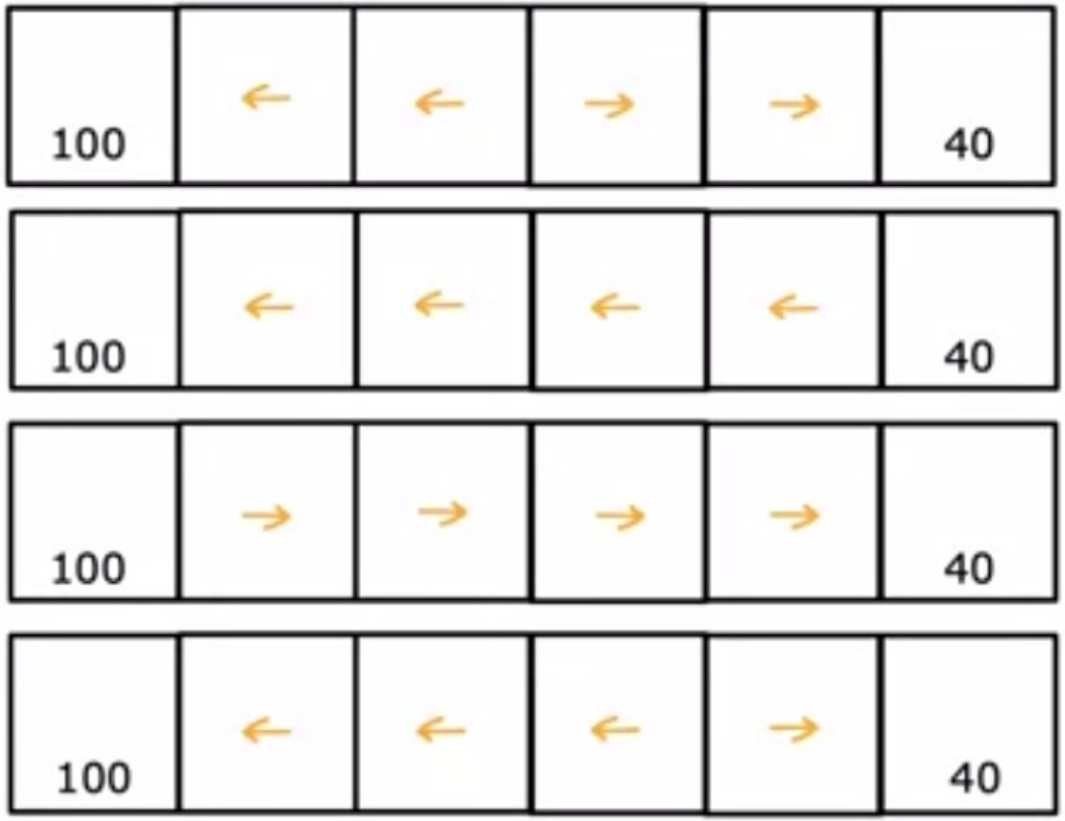
DeepLearning.AI

Stanford
ONLINE

Reinforcement Learning formalism

Making decisions: Policies in reinforcement learning

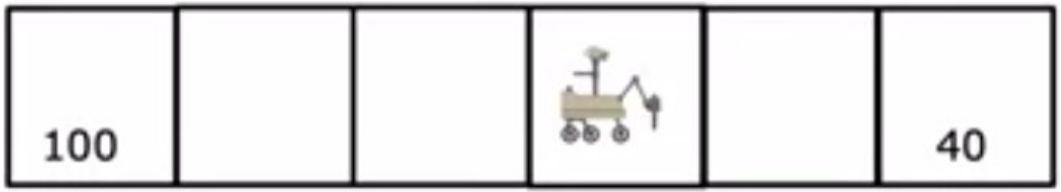
Policy



$\pi(5) = a$
$\pi(2) = \leftarrow$
$\pi(3) = \leftarrow$
$\pi(4) = \leftarrow$
$\pi(5) = \rightarrow$

A policy is a function $\pi(s) = a$ mapping from states to actions, that tells you what action *a* to take in a given state *s*.

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.

DeepLearning.AI

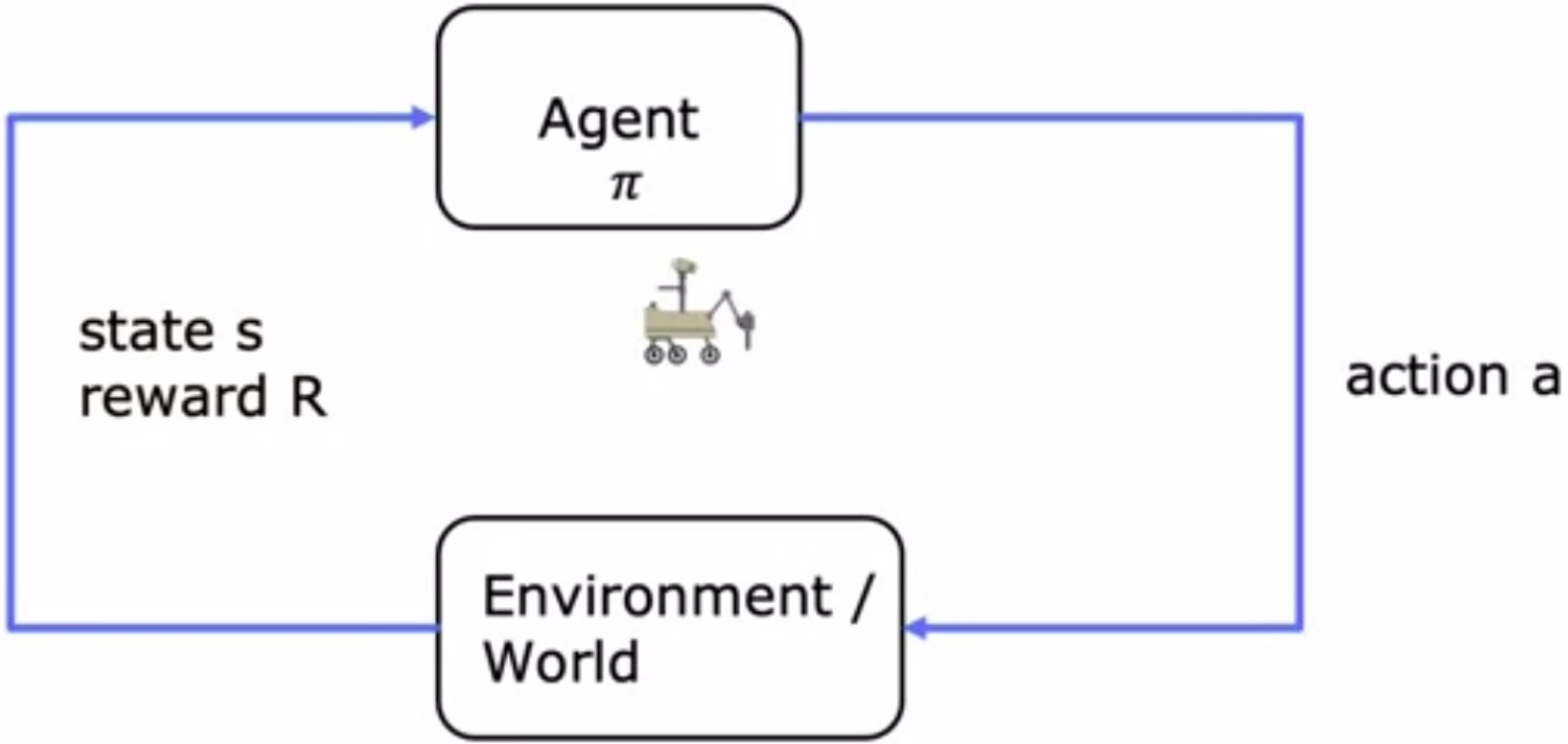
Stanford
ONLINE

Reinforcement Learning formalism

Review of key concepts

	Mars rover 	Helicopter 	Chess 
states	6 states	position of helicopter	pieces on board
actions	← →	how to move control stick	possible move
rewards	100, 0, 40	+1, −1000	+1, 0, −1
discount factor γ	0.5	0.99	0.995
return	$R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$	$R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$	$R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$
policy π		Find $\pi(s) = a$	Find $\pi(s) = a$

Markov Decision Process (MDP)



Congratulations! You passed!

Grade received 100% **Latest Submission Grade** 100% **To pass** 80% or higher

Go to next item

1. 1 / 1 point

You are using reinforcement learning to control a four legged robot. The position of the robot would be its _____.

- ○ action
- ○ reward
- ○ return
- ◉ state

Correct
Great!

2. 1 / 1 point

2. **1 / 1 point**

You are controlling a Mars rover. You will be very very happy if it gets to state 1 (significant scientific discovery), slightly happy if it gets to state 2 (small scientific discovery), and unhappy if it gets to state 3 (rover is permanently damaged). To reflect this, choose a reward function so that:

- ○ R(1) > R(2) > R(3), where R(1), R(2) and R(3) are positive.
- ○ R(1) > R(2) > R(3), where R(1), R(2) and R(3) are negative.
- ◉ R(1) > R(2) > R(3), where R(1) and R(2) are positive and R(3) is negative.
- ○ R(1) < R(2) < R(3), where R(1) and R(2) are negative and R(3) is positive.

Correct
Good job!

3. **1 / 1 point**

You are using reinforcement learning to fly a helicopter. Using a discount factor of 0.75, your helicopter starts in some state and receives rewards -100 on the first step, -100 on the second step, and 1000 on the third and final step (where it has reached a terminal state). What is the return?

Good job!

3. **1 / 1 point**

You are using reinforcement learning to fly a helicopter. Using a discount factor of 0.75, your helicopter starts in some state and receives rewards -100 on the first step, -100 on the second step, and 1000 on the third and final step (where it has reached a terminal state). What is the return?

- ○ -0.75*100 - 0.75^2*100 + 0.75^3*1000
- ◉ -100 - 0.75*100 + 0.75^2*1000
- ○ -100 - 0.25*100 + 0.25^2*1000
- ○ -0.25*100 - 0.25^2*100 + 0.25^3*1000

Correct
Awesome!

4. **1 / 1 point**

Given the rewards and actions below, compute the return from state 3 with a discount factor of $\gamma = 0.25$.

4. **1 / 1 point**

Given the rewards and actions below, compute the return from state 3 with a discount factor of $\gamma = 0.25$.

- 0.39
- 25
- **6.25**
- 0

Correct

If starting from state 3, the rewards are in states 3, 2, and 1. The return is $0 + (0.25) \times 0 + (0.25)^2 \times 100 = 6.25$.

DeepLearning.AI

Stanford
ONLINE

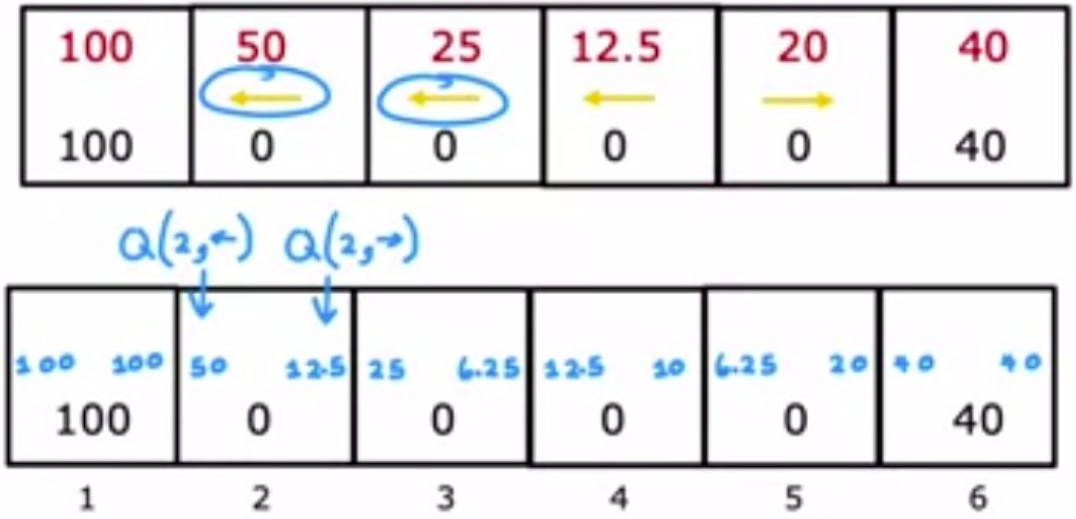
State-action value function

State-action value function definition

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.

$Q(s, a)$



← return
← action
← reward

$Q(2, \rightarrow) = 12.5$

$0 + (0.5)0 + (0.5)^2 0 + (0.5)^3 100$

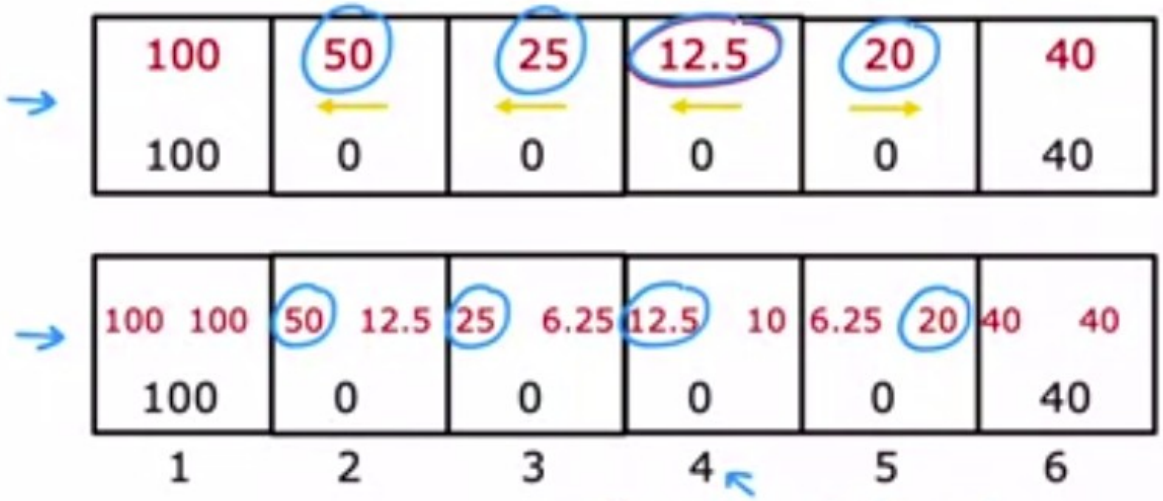
$Q(2, \leftarrow) = 50$

$0 + (0.5)100$

$Q(4, \leftarrow) = 12.5$

$0 + (0.5)0 + (0.5)^2 0 + (0.5)^3 100$

Picking actions



← return
← action
← reward

$\max_a Q(s,a)$

$\rightarrow \pi(s) = a$

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

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

DeepLearning.AI

Stanford
ONLINE

State-action value function

State-action value function example

In this Jupyter notebook, you can modify the mars rover example to see how the values of Q(s,a) will change depending on the rewards and discount factor changing.

In [1]:
```
import numpy as np
from utils import *
```

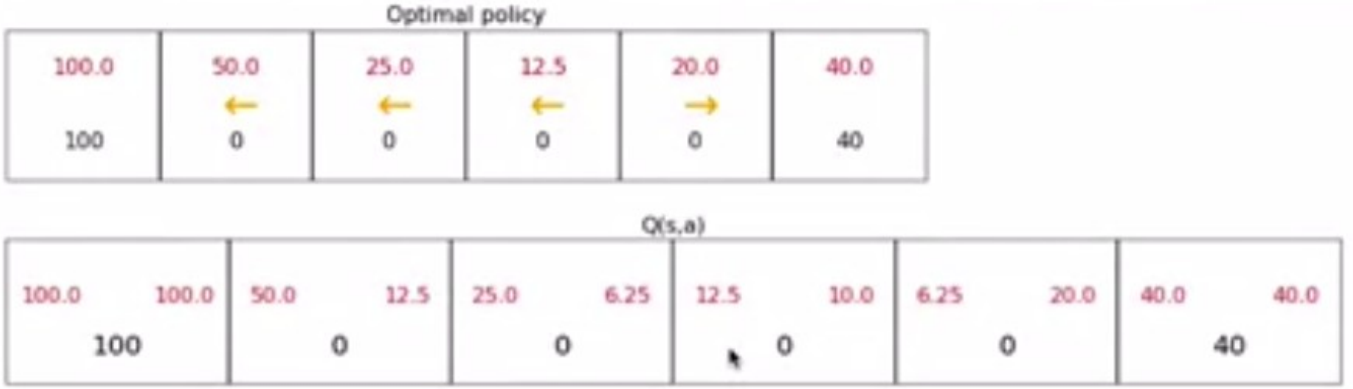
In [2]:
```
# Do not modify
num_states = 6
num_actions = 2
```

In [3]:
```
terminal_left_reward = 100
terminal_right_reward = 40
each_step_reward = 0

# Discount factor
gamma = 0.5

# Probability of going in the wrong direction
misstep_prob = 0
```

In [4]:
```
generate_visualization(terminal_left_reward, terminal_right_reward, each_step_reward, gamma, misstep_prob)
```



In [ ]:

State Action Value Function Example

In this Jupyter notebook, you can modify the mars rover example to see how the values of Q(s,a) will change depending on the rewards and discount factor changing.

In [1]:
```
import numpy as np
from utils import *
```

In [2]:
```
# Do not modify
num_states = 6
num_actions = 2
```

In [5]:
```
terminal_left_reward = 100
terminal_right_reward = 10
each_step_reward = 0

# Discount factor
gamma = 0.5

# Probability of going in the wrong direction
misstep_prob = 0
```

In [6]:
```
generate_visualization(terminal_left_reward, terminal_right_reward, each_step_reward, gamma, misstep_prob)
```

Optimal policy

100.0	50.0	25.0	12.5	6.25	10.0
	←	←	←	←	
100	0	0	0	0	10

Q(s,a)

100.0	100.0	50.0	12.5	25.0	6.25	12.5	3.12	6.25	5.0	10.0	10.0
100		0		0		0		0		10	

In [ ]:

State Action Value Function Example

In this Jupyter notebook, you can modify the mars rover example to see how the values of Q(s,a) will change depending on the rewards and discount factor changing.

In [1]:
```
import numpy as np
from utils import *
```

In [2]:
```
# Do not modify
num_states = 6
num_actions = 2
```

In [7]:
```
terminal_left_reward = 100
terminal_right_reward = 40
each_step_reward = 0

# Discount factor
gamma = 0.9

# Probability of going in the wrong direction
misstep_prob = 0
```

In [8]:
```
generate_visualization(terminal_left_reward, terminal_right_reward, each_step_reward, gamma, misstep_prob)
```

Optimal policy

100.0	90.0	81.0	72.9	65.61	40.0
100	← 0	← 0	← 0	← 0	40

Q(s,a)

100.0	100.0	90.0	72.9	81.0	65.61	72.9	59.05	65.61	36.0	40.0	40.0
100		0		0		0		0		40	

State Action Value Function Example

In this Jupyter notebook, you can modify the mars rover example to see how the values of Q(s,a) will change depending on the rewards and discount factor changing.

In [1]:
```
import numpy as np
from utils import *
```

In [2]:
```
# Do not modify
num_states = 6
num_actions = 2
```

In [9]:
```
terminal_left_reward = 100
terminal_right_reward = 40
each_step_reward = 0

# Discount factor
gamma = 0.3

# Probability of going in the wrong direction
misstep_prob = 0
```

In [10]:
```
generate_visualization(terminal_left_reward, terminal_right_reward, each_step_reward, gamma, misstep_prob)
```

Optimal policy

100.0	30.0	9.0	3.6	12.0	40.0
	←	←	→	→	
100	0	0	0	0	40

Q(s,a)

100.0	100.0	30.0	2.7	9.0	1.08	2.7	3.6	1.08	12.0	40.0	40.0
100		0		0		0		0		40	

In [ ]:

DeepLearning.AI

Stanford
ONLINE

State-action value function

Bellman Equation

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

$R(s)$ = reward of current state

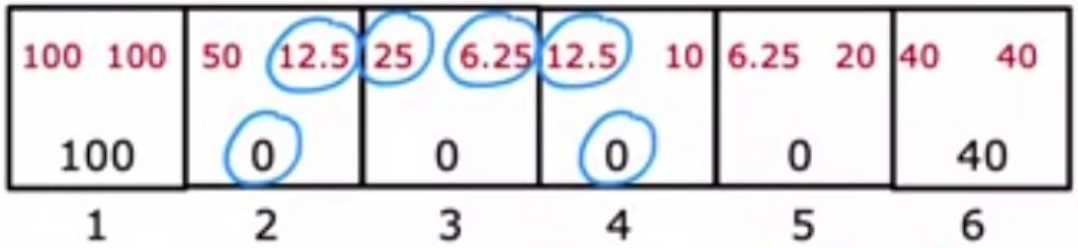
s' : state you get to after taking action a
a' : action that you take in state s'

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

Bellman Equation

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

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



$s = 2$
$a = \rightarrow$
$s' = 3$

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

$s = 4$
$a = \leftarrow$
$s' = 3$

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

Explanation of Bellman Equation

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

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

Explanation of Bellman Equation

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

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

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

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

Reward you get right away

Return from behaving optimally starting from state s'.

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

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

Explanation of Bellman Equation

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

100	100	50	12.5	25	6.25	12.5	10	6.25	20	40	40
100		0		0		0		0		40	
1		2		3		4		5		6	

$Q(4, \leftarrow)$

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

$= R(4) + (0.5)[0 + (0.5)0 + (0.5)^2 100]$

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

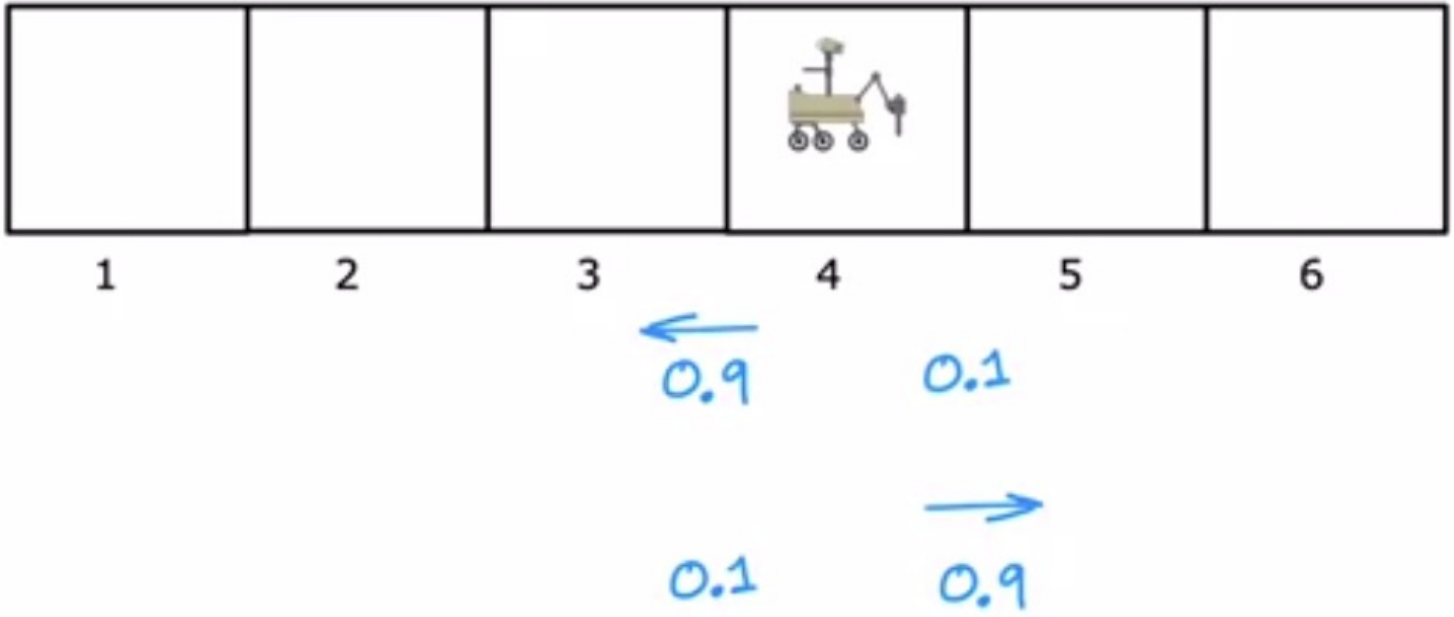
DeepLearning.AI

Stanford
ONLINE

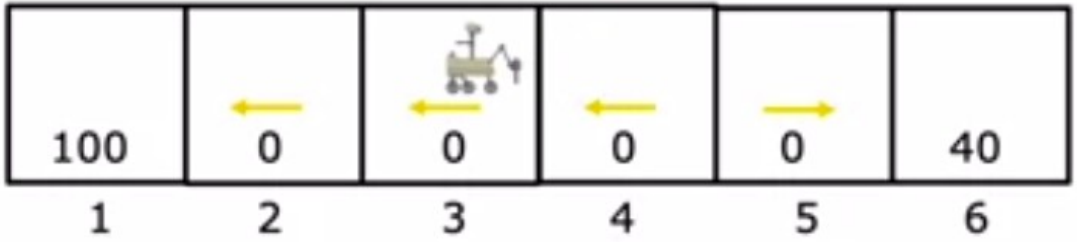
State-action value function

Random (stochastic) environment (Optional)

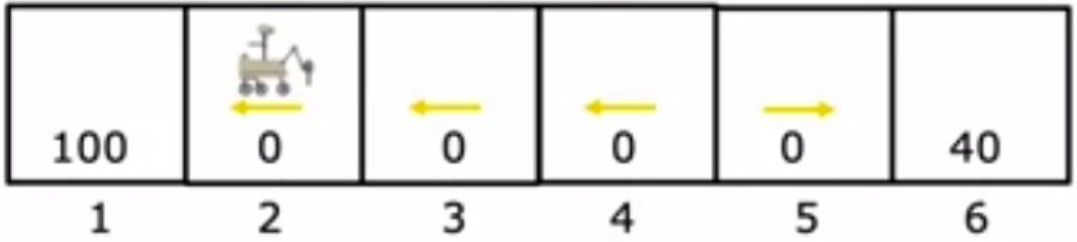
Stochastic Environment



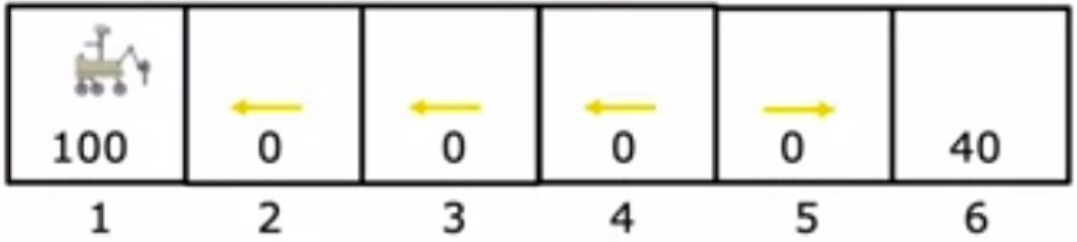
Expected Return



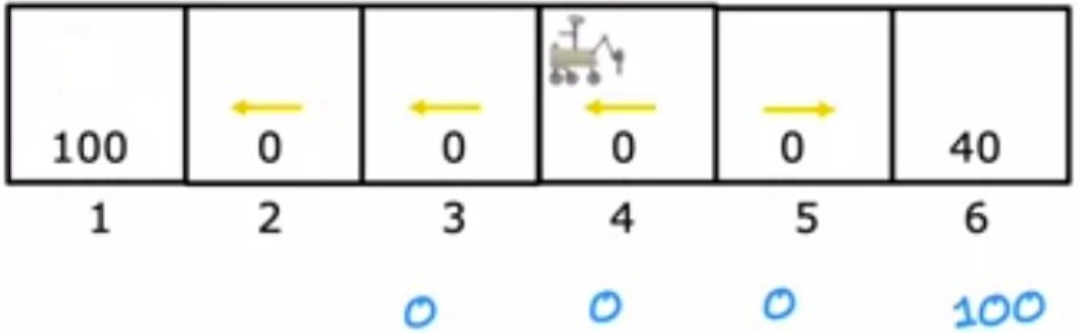
Expected Return



Expected Return



Expected Return

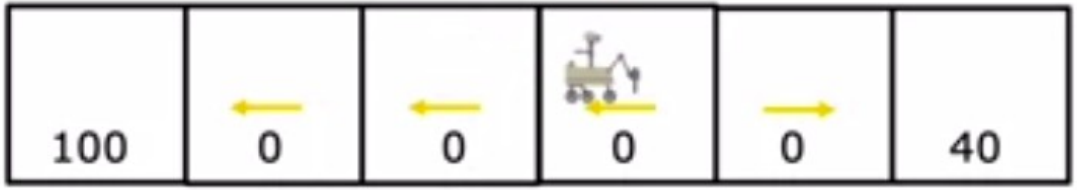


Expected Return

0 0 0 100

0 0

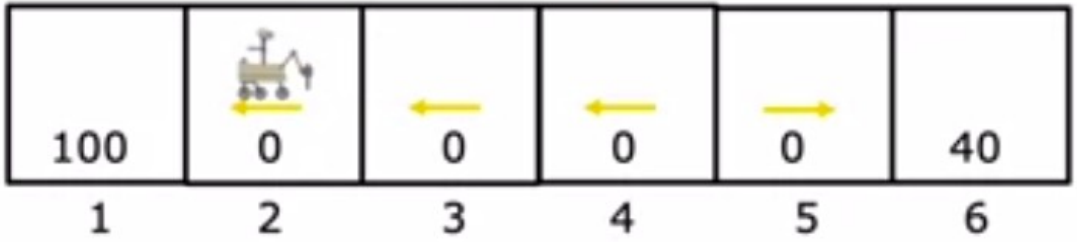
Expected Return



0 0 0 100

0 0

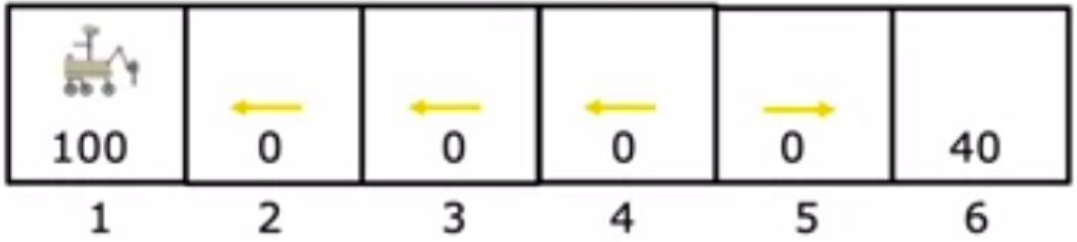
Expected Return



0 0 0 100

0 0 0

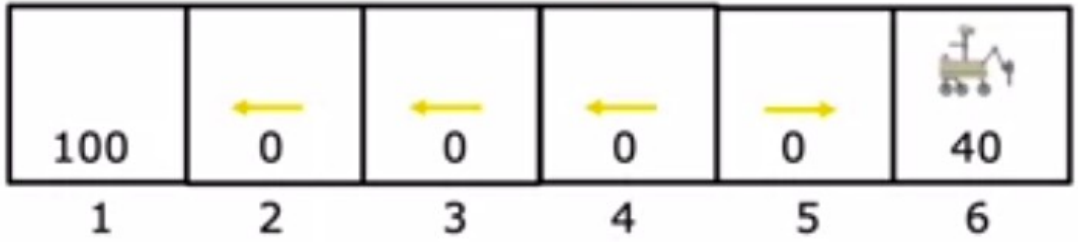
Expected Return



0 0 0 100

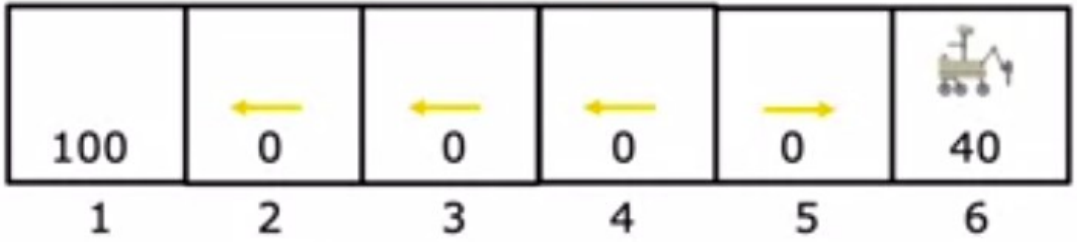
0 0 0 0 0 100

Expected Return



0 0 0 100
0 0 0 0 0 100
0 0 40

Expected Return



0 0 0 100
0 0 0 0 0 100
0 0 40

Expected Return = Average($R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \cdots$)

$= \mathrm{E}[R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \cdots]$

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:

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

3 ←

2 or 4

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

State Action Value Function Example

In this Jupyter notebook, you can modify the mars rover example to see how the values of Q(s,a) will change depending on the rewards and discount factor changing.

In [1]:
```
import numpy as np
from utils import *
```

In [2]:
```
# Do not modify
num_states = 6
num_actions = 2
```

In [3]:
```
terminal_left_reward = 100
terminal_right_reward = 40
each_step_reward = 0

# Discount factor
gamma = 0.5

# Probability of going in the wrong direction
misstep_prob = 0.1
```

In [4]:
```
generate_visualization(terminal_left_reward, terminal_right_reward, each_step_reward, gamma, misstep_prob)
```

Optimal policy

100.0	46.06	21.25	10.49	18.52	40.0
100	← 0	← 0	← 0	→ 0	40

Q(s,a)

100.0	100.0	46.06	14.56	21.25	7.02	10.49	9.4	6.72	18.52	40.0	40.0
100		0		0		0		0		40	

In [ ]:

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In this Jupyter notebook, you can modify the mars rover example to see how the values of Q(s,a) will change depending on the rewards and discount factor changing.

In [1]:
```
import numpy as np
from utils import *
```

In [2]:
```
# Do not modify
num_states = 6
num_actions = 2
```

In [5]:
```
terminal_left_reward = 100
terminal_right_reward = 40
each_step_reward = 0

# Discount factor
gamma = 0.5

# Probability of going in the wrong direction
misstep_prob = 0.4
```

In [6]:
```
generate_visualization(terminal_left_reward, terminal_right_reward, each_step_reward, gamma, misstep_prob)
```

Optimal policy

100.0	32.18	10.88	6.15	13.23	40.0
	←	←	→	→	
100	0	0	0	0	40

Q(s,a)

100.0	100.0	32.18	23.26	10.88	8.28	5.91	6.15	9.84	13.23	40.0	40.0
100		0		0		0		0		40	

In [ ]:

Congratulations! You passed!

Grade received 100% **Latest Submission Grade** 100% **To pass** 80% or higher

Go to next item

1. 1 / 1 point

Which of the following accurately describes the state-action value function $Q(s, a)$?

- (●) It is the return if you start from state s, take action a (once), then behave optimally after that.
- ( ) It is the return if you start from state s and repeatedly take action a.
- ( ) It is the return if you start from state s and behave optimally.
- ( ) It is the immediate reward if you start from state s and take action a (once).

Correct
Great!

2. 1 / 1 point

2. **1 / 1 point**

You are controlling a robot that has 3 actions: ← (left), → (right) and STOP. From a given state s, you have computed Q(s, ←) = -10, Q(s, →) = -20, Q(s, STOP) = 0.

What is the optimal action to take in state s?

- (●) STOP
- ( ) ← (left)
- ( ) → (right)
- ( ) Impossible to tell

Correct
Yes, because this has the greatest value.

3. **1 / 1 point**

For this problem, $\gamma = 0.25$. The diagram below shows the return and the optimal action from each state. Please compute Q(5, ←).

Back

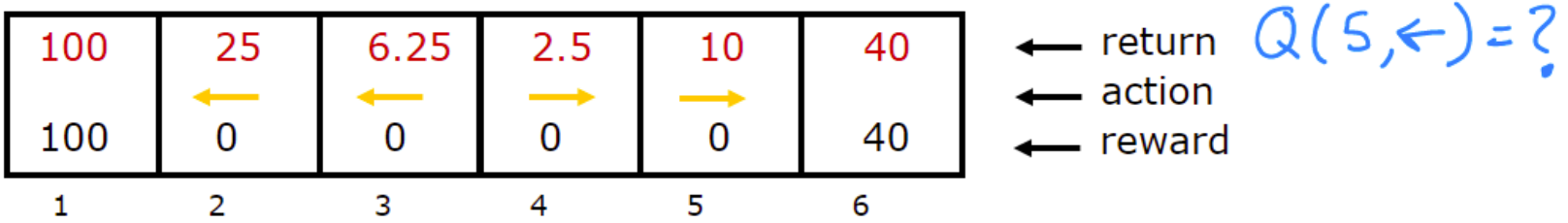
State-action value function

Graded Quiz • 30 min

Due Aug 13, 11:59 PM IST

3. **1 / 1 point**

For this problem, $\gamma = 0.25$. The diagram below shows the return and the optimal action from each state. Please compute Q(5, ←).



- (●) 0.625
- ( ) 0.391
- ( ) 1.25
- ( ) 2.5

Correct

Yes, we get 0 reward in state 5. Then $0 * 0.25$ discounted reward in state 4, since we moved left for our action. Now we behave optimally starting from state 4 onwards. So, we move right to state 5 from state 4 and receive $0 * 0.25^2$ discounted reward. Finally, we move right in state 5 to state 6 to receive a discounted reward of $40 * 0.25^3$. Adding these together we get 0.625.

DeepLearning.AI

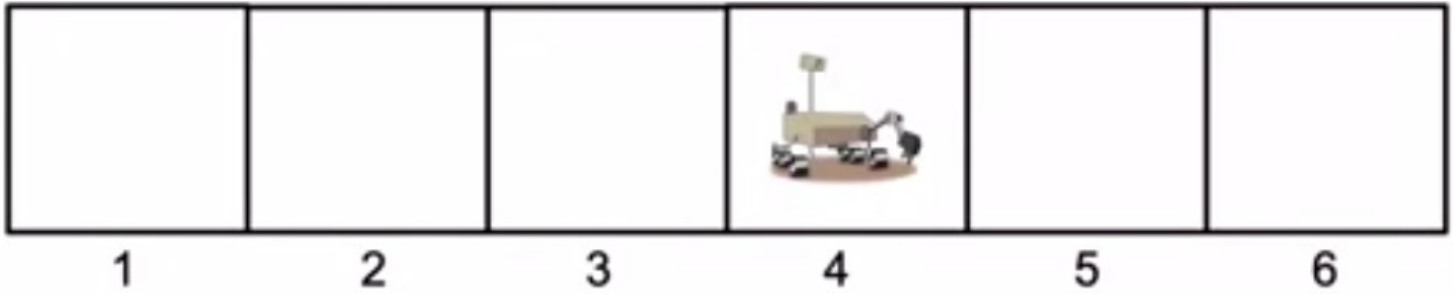
Stanford ONLINE

Continuous State Spaces

Example of continuous state applications

Discrete vs Continuous State

Discrete State:

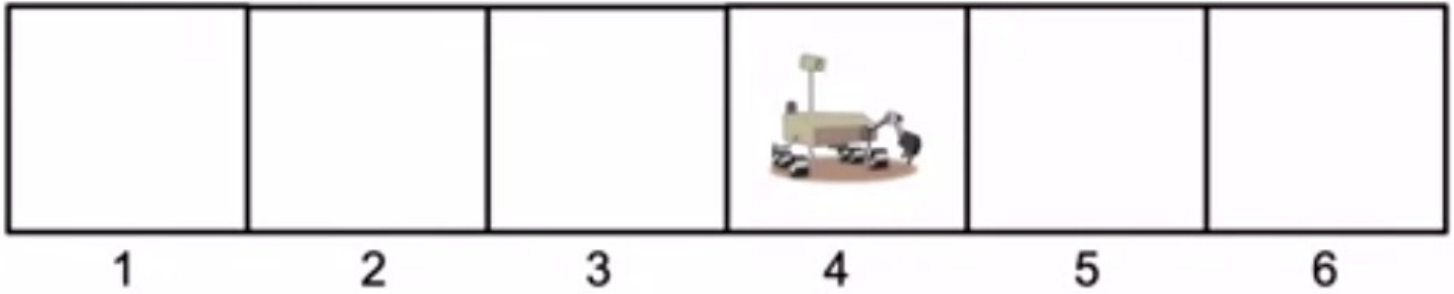


Continuous State:


Discrete vs Continuous State
Discrete State:
1
2
3
4
5
6
Continuous State:

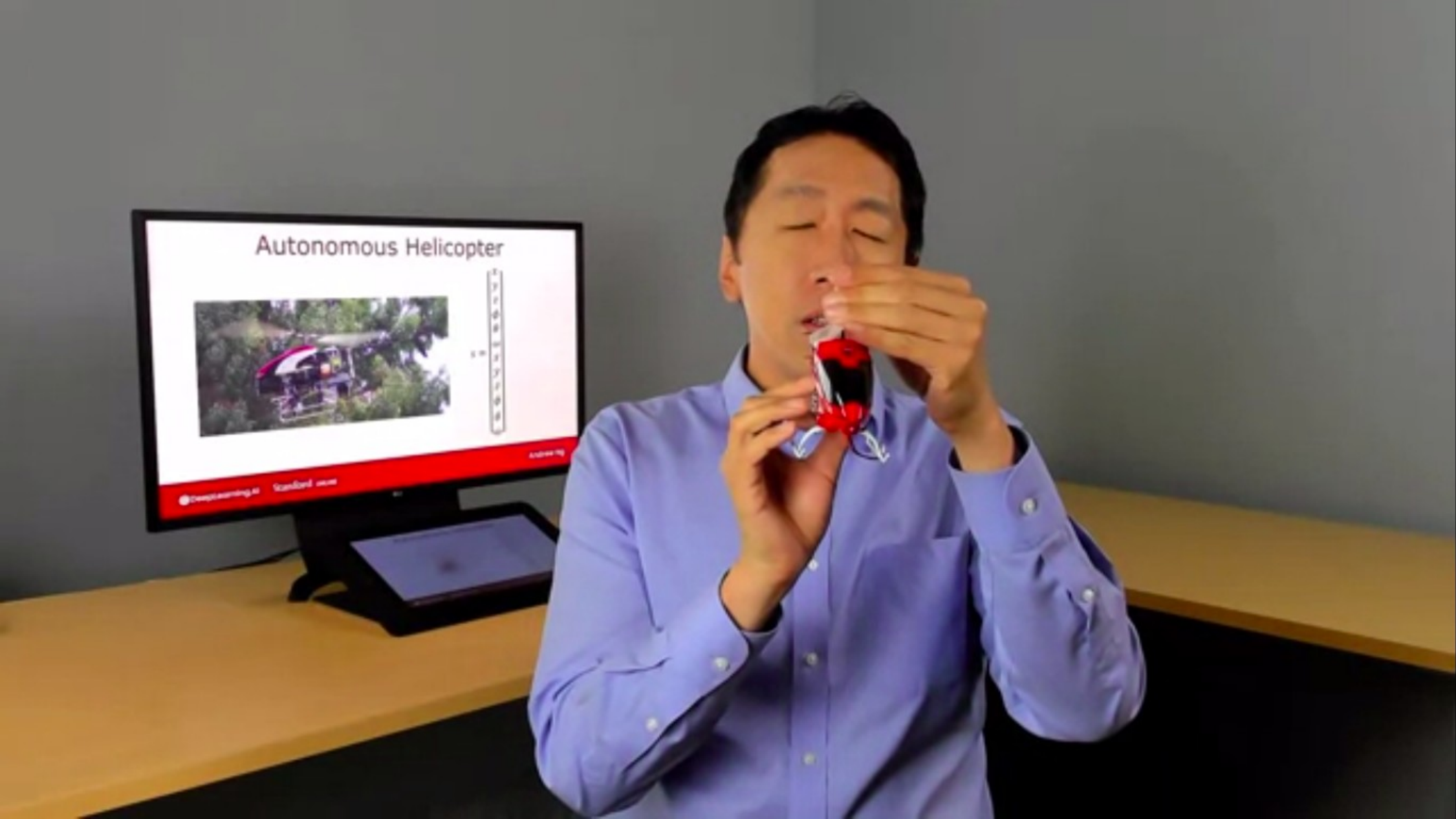
Discrete vs Continuous State

Discrete State:



Continuous State:

$$s = \begin{bmatrix} x \\ y \\ \theta \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$$


Autonomous Helicopter

Autonomous Helicopter

$$s = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \omega \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\omega} \end{bmatrix}$$

DeepLearning.AI

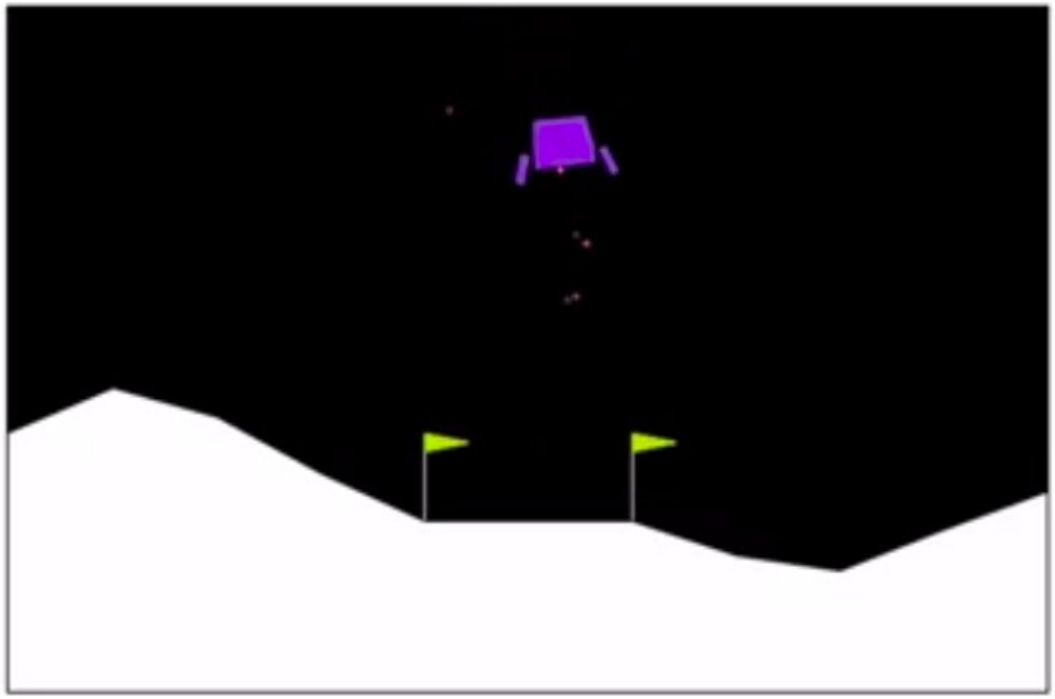
Stanford ONLINE

Continuous State Spaces

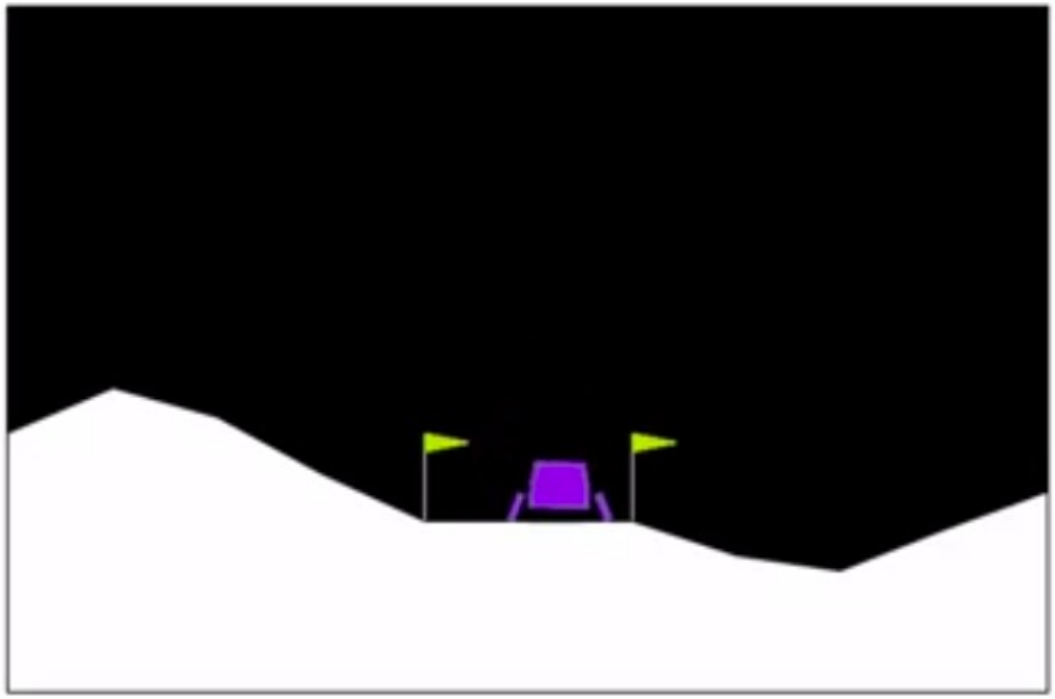
Lunar Lander

Lunar Lander

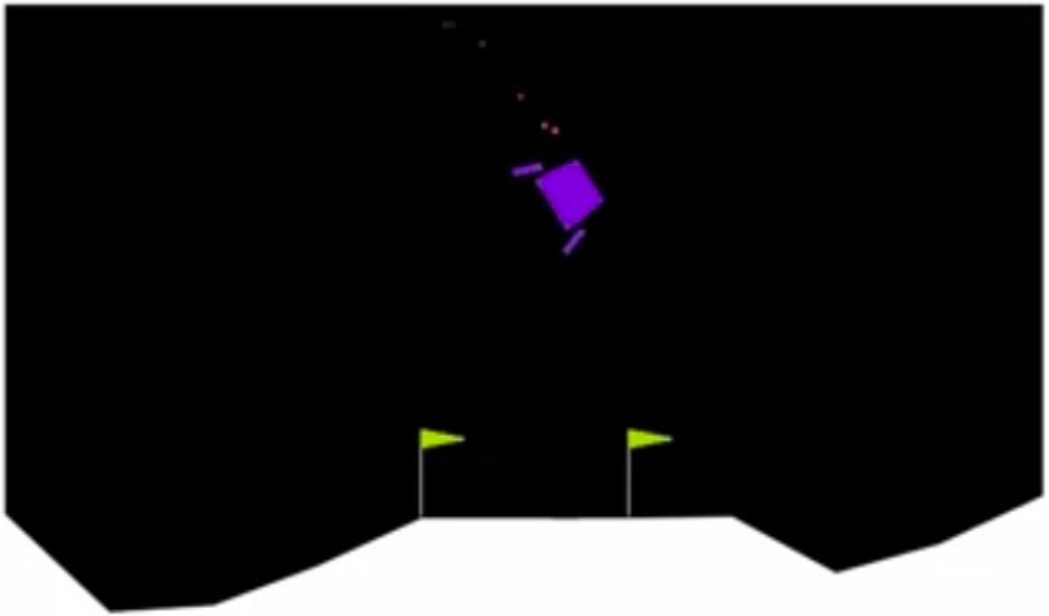
Lunar Lander



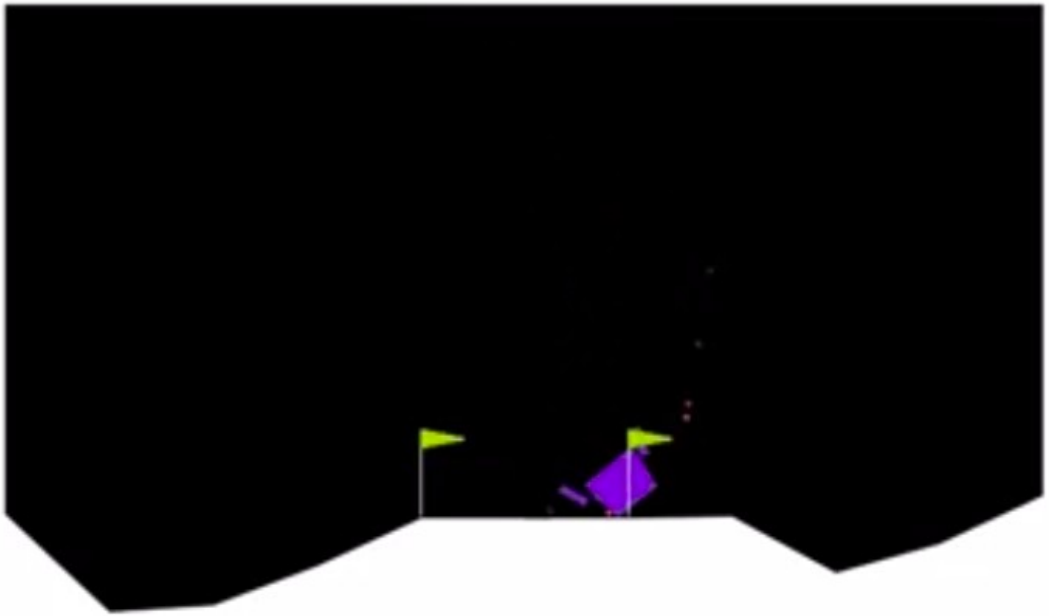
Lunar Lander



Lunar Lander



Lunar Lander



Lunar Lander

actions:

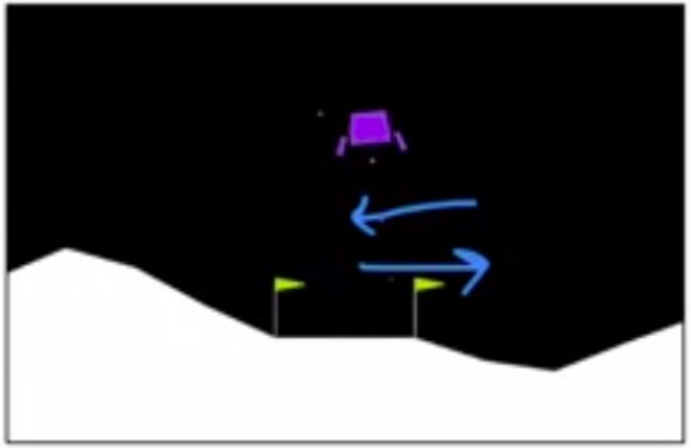
- do nothing
- left thruster
- main thruster
- right thruster

$$s = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ \theta \\ \dot{\theta} \\ l \\ r \end{bmatrix}$$

l, r: 0 or 1

Reward Function

- Getting to landing pad: 100 – 140
- Additional reward for moving toward/away from pad.
- Crash: -100
- Soft landing: +100
- Leg grounded: +10
- Fire main engine: -0.3
- Fire side thruster: -0.03



Lunar Lander Problem

Learn a policy π that, given

$$s = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ \theta \\ \dot{\theta} \\ l \\ r \end{bmatrix}$$

picks action $a = \pi(s)$ so as to maximize the return.

$$\gamma = 0.985$$

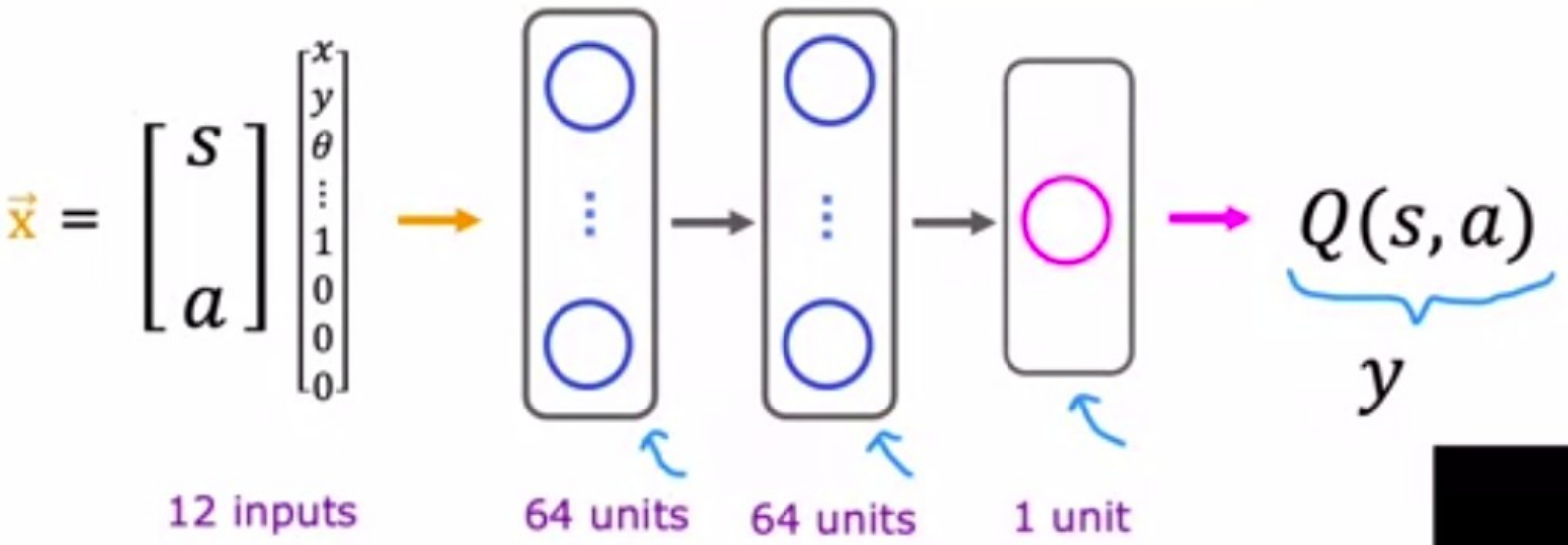
DeepLearning.AI

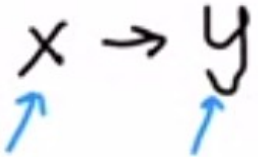
Stanford ONLINE

Continuous State Spaces

Learning the state-value function

Deep Reinforcement Learning

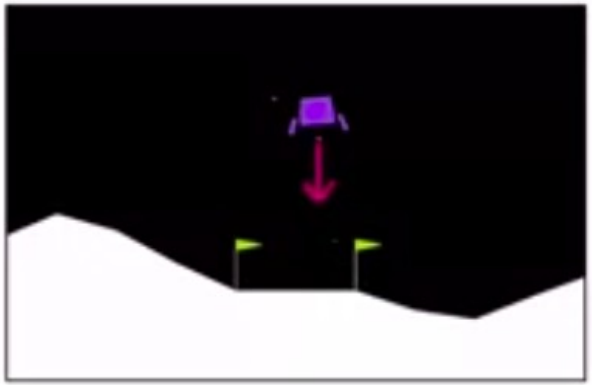




In a state s, use neural network to compute

$Q(s, \text{nothing})$, $Q(s, \text{left})$, $Q(s, \text{main})$ $Q(s, \text{right})$

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



Bellman Equation

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

x $\quad$ y

$$f_{w,B}(x) \approx y$$

$$y^{(1)} = R(s^{(1)}) + \gamma \max_{a'} Q(s'^{(1)}, a')$$

$$y^{(2)} = R(s^{(2)}) + \gamma \max_{a'} Q(s'^{(2)}, a')$$

$(s, a, R(s), s')$

$(s^{(1)}, a^{(1)}, R(s^{(1)}), s'^{(1)})$ ←

$(s^{(2)}, a^{(2)}, R(s^{(2)}), s'^{(2)})$ ←

$(s^{(3)}, a^{(3)}, R(s^{(3)}), s'^{(3)})$ ←

x	y
$x^{(1)} = (s^{(1)}, a^{(1)})$	$y^{(1)}$
$x^{(2)} = (s^{(2)}, a^{(2)})$	$y^{(2)}$
$x^{(10,000)}$	$y^{(10,000)}$

8 4

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 neural network:

Create training set of 10,000 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 $Q = Q_{new}$.

$f_{w,B}(x) \approx y$

x, y

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

$\vdots$

$x^{(10000)}, y^{(10000)}$

DeepLearning.AI

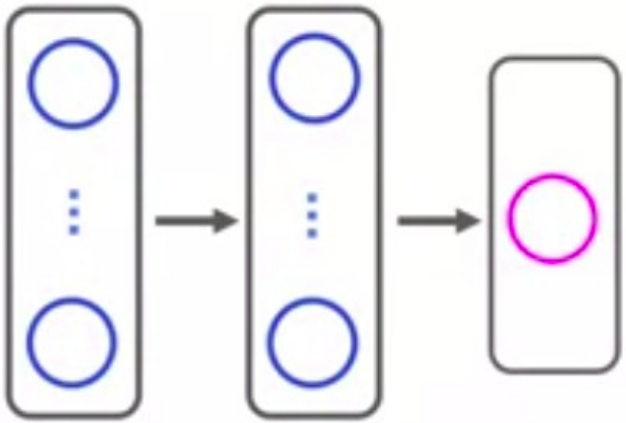
Stanford ONLINE

Continuous State Spaces

Algorithm refinement: Improved neural network architecture

Deep Reinforcement Learning

$$\vec{x} = \begin{bmatrix} s \\ a \end{bmatrix} \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ \theta \\ \dot{\theta} \\ l \\ r \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



$Q(s,a)$

12 inputs

64 units

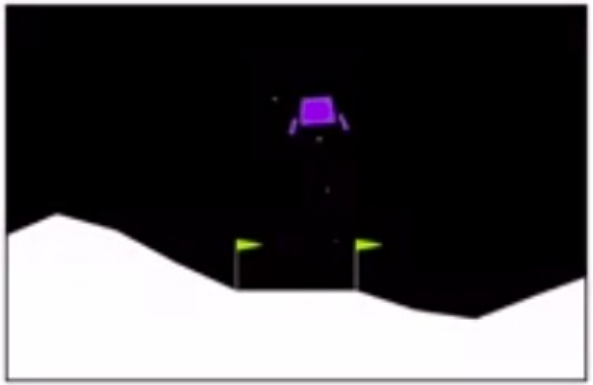
64 units

1 unit

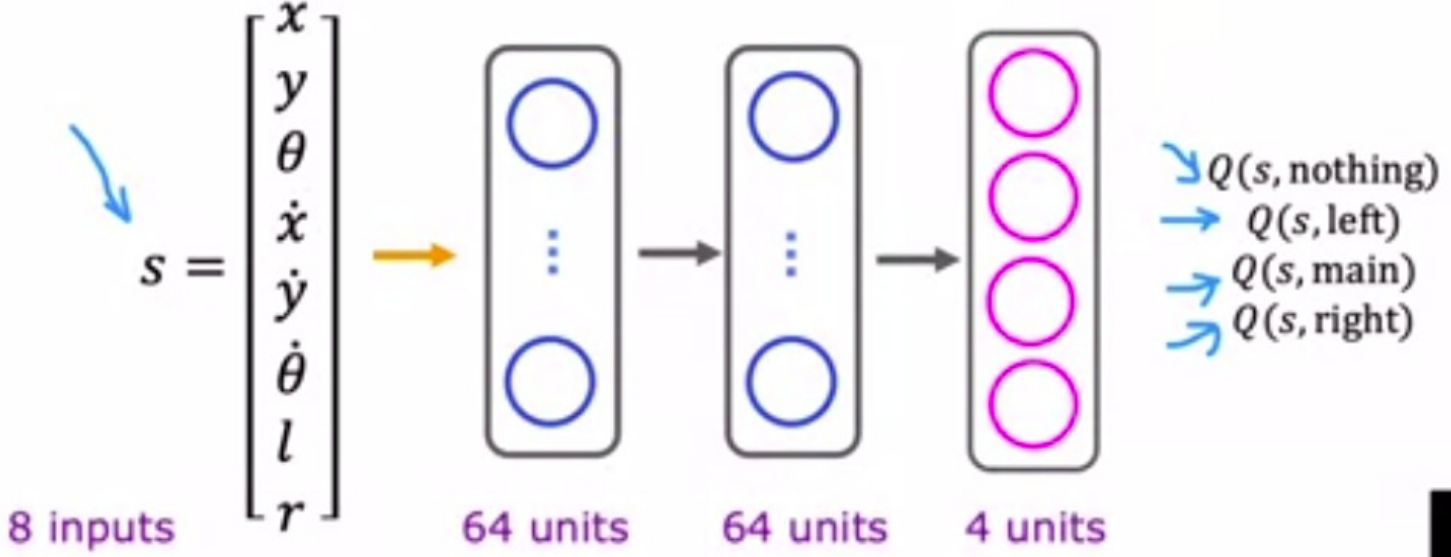
In a state s, use neural network to compute

$Q(s,\text{nothing})$, $Q(s,\text{left})$, $Q(s,\text{main})$, $Q(s,\text{right})$

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

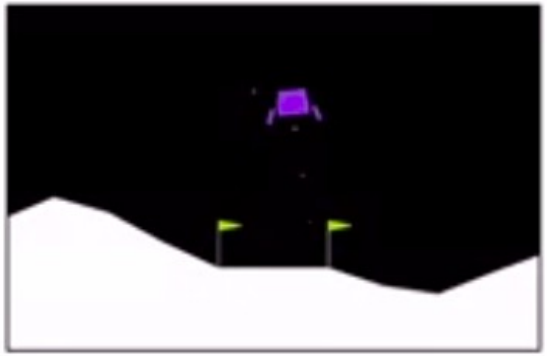


Deep Reinforcement Learning



In a state s, input s to neural network.

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



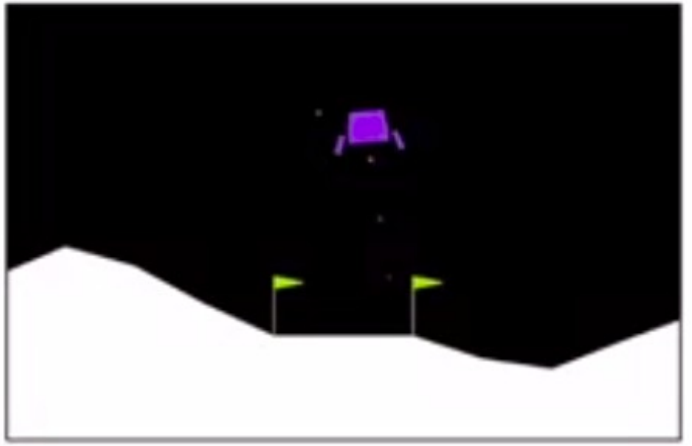
DeepLearning.AI

Stanford ONLINE

Continuous State Spaces

Algorithm refinement: ε-greedy policy

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 using

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

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

Set $Q = Q_{new}$.

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

↑

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

Start ε high

1.0 → 0.01

Gradually decrease

DeepLearning.AI

Stanford ONLINE

Continuous State Spaces

Algorithm refinement: Mini-batch and soft update (optional)

How to choose actions while still learning?

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

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2$$

$m = 100{,}000{,}000$

repeat {

$$w = w - \alpha \frac{\partial}{\partial w} \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2$$

$$b = b - \alpha \frac{\partial}{\partial b} \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2$$

}

How to choose actions while still learning?

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

100,000,000

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2$$

m = 100,000,000

m' = 1,000

repeat {

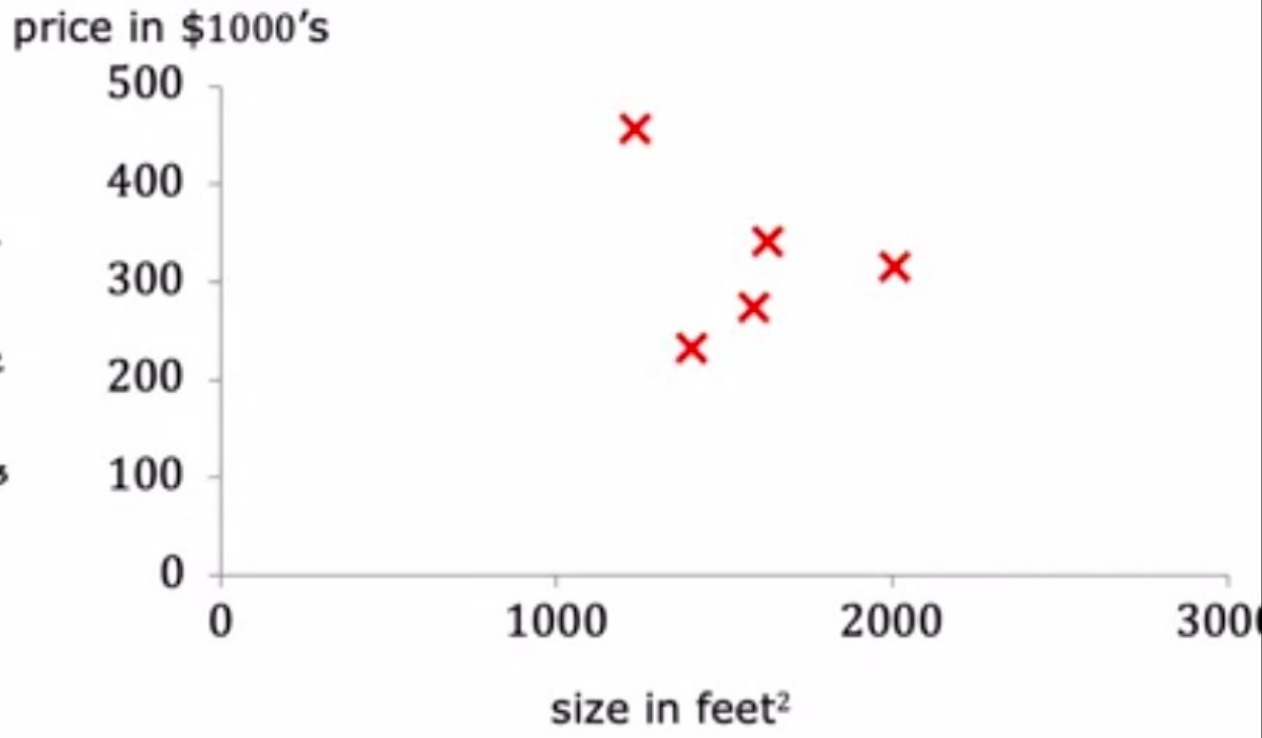
$$w = w - \alpha\frac{\partial}{\partial w}\frac{1}{2m'}\sum_{i=1}^{m'}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2$$

$$b = b - \alpha\frac{\partial}{\partial b}\frac{1}{2m'}\sum_{i=1}^{m'}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2$$

}

Mini-batch

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



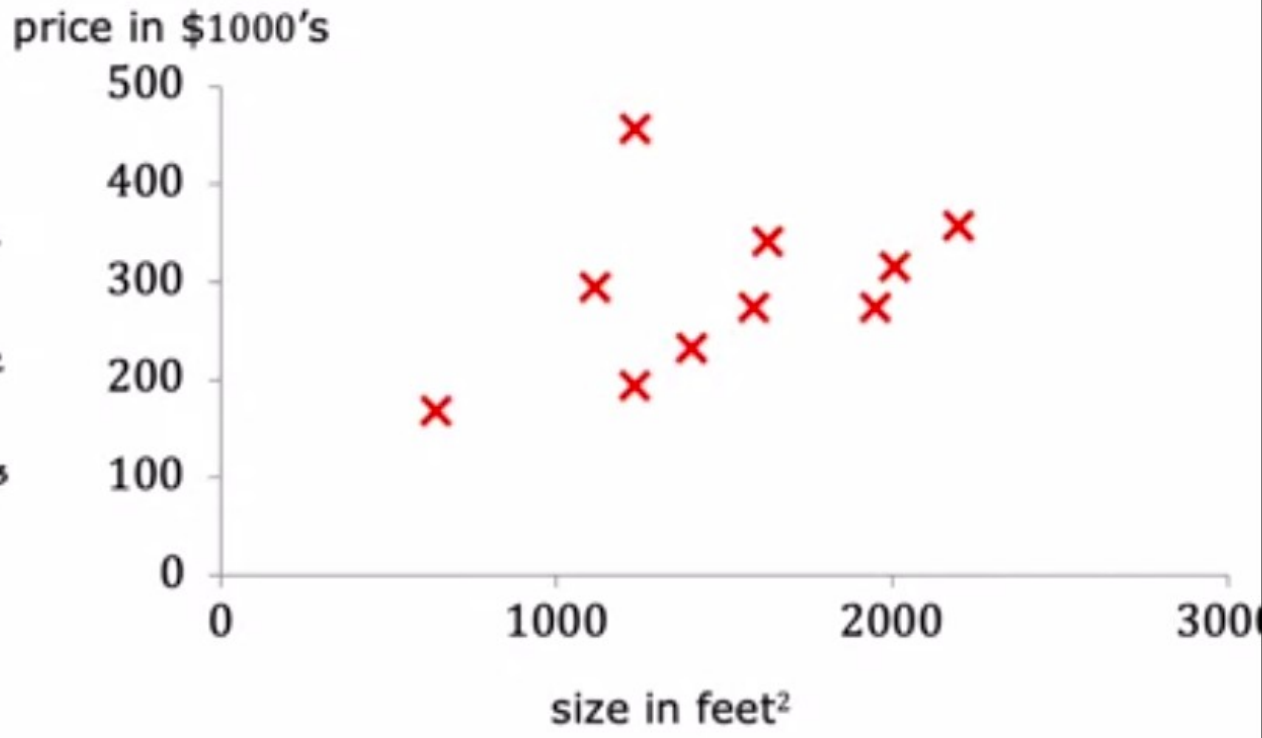
Mini-batch

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

mini-batch 1

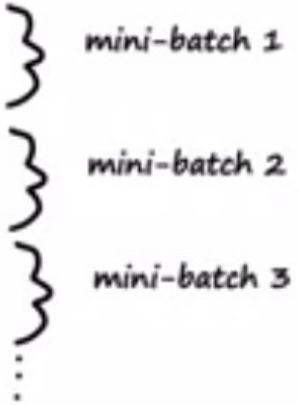
mini-batch 2

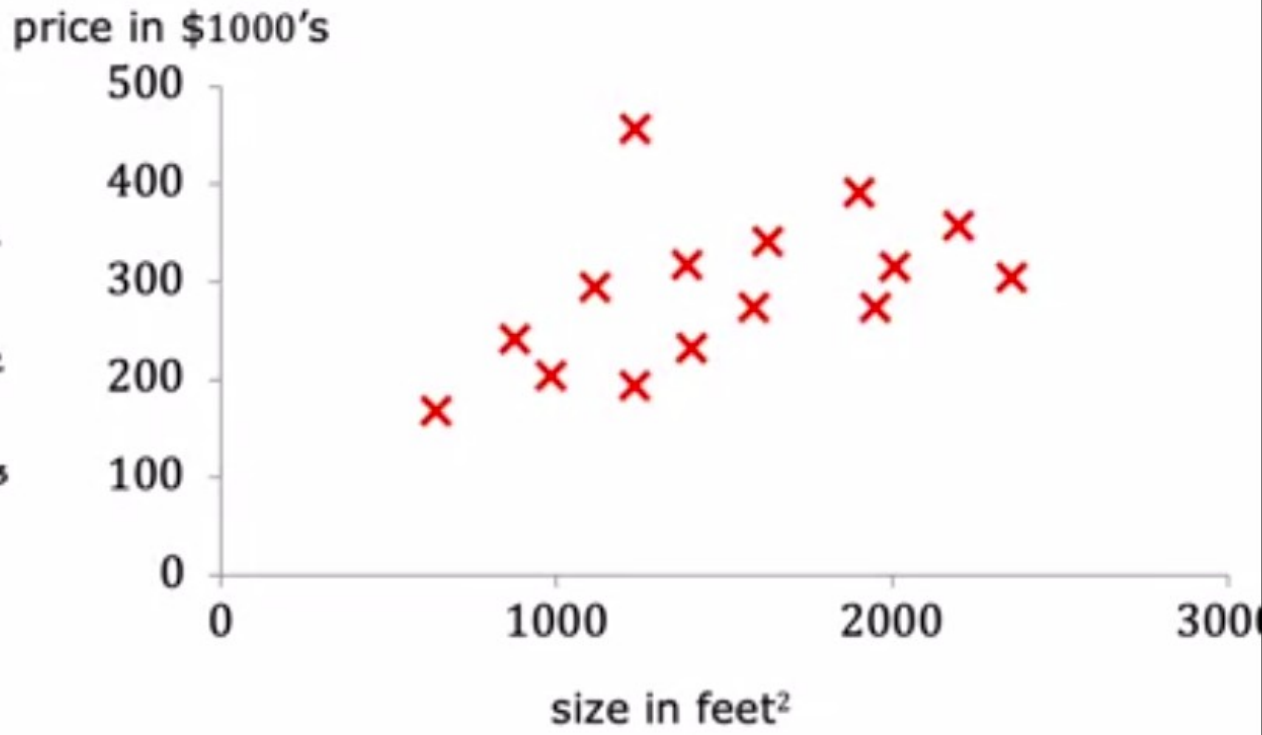
mini-batch 3



Mini-batch

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





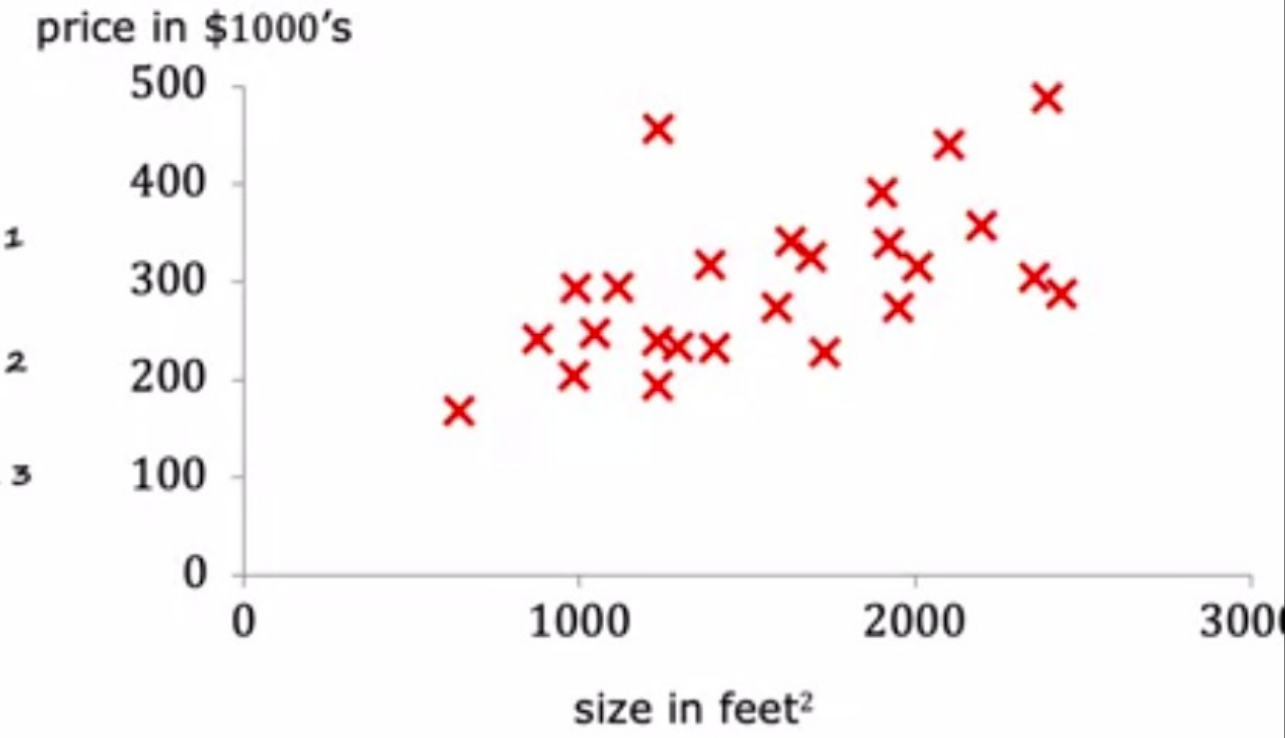
Mini-batch

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

mini-batch 1

mini-batch 2

mini-batch 3



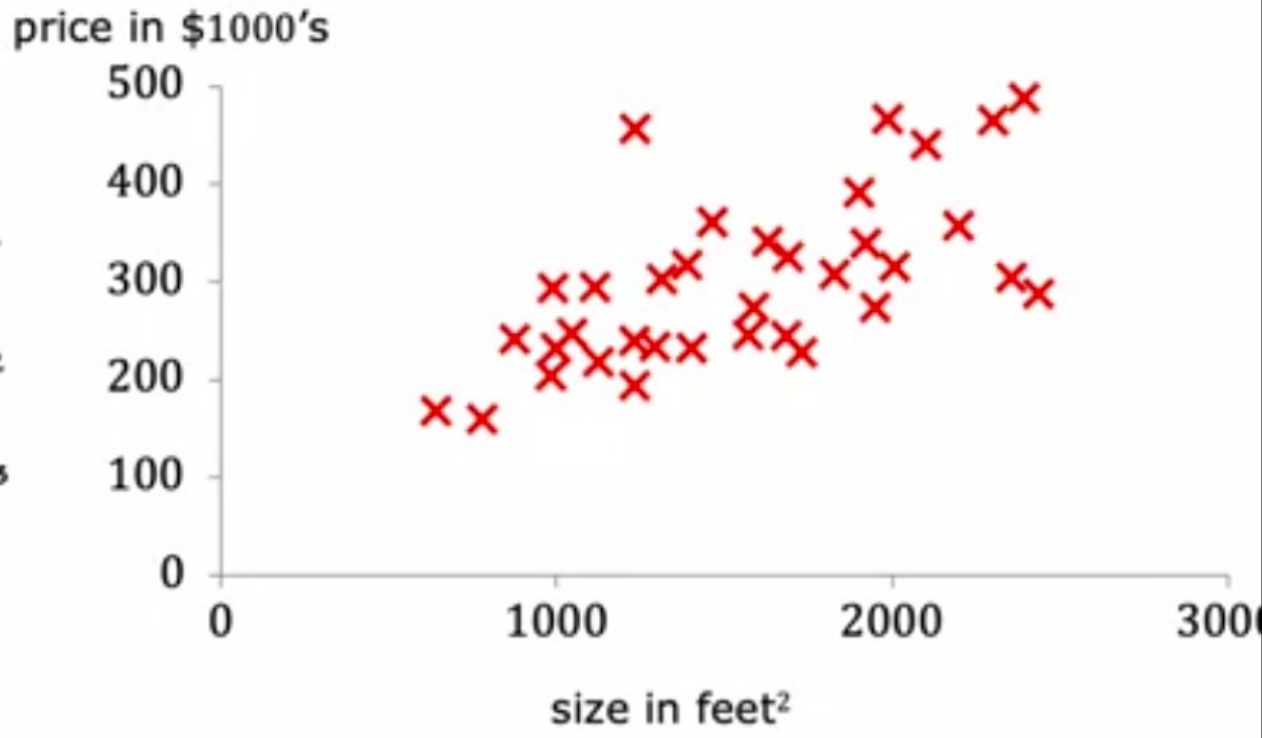
Mini-batch

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

mini-batch 1

mini-batch 2

mini-batch 3



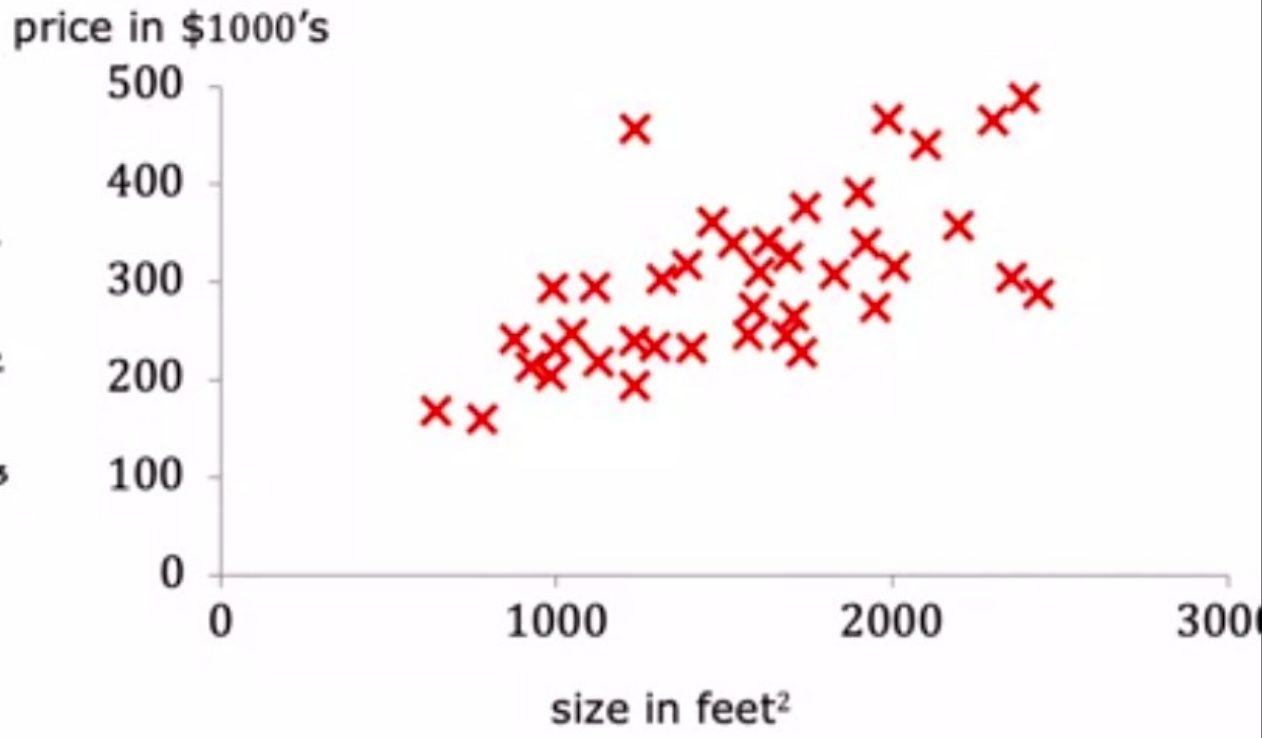
Mini-batch

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

mini-batch 1

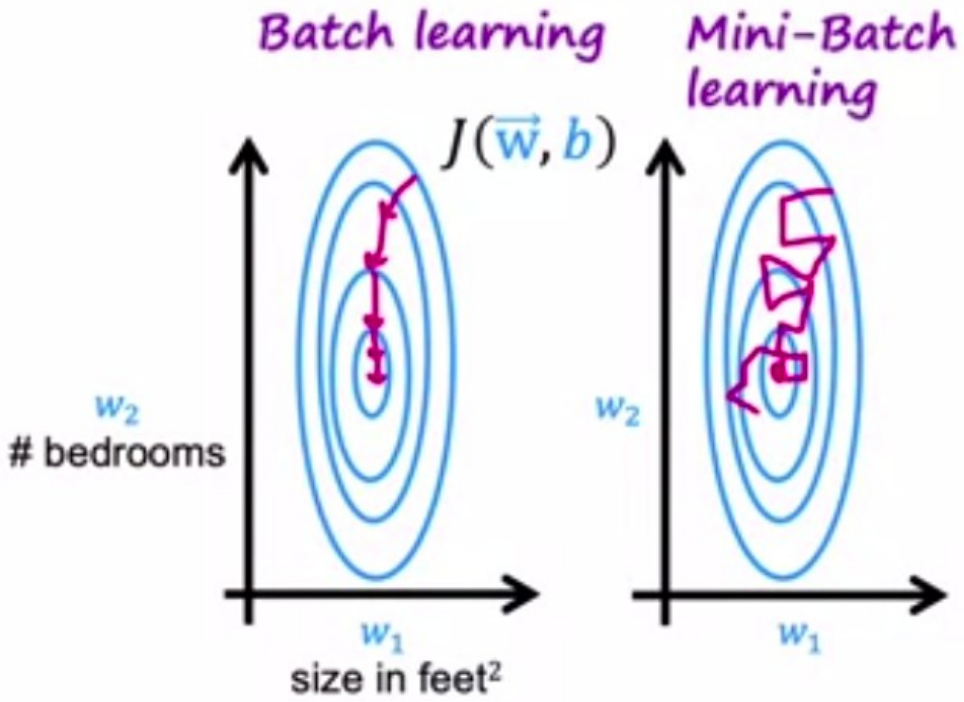
mini-batch 2

mini-batch 3

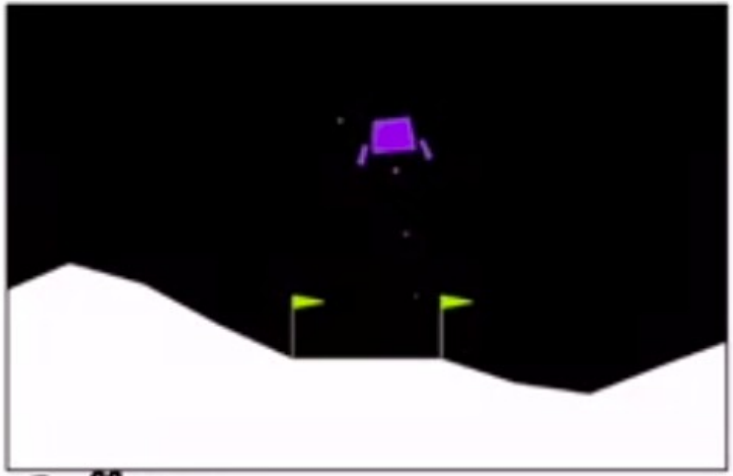


Mini-batch

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



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:

1,000

Create training set of ~~10,000~~ 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 $Q = Q_{new}$.

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

$\vdots$

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

Soft Update

Set $Q = Q_{new}$.

$Q(s,a)$

W, B $\quad$ W_{new}, B_{new}

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

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

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

DeepLearning.AI

Stanford
ONLINE

Continuous State Spaces

The state 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.

Congratulations! You passed!

Grade received 100% **Latest Submission Grade** 100% **To pass** 80% or higher

Go to next item

1. **1 / 1 point**

The Lunar Lander is a continuous state Markov Decision Process (MDP) because:

- ○ The state-action value $Q(s, a)$ function outputs continuous valued numbers
- ○ The state has multiple numbers rather than only a single number (such as position in the x-direction)
- ◉ The state contains numbers such as position and velocity that are continuous valued.
- ○ The reward contains numbers that are continuous valued

Correct
That's right!

That's right!

2. **1 / 1 point**

In the learning algorithm described in the videos, we repeatedly create an artificial training set to which we apply supervised learning where the input $x = (s, a)$ and the target, constructed using Bellman's equations, is y = _____?

- (●) $y = R(s) + \gamma \max_{a'} Q(s', a')$ where s' is the state you get to after taking action a in state s
- ( ) $y = R(s')$ where s' is the state you get to after taking action a in state s
- ( ) $y = \max_{a'} Q(s', a')$ where s' is the state you get to after taking action a in state s
- ( ) $y = R(s)$

Correct

3. **1 / 1 point**

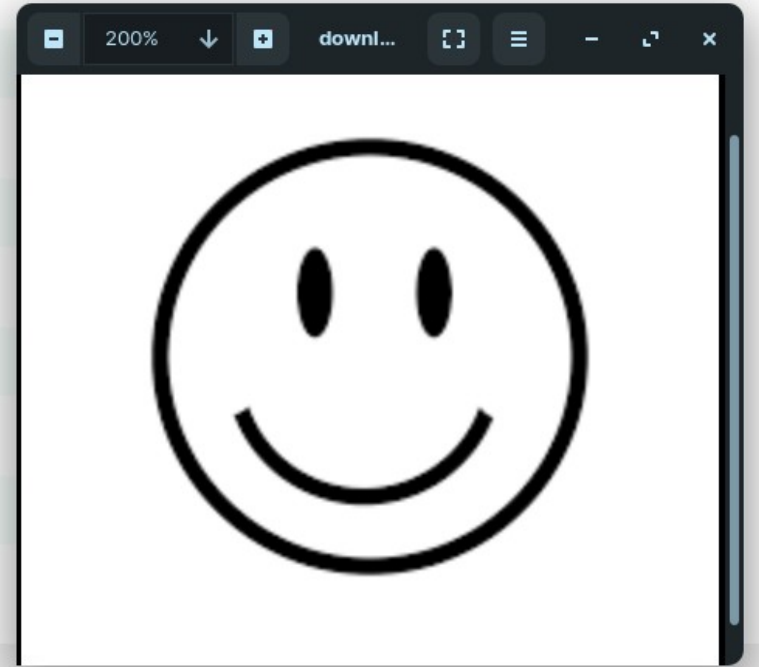
You have reached the final practice quiz of this class! What does that mean? (Please check all the answers, because all of them are correct!)

- [x] Andrew sends his heartfelt congratulations to you!

3. **1 / 1 point**

You have reached the final practice quiz of this class! What does that mean? (Please check all the answers, because all of them are correct!)

- [x] Andrew sends his heartfelt congratulations to you!
  - **Correct**
- [x] The DeepLearning.AI and Stanford Online teams would like to give you a round of applause!
  - **Correct**
- [x] What an accomplishment -- you made it!
  - **Correct**
- [x] You deserve to celebrate!
  - **Correct**



DeepLearning.AI

Stanford
ONLINE

Conclusion

Summary and
Thank you

Courses

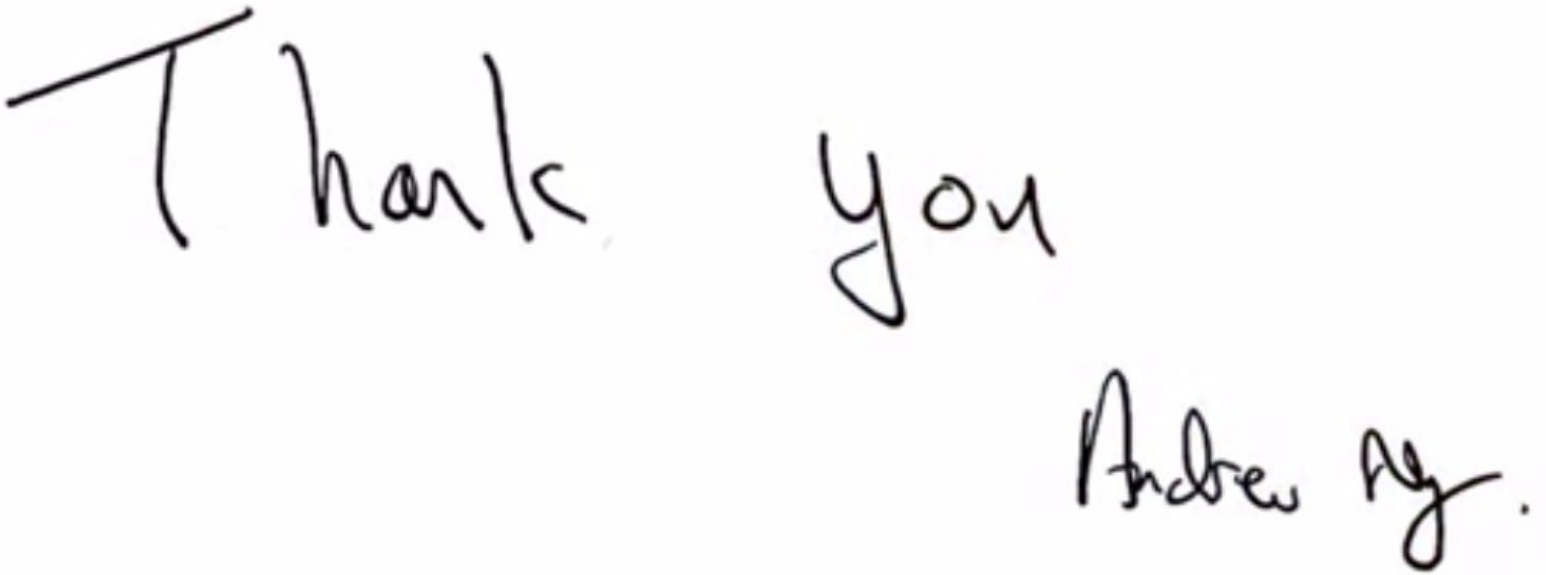
- Supervised Machine Learning: Regression and Classification

  Linear regression, logistic regression, gradient descent
- Advanced Learning Algorithms

  Neural networks, decision trees, advice for ML
- Unsupervised Learning, Recommenders, Reinforcement Learning

  Clustering, anomaly detection, collaborative filtering, content-based filtering, reinforcement learning


Courses
• Supervised Machine Learning: Regression and Classification
Linear regression, logistic regression, gradient descent
• Advanced Learning Algorithms
Neural networks, decision trees, advice for ML
• Unsupervised Learning, Recommenders, Reinforcement Learning
Clustering, anomaly detection, collaborative filtering, content-based filtering, reinforcement learning


Thank you
Andrew Ng.