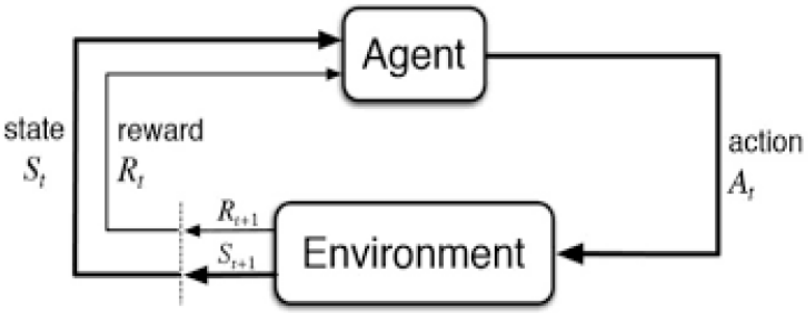
Deep Learning

Supervised
* FFNN
* CNN
* RNN, LSTM
* Forward propagation
* Error / Loss Function
* Gradient Descent
* Back propagation
* overfitting
* Normalizing

Unsupervised
* Autoencoders
* GAN

Reinforcement Learning
* Deep Q Nets
* Bellman's equation
* Q-Learning Algorithm
* State, action learning

RL Trading Bot



Agent : Bot (RL Algorithm)

Actions : Buy, Sell, Hold

Environment : Stock market

State : # stocks you have
stock price
Account Balance

Reward : Profit / Loss

Value : maximize the profit

example: current state → 10000, 2, 2000

2000 + 2000 = 22000

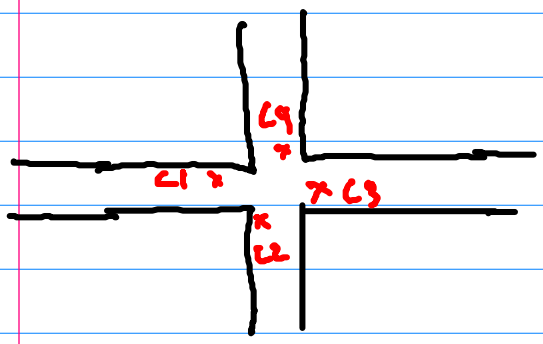
Buy 500 stocks ← defined by Random

new state → 10500, 2.01, 1000 ← 2000 − (2 × 500)

21105 + 1000 = 22105

Profit 105 (+ Reward)

RL for Traffic Light control



Agent : RL Algorithm

Actions : Lt_1, Lt_2, Lt_3, Lt_4

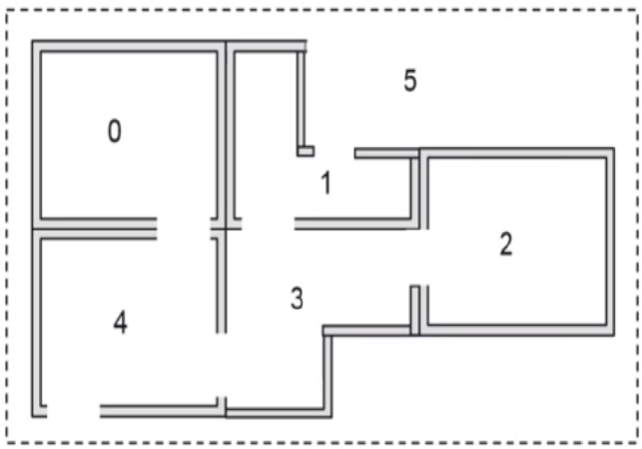
Environment : Junction

State : Total waiting time
wt_1, wt_2, wt_3, wt_4

Reward : depends on increasing / decreasing TWT

Value : minimize the TWT

House Example



Environment : House

Agent : RL Algorithm

States : current position (Room #)

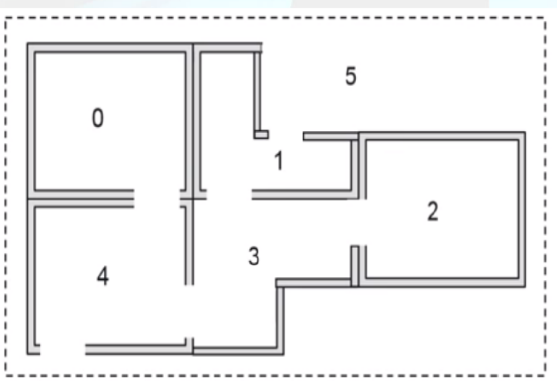
Action : movement

Reward : +/- Reward

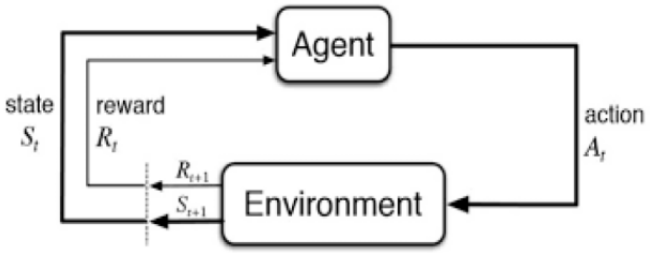
value : Reach #5

Reward Matrix → Define while exploring the environment

States \ Actions	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	-1	-1	100
5	-1	-1	-1	-1	0	100



RL Trading Bot



Agent : Bot (RL Algorithm)
Actions : Buy, Sell, Hold
Environment : Stock market
State : # stocks you have
stock price
Account Balance
Reward : Profit / Loss
Value : maximize the profit

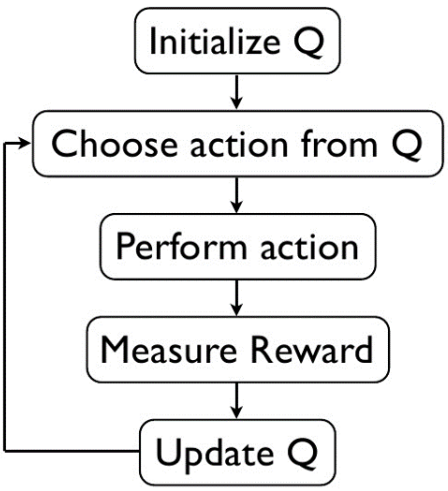
Reward matrix

		Actions		
		Buy	Sell	Hold ← action space
States	S1 (1000, 2, 500)	0	100	70
	S2 (1010, 3, 200)	-1	0	100
state space	S3 (1500, 5, 100)	0	100	20
	⋮	⋮		
	Sn			

Q matrix

$\alpha = 1$ $\quad Q = R + \gamma \cdot \max Q$

$$Q(state, action) \leftarrow (1-\alpha)Q(state, action) + \alpha\left(reward + \gamma \max_a Q(next\ state, all\ actions)\right)$$



R matrix

State \ Action	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100

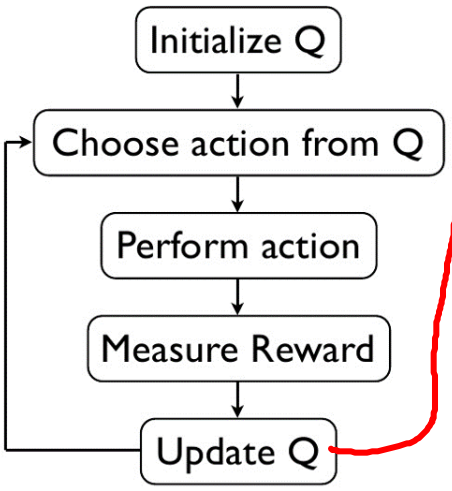
The -1's in the table represent null values

Q matrix

States \ Actions	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	-1	-1	100
5	-1	-1	-1	-1	0	100

$0 \le \gamma \le 1$ $\quad \gamma = 0.8$

$$Q(S,A) \leftarrow reward + \gamma \max_a Q(next\ state, all\ actions)$$



R matrix

Action

State	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100

$R =$

The -1's in the table represent null values

Q matrix

States \ Actions	0	1	2	3	4	5
0	0	0	0	0	80	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	80	0
4	0	0	0	0	0	100
5	0	0	0	0	0	0

$Q(2,3) = 0 + 0.8(0) = 0$

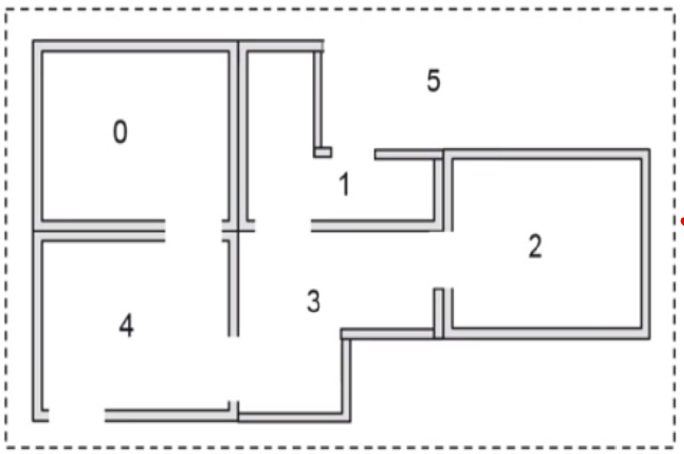
$Q(3,4) = 0 + 0.8(0) = 0$

$Q(4,5) = 100 + 0.8(0) = 100$

$Q(0,4) = 0 + 0.8(100) = 80$

$Q(4,3) = 0 + 0.8(0) = 0$

$Q(3,4) = 0 + 0.8(100) = 80$

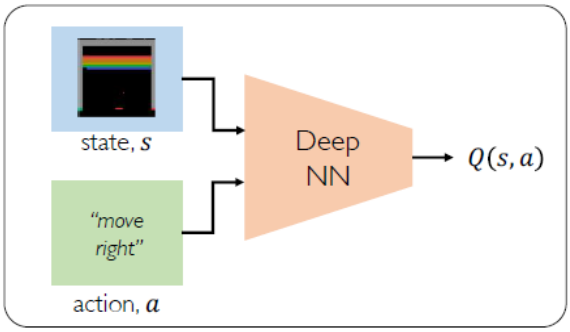


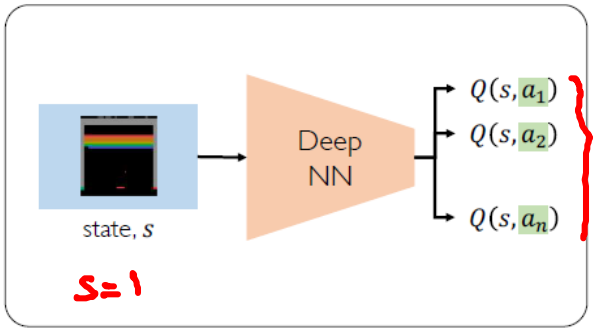
trained Q

Actions

states	0	1	2	3	4	5
0	0.	0.	0.	0.	80.	0.
1	0.	0.	0.	64.	0.	100.
2	0.	0.	0.	-64.	0.	0.
3	0.	80.	51.2	0.	-80.	0.
4	64.	0.	0.	64.	0.	-100.
5	0.	80.	0.	0.	80.	100.

How can we use deep neural networks to model Q-functions?





$$\mathcal{L} = \mathbb{E}\left[\left\|\overbrace{\left(r + \gamma \max_{a'} Q(s', a')\right)}^{\text{target}} - \overbrace{Q(s,a)}^{\text{predicted}}\right\|^2\right]$$

↑ actual