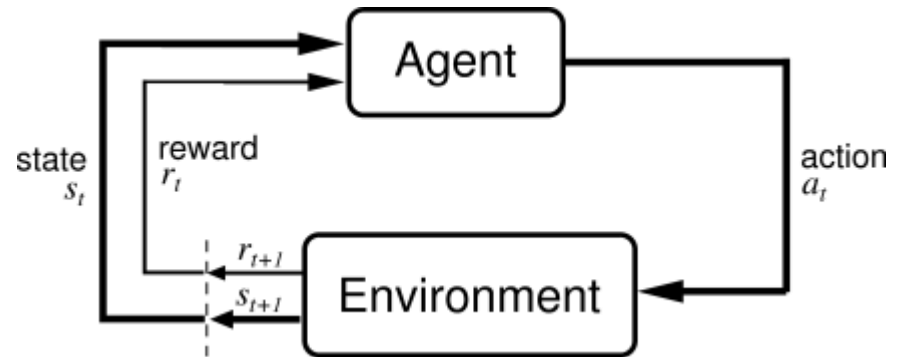




# Q-LEARNING LAB

---

# Q-Learning

- Simulation
- Learning
- Demonstration

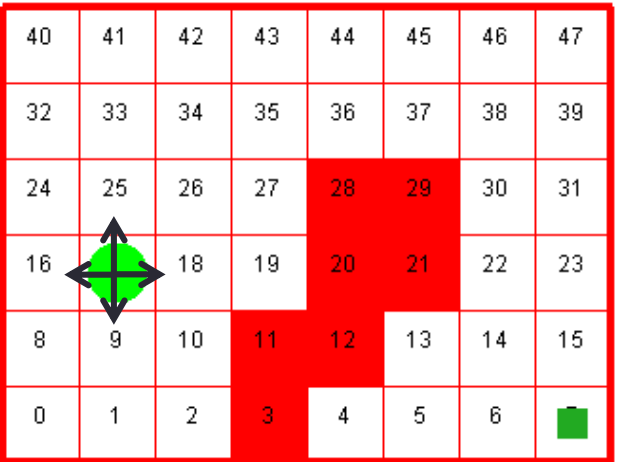


40	41	42	43	44	45	46	47
32	33	34	35	36	37	38	39
24	25	26	27	28	29	30	31
16		18	19	20	21	22	23
8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	

The diagram illustrates the structure of the Q-table and the reward vector. The Q-table is a 6x4 grid, and the reward vector  $r(s')$  is a 1x6 grid. The Q-table is labeled  $Q(s,a)$  on the left, and the reward vector is labeled  $r(s')$  on the left. The Q-table has 6 rows and 4 columns. The reward vector has 6 cells, with the last cell containing an ellipsis ( $\dots$ ).

# Simulation

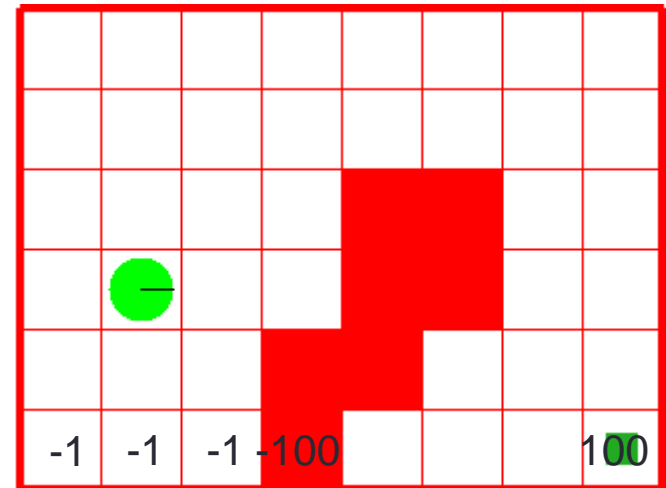
- Randomly choose an action
  - `a = rand.nextInt(4);`
- Stochastically move in that direction
  - `chance = rand.nextDouble();`
  - if `chance < 0.6`
    - go in the direction of action
  - else if `chance < 0.7`
    - go to another direction unless wall (stay in place if wall)
  - ....
  - else
    - stay in place



40	41	42	43	44	45	46	47
32	33	34	35	36	37	38	39
24	25	26	27	28	29	30	31
16	25	18	19	20	21	22	23
8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7

# Learning

- Construct the reward vector
  - -1 for empty cells
  - 100 for the goal state
  - -100 for wall cells (unimportant)



$r(s')$

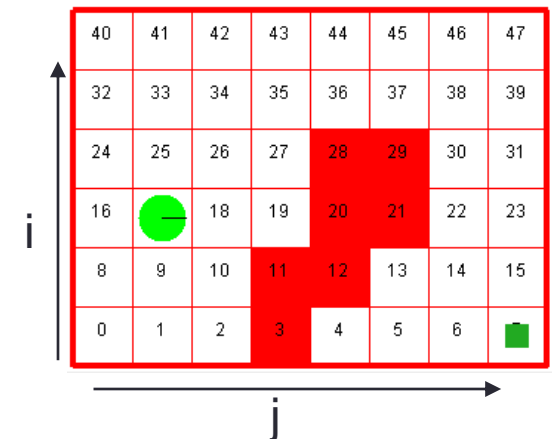
-1	-1	-1	-100				100	...	-1
----	----	----	------	--	--	--	-----	-----	----

# Learning

- Construct the Q-Table
  - Each row  $\rightarrow$  a state
  - Each column  $\rightarrow$  an action
  - Initialize table with 0s
- Useful operations
  - i.e.  $Q(s_3, E)$
  - i.e.  $\max_{a'} Q(s_4', a')$
- $s = (\text{roboti} * \text{colNum}) + \text{robotj}$
- $\text{roboti} = (\text{int}) (s / \text{colNum})$
- $\text{robotj} = s \% \text{colNum}$

Q (s,a)

	N	E	S	W
$s_0$	0	0	0	0
$s_1$	0	0	0	0
$s_2$	0	0	0	0
$s_3$	0	0	0	0
...	0	0	0	0
$s_{n-1}$	0	0	0	0



# Learning

---

**Algorithm 1** Q-Learning Algorithm

---

Define  $\alpha = 0.65, \gamma = 0.99$

**repeat**

    Initialize  $s$  to robot or random location

**repeat**

        Choose an action  $a$  from state  $s$  using random exploration policy

        Take the action  $a$ , observe reward  $r(s')$  and next state  $s'$

$Q(s, a) \leftarrow Q(s, a) + \alpha[r(s') + \gamma \max_{a'} Q(s', a') - Q(s, a)]$

$s \leftarrow s'$

**until**  $s$  reaches goal or max episode length

**until** number of episodes reached (e.g. 150 steps)

---

Single Episode

# Learning

the state we ended up  
after performing action a

repeat

Choose an action  $a$  from state  $s$  using random exploration policy

Take the action  $a$ , observe reward  $r(s')$  and next state  $s'$

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r(s') + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

$$s \leftarrow s'$$

until  $s$  reaches goal or max episode length

choose action  $a'$  for  
which Q-value is  
maximized for state  $s'$

→ expect to follow  
max policy after  
action  $a$  is taken at  
state  $s$

- Current state:  $s_2$
- Choose an action randomly:  $W$
- Simulate the robot and **observe** the next state  $s'$
- Assume next state:  $s_1$

$$Q(s_2, W) \leftarrow Q(s_2, W) + 0.65 * [r(s_1) + 0.99 * Q(s_1, a') - Q(s_2, W)]$$

# Demonstration

- Start from initial state
- Find the max valued action
- Use simulator to move the robot
- E.g. initial state:  $s_1$

		N	E	S	W
Q (s,a)	$s_0$	-10	10	20	0
	$s_1$	10	10	20	0
	$s_2$	0	0	0	0
	$s_3$	0	0	0	0
	...	0	0	0	0
	$s_{n-1}$	0	0	0	0



# Questions ??

- Simulation
- Learning
- Demonstration

40	41	42	43	44	45	46	47
32	33	34	35	36	37	38	39
24	25	26	27	28	29	30	31
16	17	18	19	20	21	22	23
8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7

$r(s')$ 

					...	
--	--	--	--	--	-----	--

 $Q(s,a)$ [illegible]