Reinforcement learning

Dhia Eddine Ben Messaoud 1116625

TERMS

Reward / Punishment: The feedback from the environment to the action performed by the agent, indicating (good / bad).

Q-Value: A measure of the expected long-term return of a state-action pair.

Value Function: A function that specifies the expected cumulative q-value that the agent can achieve from a given state.

Q-table: where the expected rewards for all possible actions in a given state are stored.

Policy

Policy refers to an agent's strategy to interact with an environment.

It determines the next action an agent takes in response to the current state of the environment.

Epsilon-Greedy Policy

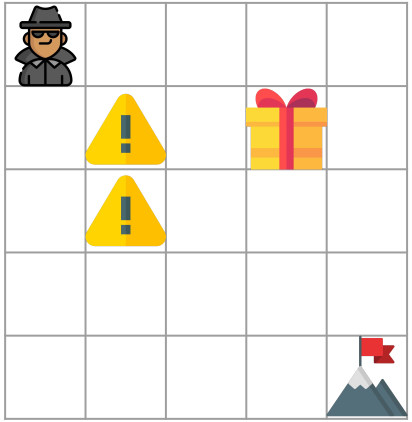
The agent selects the action with the highest Q-value most of the time but randomly chooses an action with probability ϵ.

Exploration: The agent randomly selects an action 20 % of the time. This helps the agent explore the environment.

Exploitation: The agent selects the action with the highest Q-value 80 % of the time. This ensures that the agent makes use of the knowledge it has already gained to maximize rewards.

Learning Phase

```
alpha = 0.9   # Learning rate
gamma = 0.9   # Discount factor
epsilon = 0.5   # Exploration rate
```



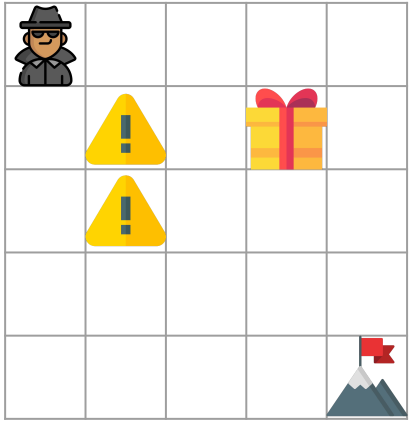
```
1.  choose action:

    if random < epsilon:
        Exploration

    else: Exploitation
    (get action based on max q_values)
```

	up	down	right	left
(0, 0)	0	0	0	0

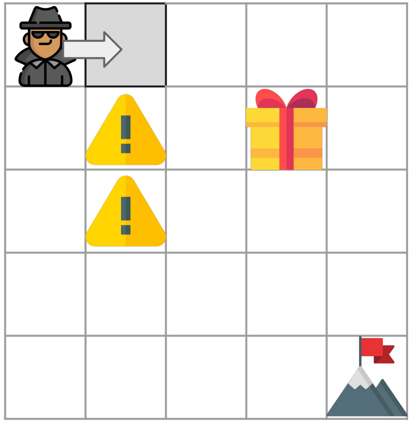
```
alpha = 0.9   # Learning rate
gamma = 0.9   # Discount factor
epsilon = 0.5   # Exploration rate
```



1. choose action:

if random < epsilon:
    Exploration (action = right)

```
alpha = 0.9   # Learning rate
gamma = 0.9   # Discount factor
epsilon = 0.5   # Exploration rate
```



2. get next state based on the action
   check if the action is valid

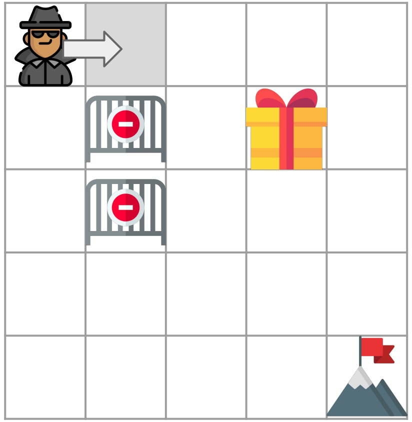
3. get reward or punishment

 = +100

 = +50

 = -50

```
alpha (α) = 0.9  # Learning rate
gamma (γ) = 0.9  # Discount factor
epsilon = 0.5  # Exploration rate
```



4. update q table value using Bellmann function:

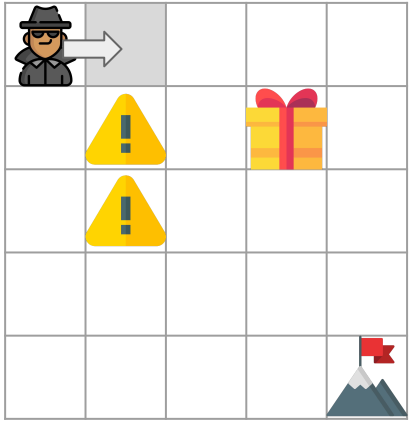
F:

```
q_table[s][a] += α * (reward + γ * max_future_q - q_table[s][a])
```

	up	down	right	left
(0 ,0)	0	0	**0**	0
(0, 1)	0	0	0	0

max

```
alpha (α) = 0.9  # Learning rate
gamma (γ) = 0.9  # Discount factor
epsilon = 0.5  # Exploration rate
```



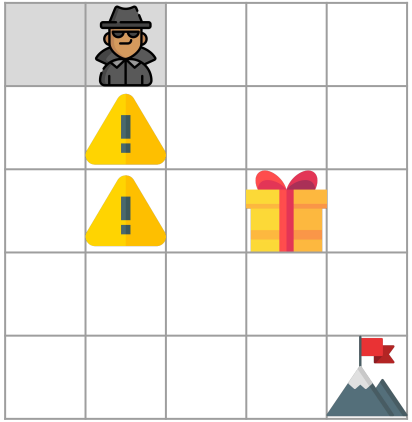
4. update q table value using Bellmann function:

```
state = (0, 0)
action = right
reward = 0
max_future_q = 0
```

```
q_table[s][a] += 0,9 * (0 + 0,9 * 0 - 0) = 0
```

	up	down	right	left
(0 ,0)	0	0	**0**	0
(0, 1)	0	0	0	0

```
alpha (α) = 0.9   # Learning rate
gamma (γ) = 0.9   # Discount factor
epsilon = 0.5   # Exploration rate
```



4. update q table value using Bellmann function:

```
state = (0, 0)
action = right
reward = 0
max_future_q = 0
```

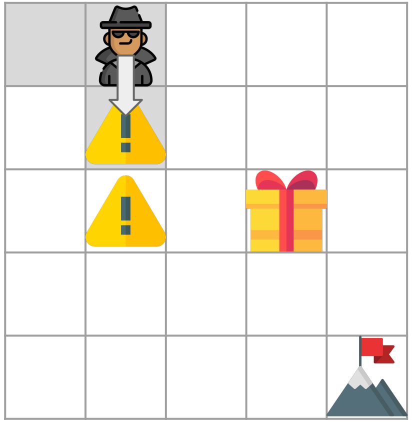
```
q_table[s][a] += 0,9 * (0 + 0,9 * 0 - 0) = 0
```

	up	down	right	left
(0 ,0)	0	0	**0**	0
(0, 1)	0	0	0	0

F:

```
q_table[s][a] += α * (reward + γ * max_future_q - q_table[s][a])
```

```
alpha (α) = 0.9   # Learning rate
gamma (γ) = 0.9   # Discount factor
epsilon = 0.5   # Exploration rate
```



1. choose action (down)
2. get next state and get reward (r = -50)
3. update q table value using **F**

q_table[s][a] += 0,9 * (**-50** + 0,9 * 0 - 0) = -45

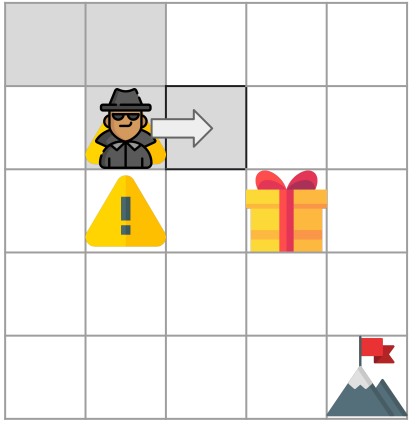
	up	down	right	left
(0 ,1)	0	**- 45**	0	0
(1, 1)	0	0	0	0

max

F:

```
q_table[s][a] += α * (reward + γ * max_future_q - q_table[s][a])
```

```
alpha (α) = 0.9   # Learning rate
gamma (γ) = 0.9   # Discount factor
epsilon = 0.5   # Exploration rate
```



1. choose action (right)
2. get next state and get reward **(r = 0)**
3. update q table value using **F**

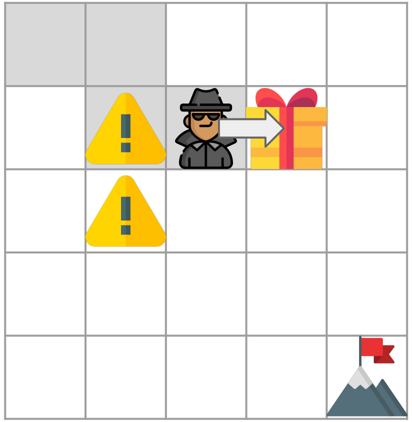
```
q_table[s][a] += 0,9 * (0 + 0,9 * 0 - 0) = 0
```

max

	up	down	right	left
(1 ,1)	0	-45	**0**	0
(1, 2)	0	0	0	0

F:
`q_table[s][a] += α * (reward + γ * max_future_q - q_table[s][a])`

```
alpha (α) = 0.9  # Learning rate
gamma (γ) = 0.9  # Discount factor
epsilon = 0.5  # Exploration rate
```



1. choose action (right)
2. get next state and get reward **(r = 50)**
3. update q table value using **F**

`q_table[s][a] += 0,9 * (+50 + 0,9 * 0 - 0) = +45`

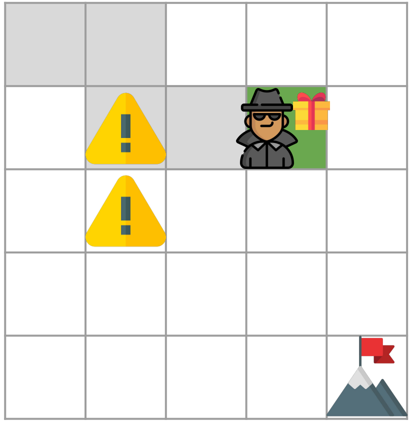
max

	up	down	right	left
(1 ,2)	0	0	**+45**	0
(1, 3)	0	0	0	0

F:

```
q_table[s][a] += α * (reward + γ * max_future_q - q_table[s][a])
```

```
alpha (α) = 0.9   # Learning rate
gamma (γ) = 0.9   # Discount factor
epsilon = 0.5   # Exploration rate
```

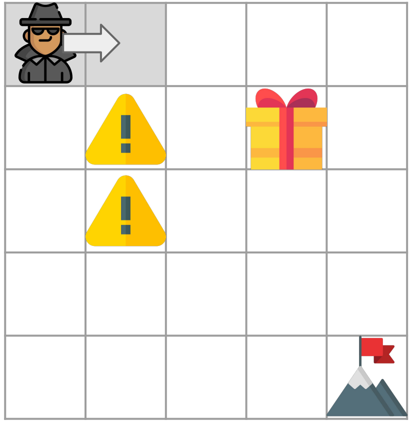


Agent has reached the goal
Episode is finished

	up	down	right	left
(0, 0)	0	0	0	0
(0, 1)	0	- 45	0	0
(1, 1)	0	0	0	0
(1, 2)	0	0	+ 45	0
(1, 3)				
..	0	0	0	0

F: q_table[s][a] += α * (reward + γ * max_future_q - q_table[s][a])

```
alpha (α) = 0.9   # Learning rate
gamma (γ) = 0.9   # Discount factor
epsilon = 0.5   # Exploration rate
```



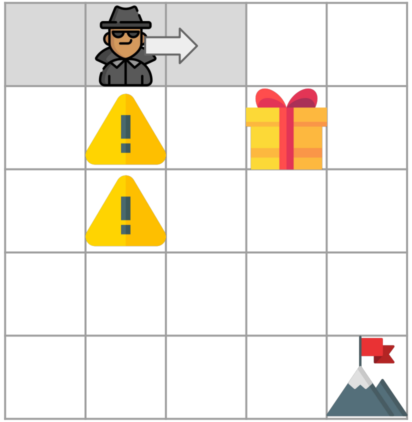
1. choose action (right)
2. get next state and get reward **(r = 0)**
3. update q table value using **F**

q_table[s][a] += 0,9 * (0 + 0,9 * 0 - 0) = 0

	up	down	right	left
(0 ,0)	0	0	**0**	0
(0, 1)	0	- 45	0	0

```
F:
q_table[s][a] += α * (reward + γ * max_future_q - q_table[s][a])
```

```
alpha (α) = 0.9   # Learning rate
gamma (γ) = 0.9   # Discount factor
epsilon = 0.5   # Exploration rate
```



1. choose action
   Exploitation:
   Max(up:0, down:-45, right:0, left:0)
   action = right (randomly between up,r and l)

2. get next state and get reward **(r = 0)**
3. update q table value using **F**

q_table[s][a] += 0,9 * (0 + 0,9 * 0 - 0) = 0

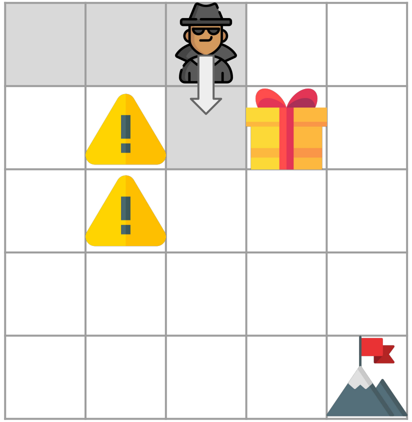
	up	down	right	left
(0, 1)	0	- 45	**0**	0
(0, 2)	0	0	0	0

max

F:

```
q_table[s][a] += α * (reward + γ * max_future_q - q_table[s][a])
```

```
alpha (α) = 0.9   # Learning rate
gamma (γ) = 0.9   # Discount factor
epsilon = 0.5   # Exploration rate
```



1. choose action (right)
2. get next state and get reward **(r = 0)**
3. update q table value using **F**

```
q_table[s][a] += 0,9 * (0 + 0,9 * 45 - 0) = 36,45
```

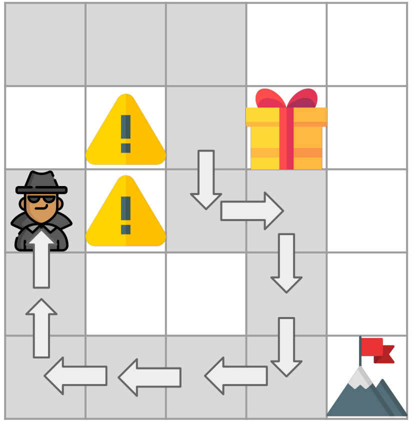
max

	up	down	right	left
(0 ,2)	0	0	**36,45**	0
(1, 2)	0	0	+ 45	0

F:

```
q_table[s][a] += α * (reward + γ * max_future_q - q_table[s][a])
```

```
alpha (α) = 0.9   # Learning rate
gamma (γ) = 0.9   # Discount factor
epsilon = 0.5  # Exploration rate
```



1. choose action (random)
2. get next state and get reward
3. update q table value using **F**

after max steps without reaching the goal, the episode is finished and the agent starts a new episode.

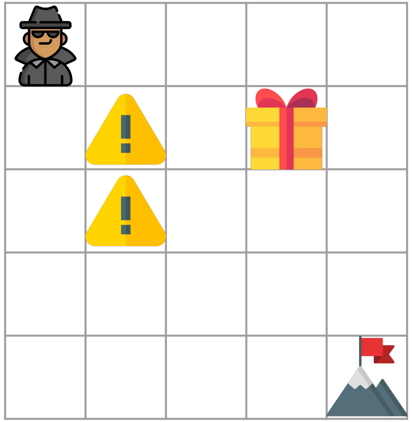
Result Q _table after learning phase

html link

	up	down	left	right
(0, 0)	0.00	47.83	0.00	47.83
(0, 1)	0.00	-46.86	43.05	53.14
(0, 2)	0.00	59.05	47.83	59.05
(0, 3)	0.00	0.00	53.14	65.61
(0, 4)	0.00	72.90	59.05	0.00
(1, 0)	43.05	53.14	0.00	-46.86
(1, 1)	47.83	-40.95	47.83	59.05
(1, 2)	53.14	65.61	-46.86	0.00
(1, 3)	0.00	0.00	0.00	0.00
(1, 4)	65.61	81.00	0.00	0.00
(2, 0)	47.83	59.05	0.00	-40.95
(2, 1)	-46.86	65.61	53.14	65.61
(2, 2)	59.05	72.90	-40.95	72.90
(2, 3)	0.00	81.00	65.61	81.00
(2, 4)	72.90	90.00	72.90	0.00

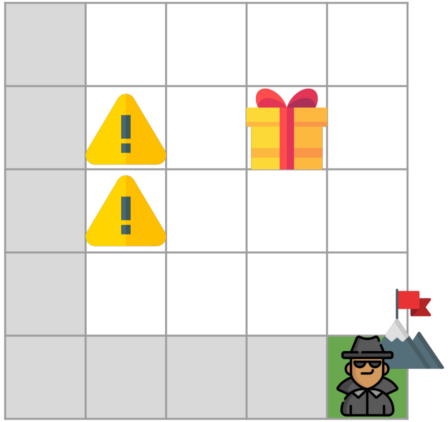
	up	down	left	right
(3, 0)	53.14	65.61	0.00	65.61
(3, 1)	-40.95	72.90	59.05	72.90
(3, 2)	65.61	81.00	65.61	81.00
(3, 3)	72.90	90.00	72.90	90.00
(3, 4)	81.00	100	81.00	0.00
(4, 0)	59.05	0.00	0.00	72.90
(4, 1)	65.61	0.00	65.61	81.00
(4, 2)	72.90	0.00	72.90	90.00
(4, 3)	81.00	0.00	81.00	100
(4, 4)	0.00	0.00	0.00	0.00
(3, 0)	53.14	65.61	0.00	65.61
(3, 1)	-40.95	72.90	59.05	72.90
(3, 2)	65.61	81.00	65.61	81.00
(3, 3)	72.90	90.00	72.90	90.00
(3, 4)	81.00	100	81.00	0.00

Testing Phase



1. choose action:
   based on the q_table from the learning phase choose the action with the highest value

2. repeat until the agent reaches the goal



state \ action	up	down	left	right
(0, 0)	0.00	47.83	0.00	47.83
(1, 0)	43.05	53.14	0.00	-46.86
(2, 0)	47.83	59.05	0.00	-40.95
(3, 0)	53.14	65.61	0.00	65.61
(4, 0)	59.05	0.00	0.00	72.90
(4, 1)	65.61	0.00	65.61	81.00
(4, 2)	72.90	0.00	72.90	90.00
(4, 3)	81.00	0.00	81.00	100
(4, 4)	0.00	0.00	0.00	0.00

Best path: [(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (4, 1), (4, 2), (4, 3), (4, 4)]

Danke für Ihre Aufmerksamkeit