Lecture 4: Q-learning (table)
exploit&exploration and discounted future reward

Reinforcement Learning with TensorFlow&OpenAI Gym

Sung Kim <hunkim+ml@gmail.com>

Dummy Q-learning algorithm

For each s, a initialize table entry $\hat{Q}(s, a) \leftarrow 0$

Observe current state s

Do forever:

- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s, a)$ as follows:

$$\hat{Q}(s, a) \leftarrow r + \max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$

Machine Learning, T. Mitchell, McGraw Hill, 1997

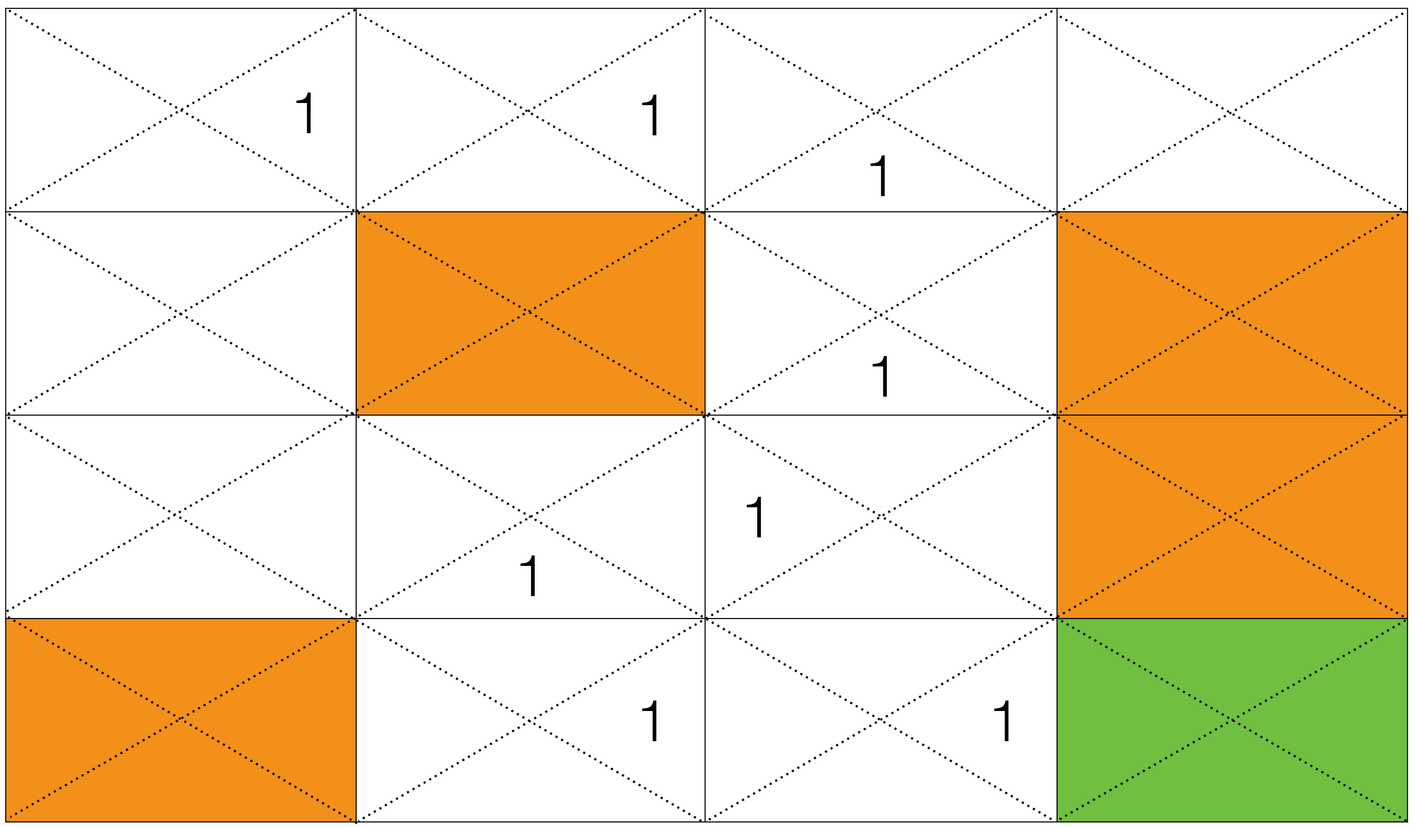
Learning Q (s, a)?

$$\hat{Q}(s,a) \leftarrow r + \quad \max_{a'} \hat{Q}(s',a')$$

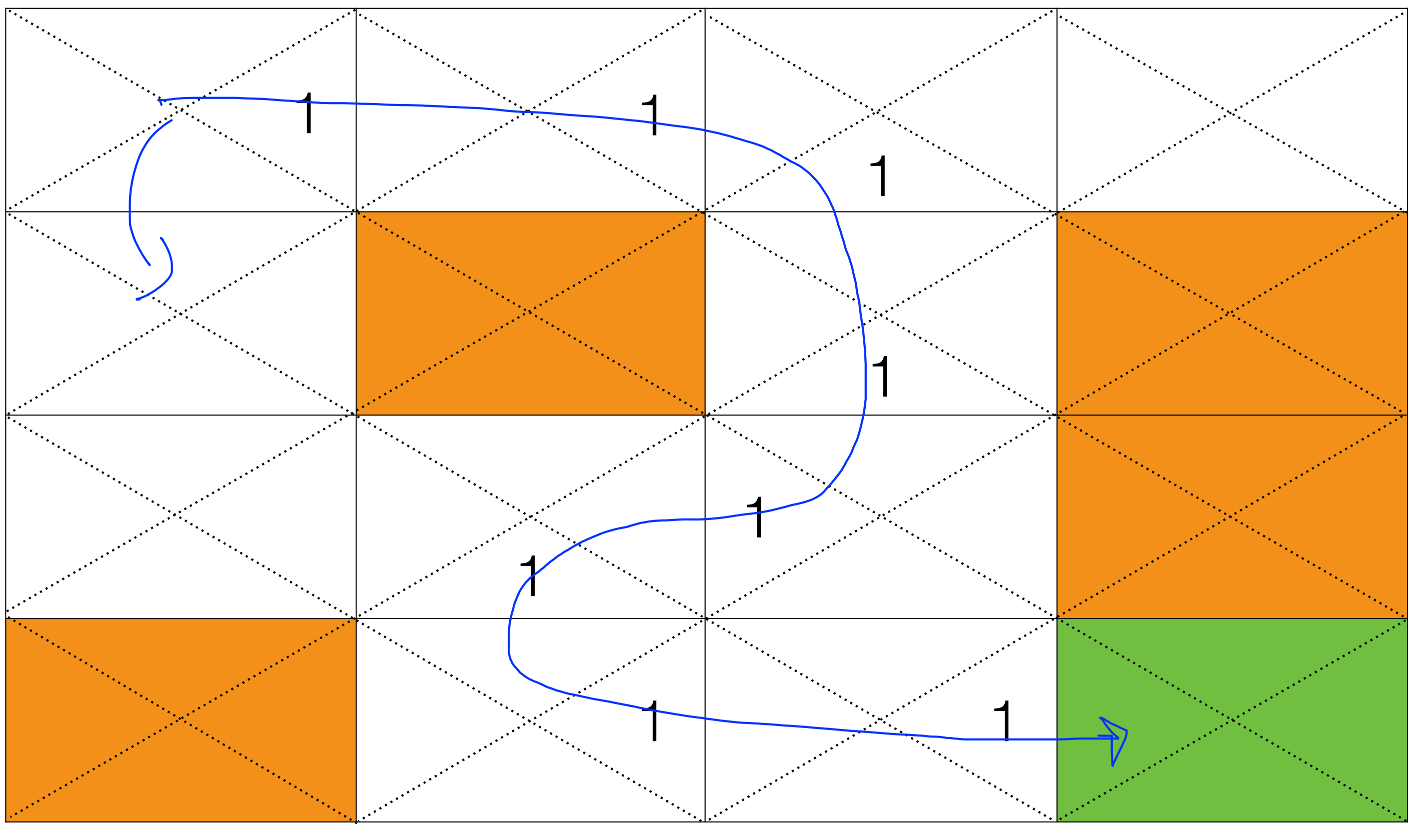
Machine Learning, T. Mitchell, McGraw Hill, 1997

Learning Q(s, a) Table: one success!

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$



Exploit VS Exploration



http://home.deib.polimi.it/restelli/MyWebSite/pdf/rl5.pdf

Exploit VS Exploration

0.0 0.0 0.0 0.0 0.0

Exploit (weekday) VS Exploration (weekend)

0.5 0.6 0.3 0.2 0.5

Exploit VS Exploration: $\boxed{\text{E-greedy}}$

```
e = 0.1

if rand < e:
   a = random    10%

else:
   a = argmax(Q(s, a))    90%
```

Exploit VS Exploration: decaying E-greedy

```
for i in range (1000)
    e = 0.1 / (i+1)
    if random(1) < e:
        a = random
    else:
        a = argmax(Q(s, a))
```

Exploit VS Exploration: add random noise

0.5 0.6 0.3 0.2 0.5

Exploit VS Exploration: add random noise

a = argmax(Q(s, a) + random_values)

+ rand

a = argmax([0.5 0.6 0.3 0.2 0.5]+[0.1 0.2 0.7 0.3 0.1])

[0.6 0.8 1.0 0.5 0.6]

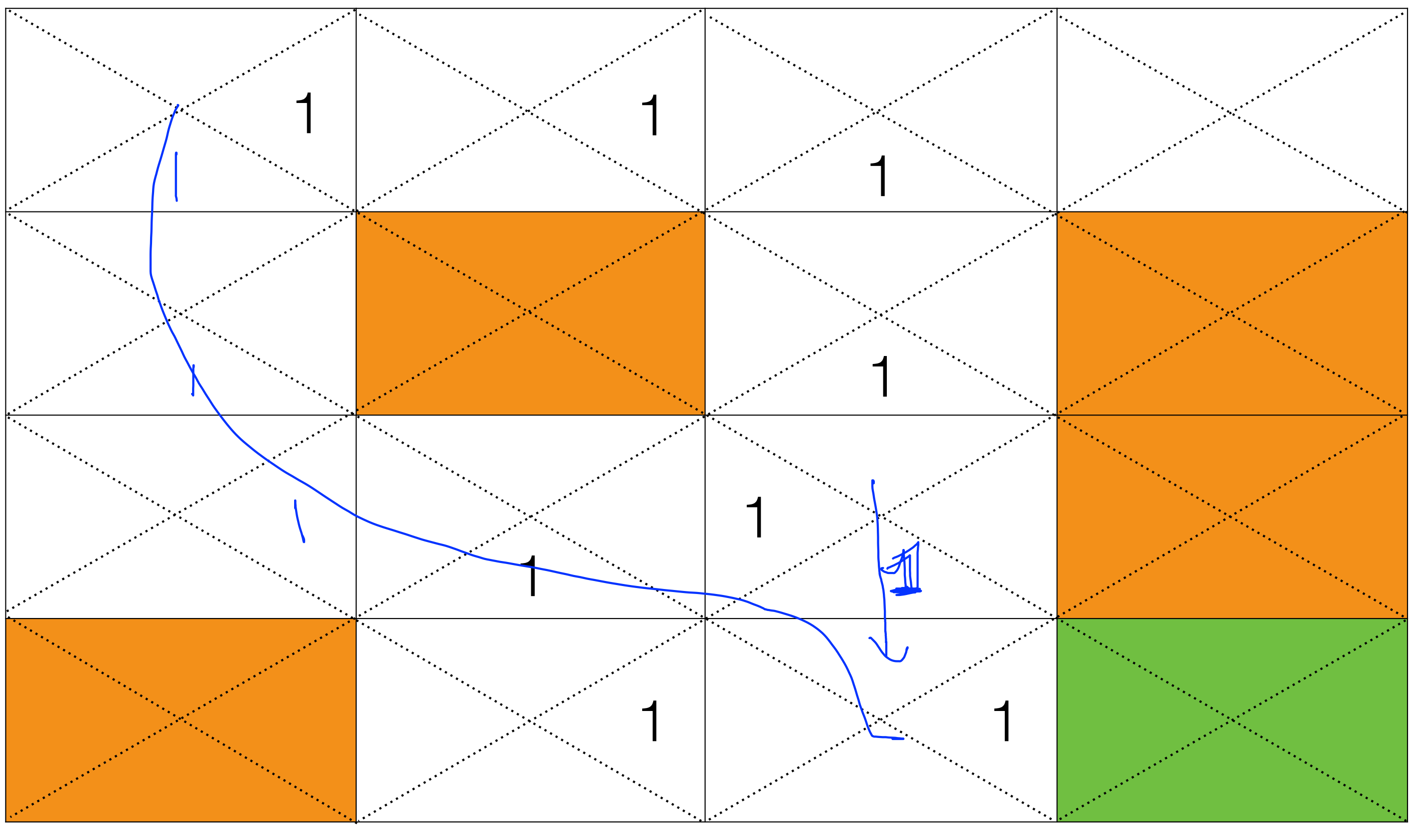
0.5 0.6 0.3 0.2 0.5

Exploit VS Exploration: add random noise

```
for i in range (1000)
    a = argmax(Q(s, a) + random_values / (i+1))
```

0.5 0.6 0.3 0.2 0.5

Exploit VS Exploration



Dummy Q-learning algorithm

For each s, a initialize table entry $\hat{Q}(s,a) \leftarrow 0$

Observe current state s

Do forever: E&E

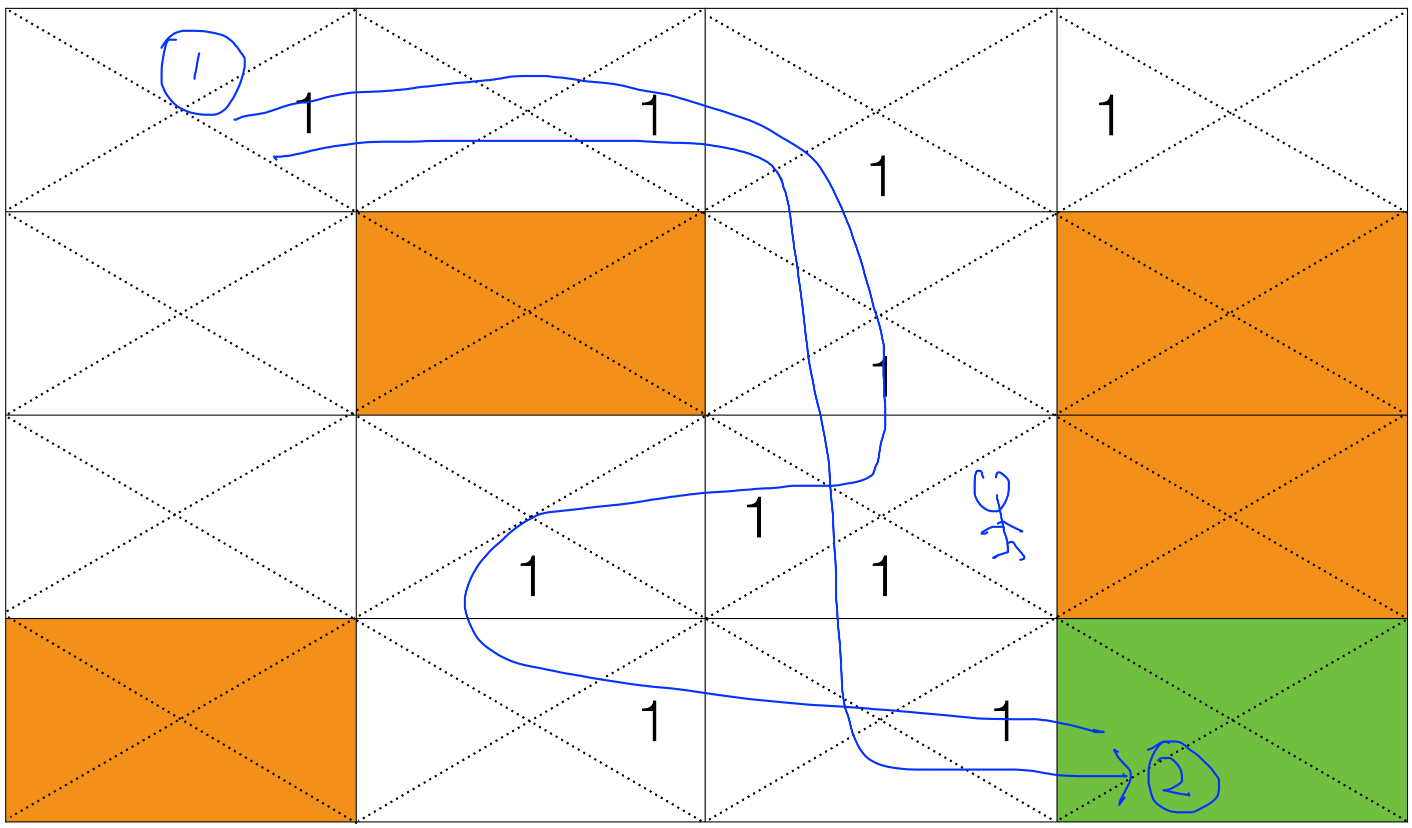
- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s,a)$ as follows:

$$\hat{Q}(s,a) \leftarrow r + \quad \max_{a'} \hat{Q}(s',a')$$

- $s \leftarrow s'$

Machine Learning, T. Mitchell, McGraw Hill, 1997

Discounted future reward



Learning Q (s, a) with discounted reward

$\gamma = 0.9$

$$\hat{Q}(s,a) \leftarrow r + \underset{\gamma}{0.9} \max_{a'} \hat{Q}(s', a')$$

Machine Learning, T. Mitchell, McGraw Hill, 1997

Discounted future reward

- Future reward $R = r_1 + r_2 + r_3 + \cdots + r_n$

  $R_t = r_t + r_{t+1} + r_{t+2} + \cdots + r_n$

- Discounted future reward (environment is stochastic)

$$\begin{aligned} R_t &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{n-t} r_n \\ &= r_t + \gamma(r_{t+1} + \gamma(r_{t+2} + \cdots)) \\ &= r_t + \gamma R_{t+1} \end{aligned}$$

$$\boxed{Q = r + \gamma \max Q(s')}$$

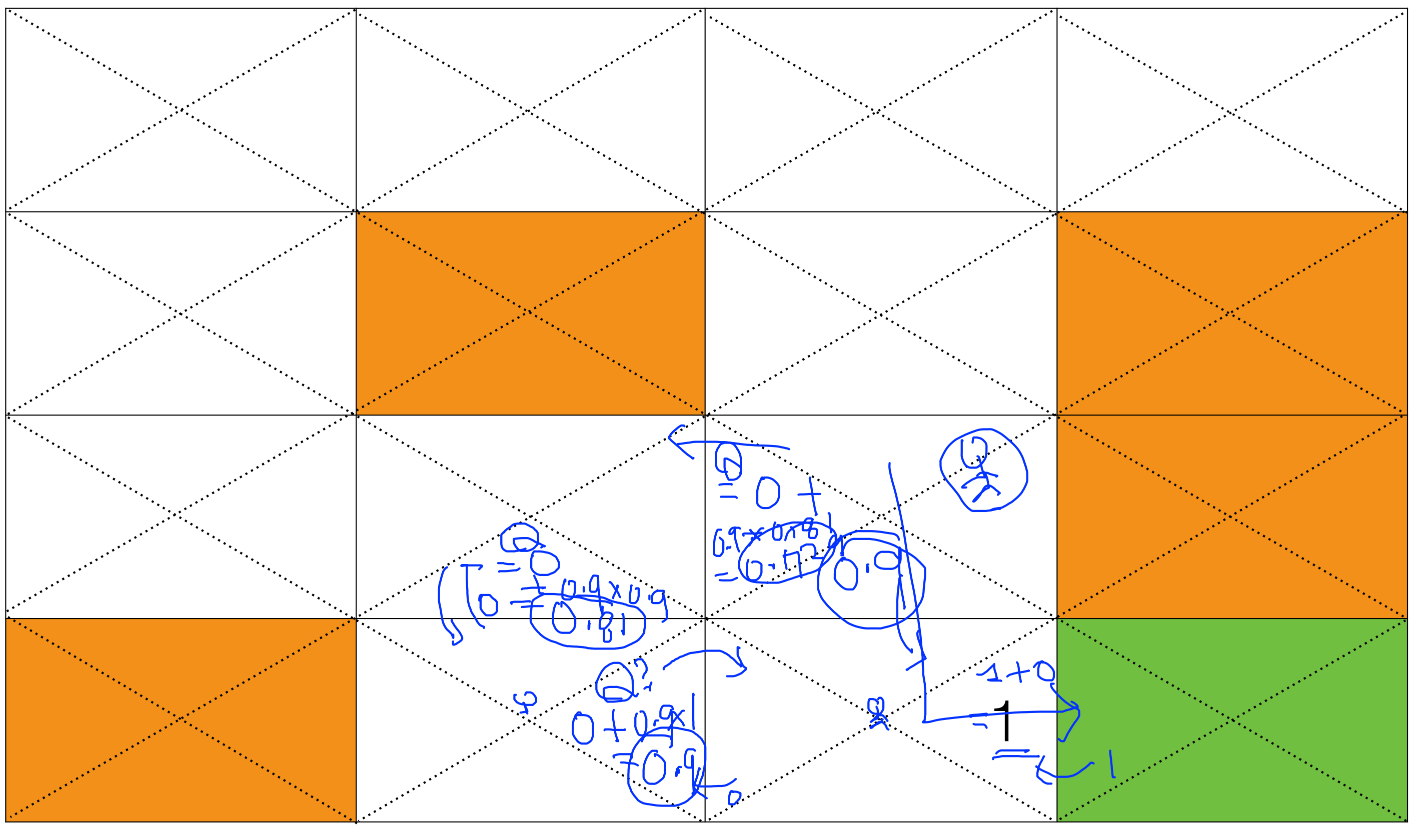
- A good strategy for an agent would be to always choose an action that maximizes the (discounted) future reward

Learning Q (s, a) with discounted reward

$$\boxed{\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')}$$

Machine Learning, T. Mitchell, McGraw Hill, 1997

Discounted reward γ = 0.9)



Q-learning algorithm

For each s, a initialize table entry $\hat{Q}(s, a) \leftarrow 0$

Observe current state s

Do forever:

- Select an action a and execute it E&E
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s, a)$ as follows:

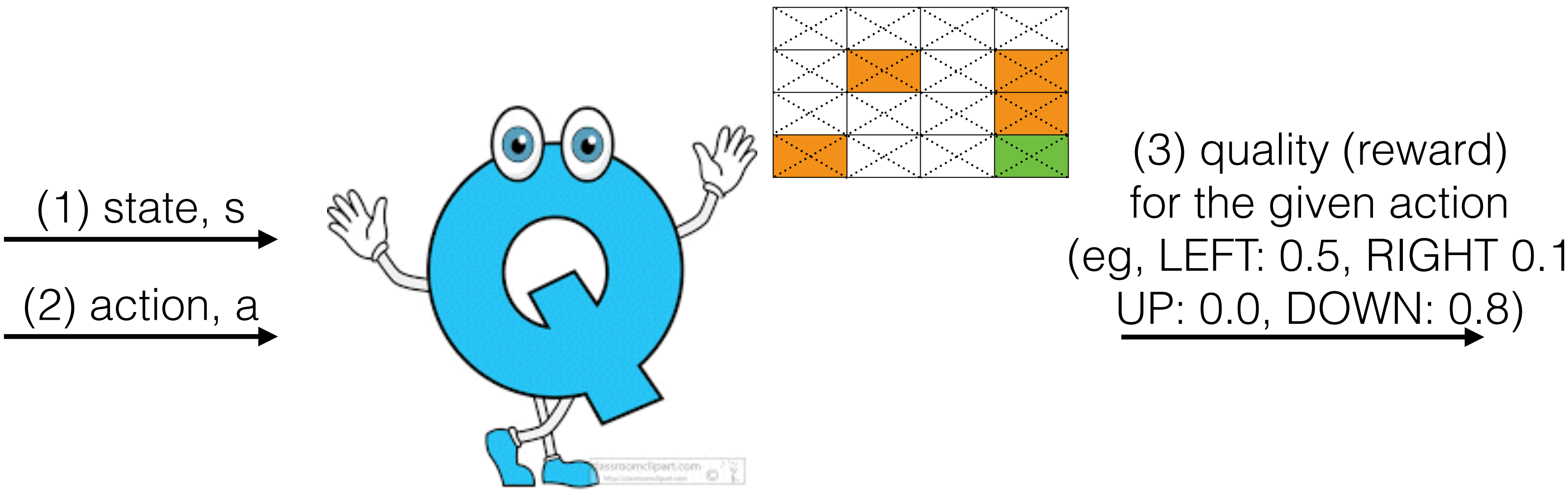
$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$

Machine Learning, T. Mitchell, McGraw Hill, 1997

Q-Table Policy

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$



Q (s, a)

Convergence

$\hat{Q}$ denote learner's current approximation to Q.

$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$$

$\hat{Q}$ converges to Q.

- In deterministic worlds
- In finite states

Machine Learning, Tom Mitchell, 1997

Next

Lab: Q-learning Table