UiO : Department of Informatics
University of Oslo

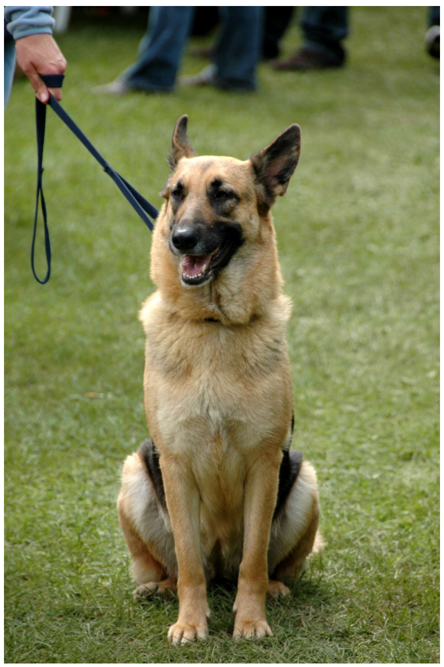
INF3490 - Biologically inspired computing

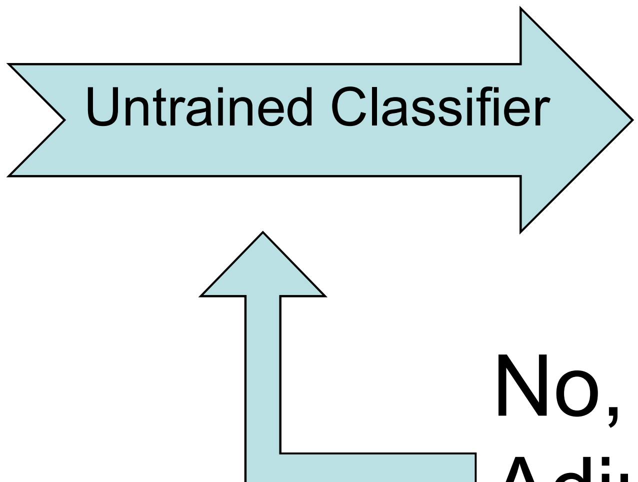
Lecture 12th October 2016

Reinforcement Learning

Kai Olav Ellefsen

Last time: Supervised learning

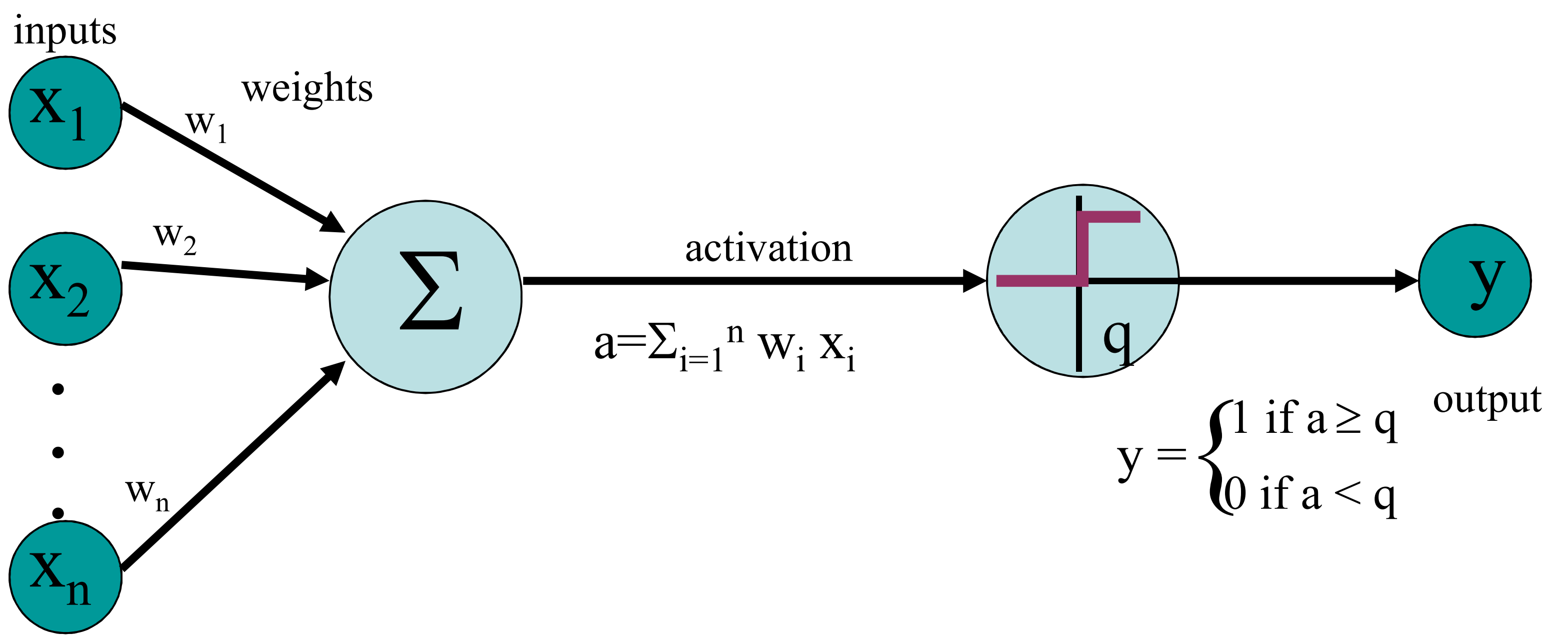




"CAT"

No, it was a dog.
Adjust classifier
parameters

Supervised learning: Weight updates



$$\Delta w_{ij} = \eta \cdot (t_j - y_j) \cdot x_i$$

Learning rate
Input
Desired output
Error
Actual output

UiO : **Department of Informatics**
University of Oslo

Reinforcement Learning: Infrequent Feedback

50 chess moves later

You lost

Update chess-playing strategy

How do we update our system now? We don't know the error.

Learning rate Input

$$\Delta w_{ij} = \eta \cdot (t_j - y_j) \cdot x_i$$

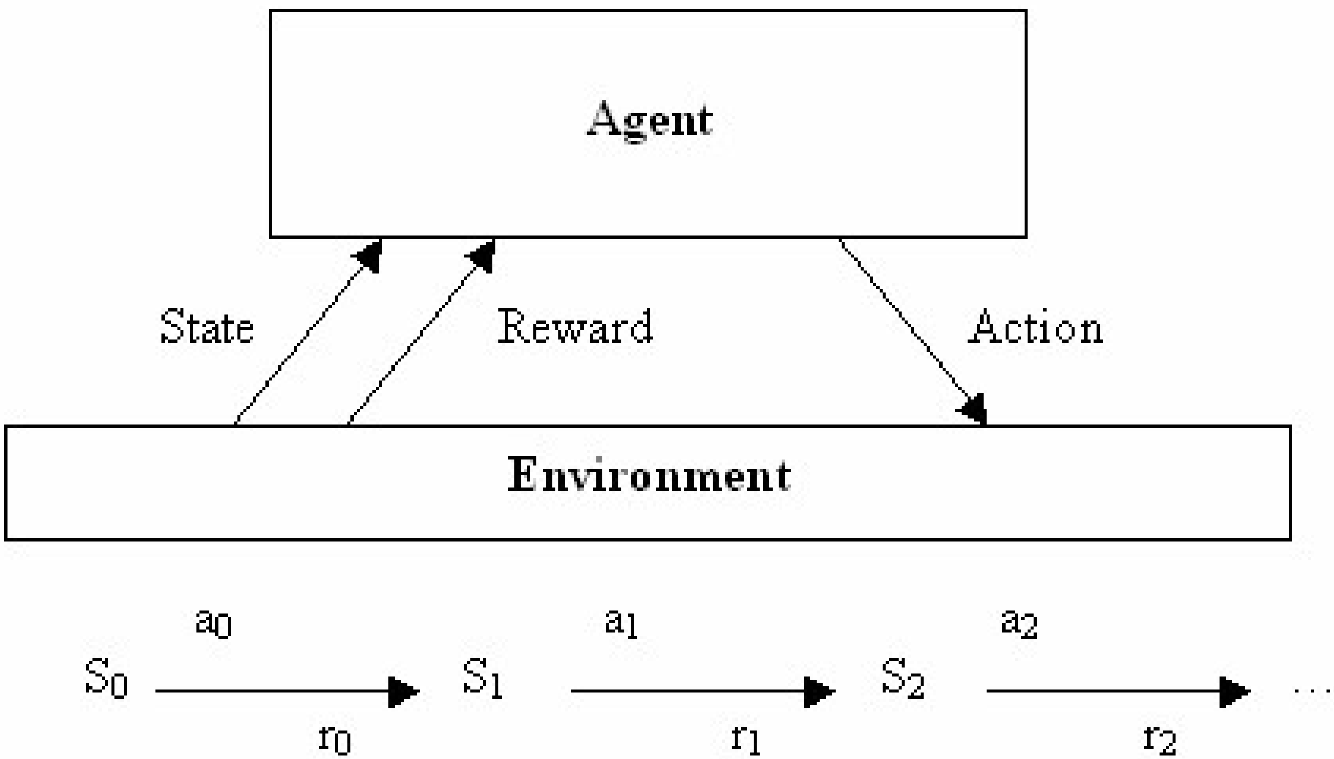
Desired output Actual output

Error

Example


Robot Motor Skill
Coordination with EM-based
Reinforcement Learning
Petar Kormushev, Sylvain Calinon,
and Darwin G. Caldwell
Italian Institute of Technology

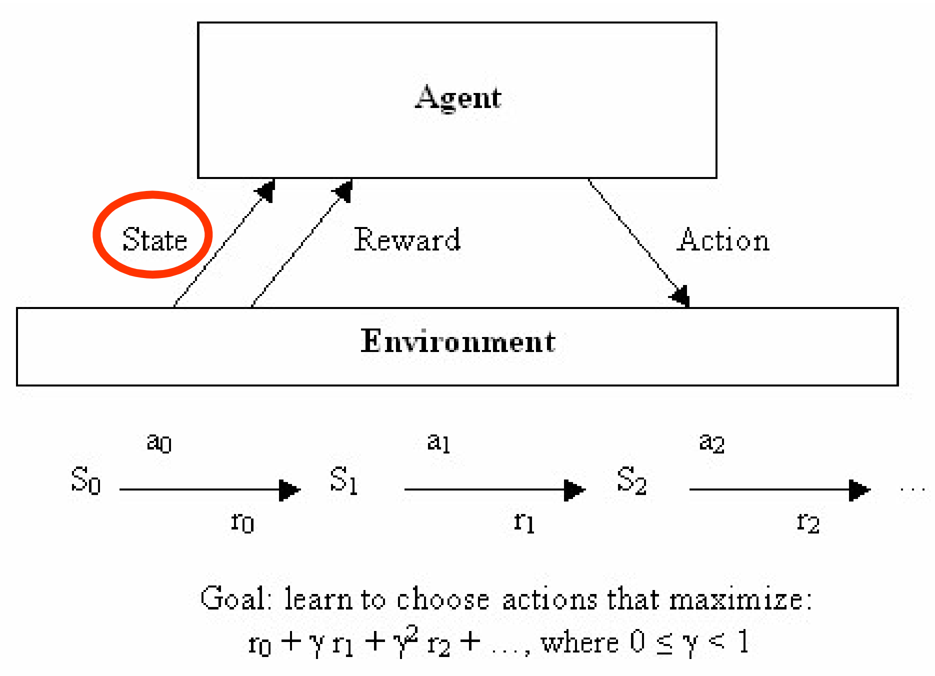
The reinforcement learning problem



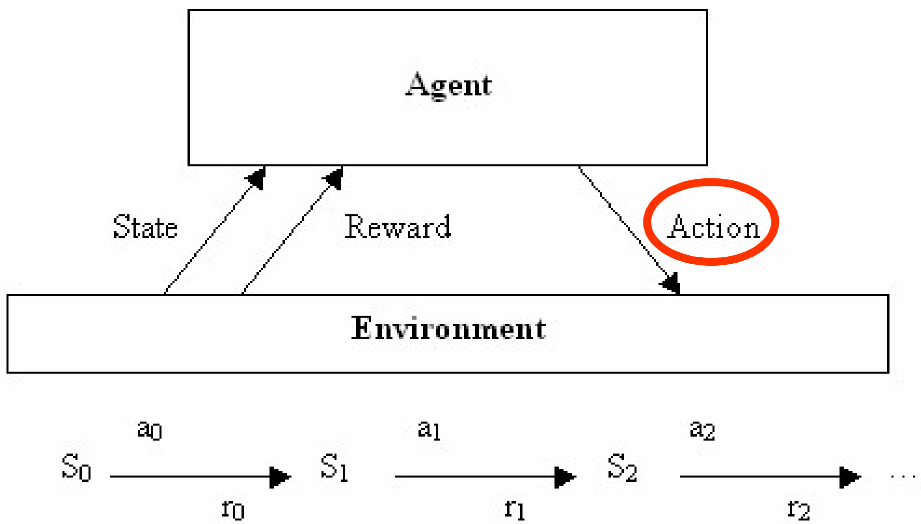
Goal: learn to choose actions that maximize:

$r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$, where $0 \leq \gamma < 1$

The reinforcement learning problem



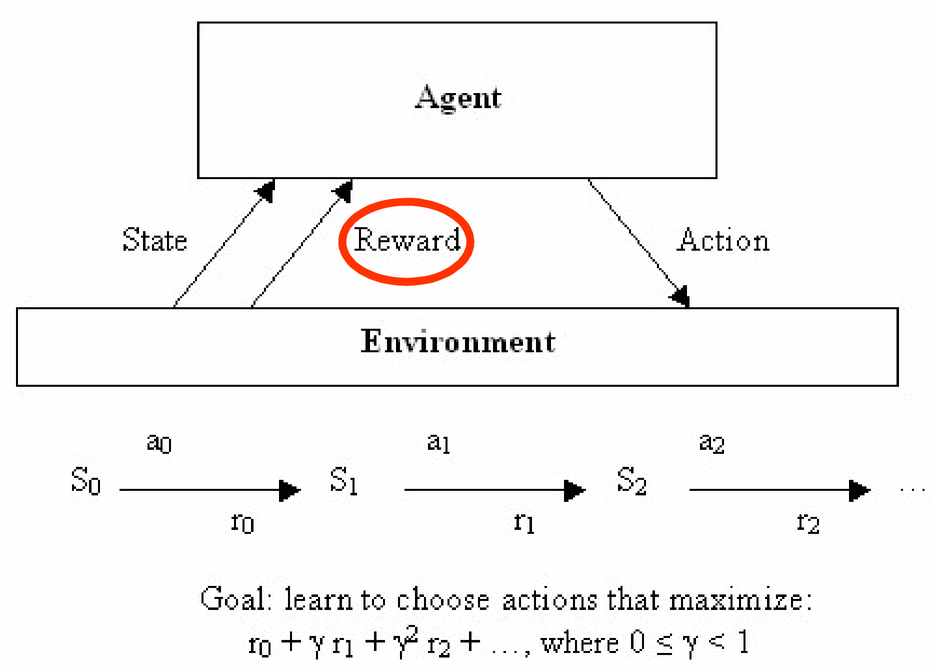
The reinforcement learning problem



Goal: learn to choose actions that maximize:
$r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$, where $0 \leq \gamma < 1$

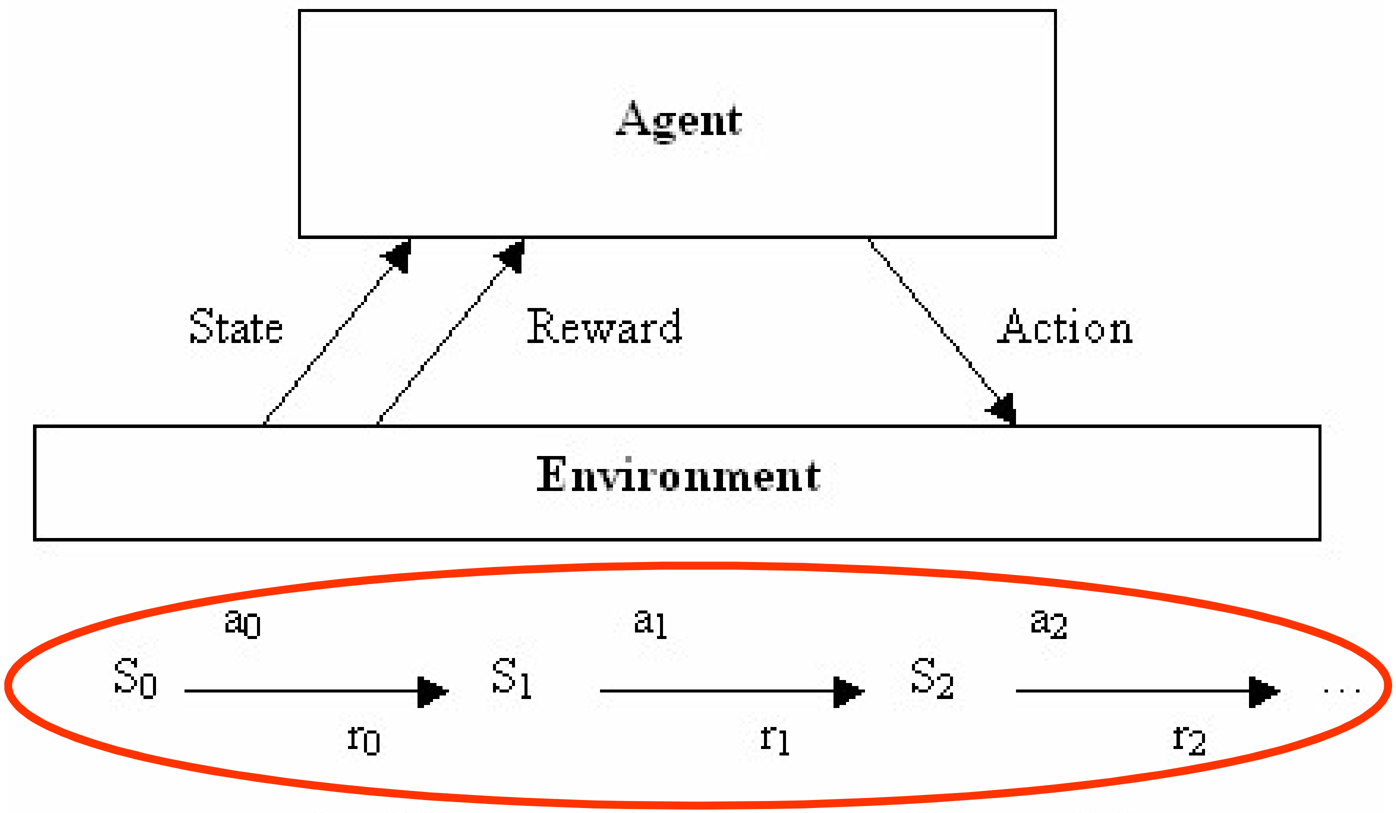
"Move piece from J1 to H1"

The reinforcement learning problem



You took an opponent's piece.
Reward=1

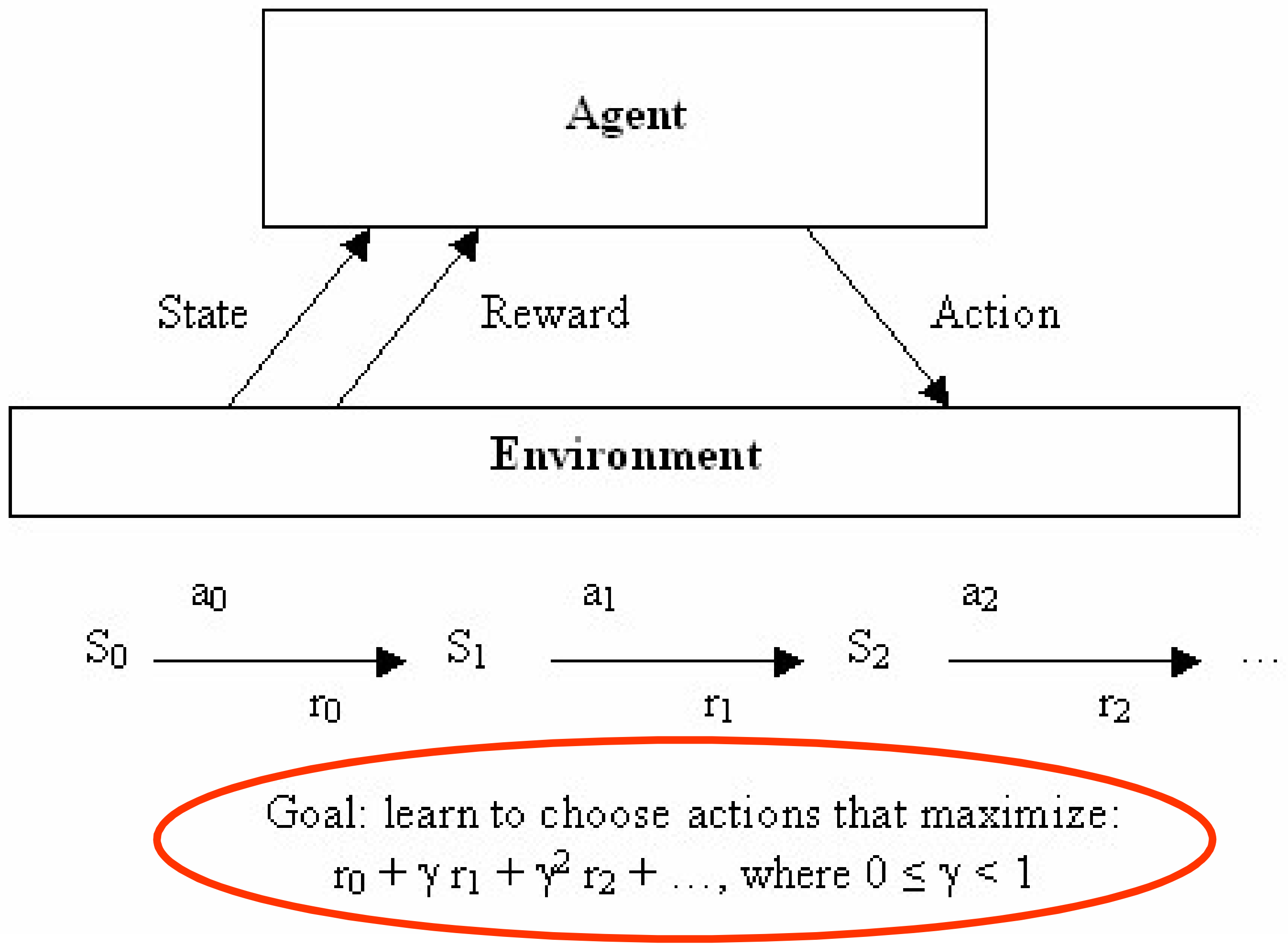
The reinforcement learning problem



Goal: learn to choose actions that maximize:

$r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$, where $0 \le \gamma < 1$

The reinforcement learning problem



Learning is guided by the reward

- An infrequent numerical feedback indicating how well we are doing
- Problems:
  - The reward does not tell us *what we should have done*
  - The reward may be *delayed* – does not always indicate when we made a mistake.

The reward function

- Corresponds to the fitness function of an evolutionary algorithm
- r_{t+1} is a function of (s_t, a_t)
- The reward is a numeric value. Can be negative (“punishment”).
- Can be given throughout the learning episode, or only in the end
- Goal: Maximize total reward

Maximizing total reward

- Total reward:

$$R = \sum_{t=0}^{N-1} r_{t+1}$$

- Future rewards may be uncertain -> We care more about rewards that come soon
- Solution: Discount future rewards:

$$R = \sum_{t=0}^{\infty} \gamma^t \, r_{t+1}, \qquad 0 \leq \gamma \leq 1$$

Discounted rewards example

$$R = \sum_{t=0}^{\infty} \gamma^t \, r_{t+1}, \qquad 0 \leq \gamma \leq 1$$

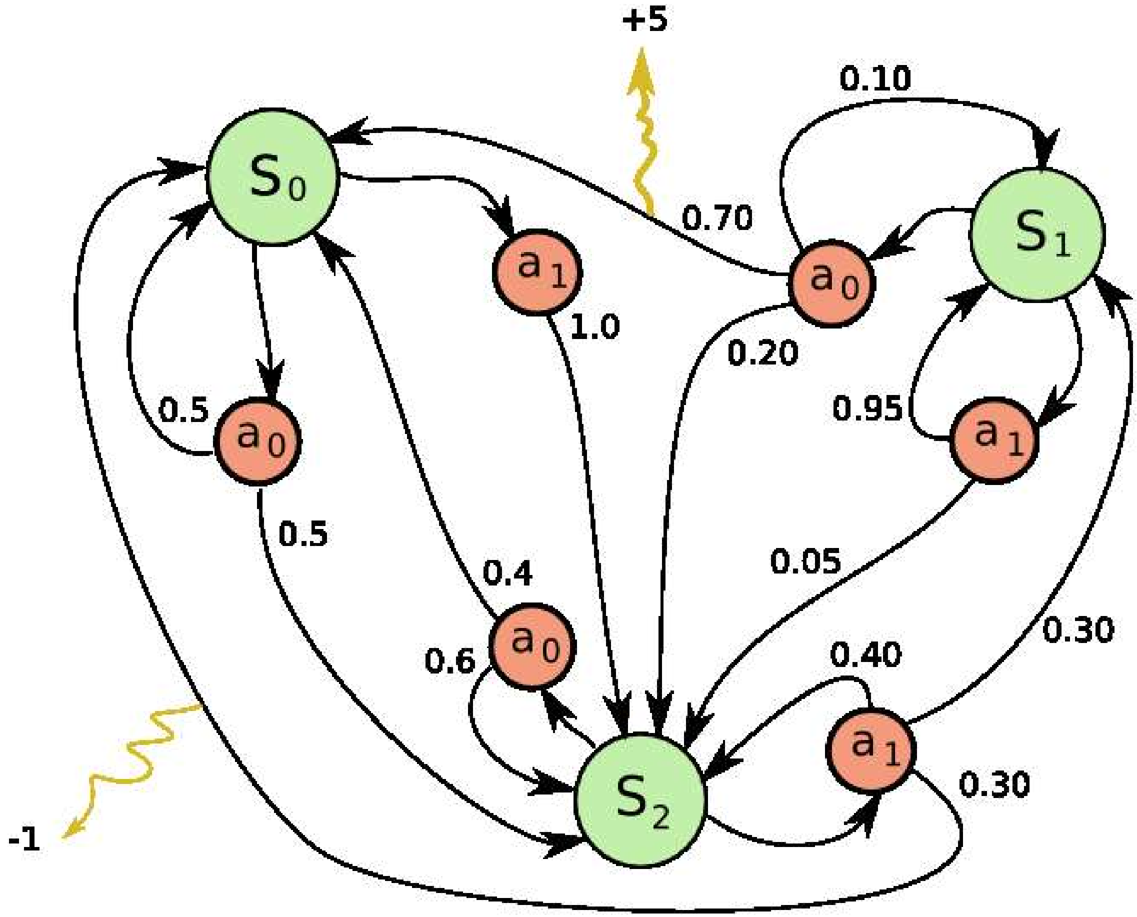
t	0.99^t	0.95^t
1	0.99	0.95
2	0.9801	0.9025
4	0.960596	0.814506
8	0.922745	0.66342
16	0.851458	0.440127
32	0.72498	0.193711
64	0.525596	0.037524

What do we need to estimate the next state and reward?

- If we only need to know the current state, this problem has the *Markov property*.

$$P(r_t = r', s_{t+1} = s' | \, s_0, a_0, r_0, \dots, r_{t-1}, s_t, a_t) =$$
$$P(r_t = r', s_{t+1} = s' | \, s_t, a_t)$$

Markov Decision Processes



Value

- The expected future reward is known as the *value*
- Two ways to compute the value:
  – The value of a state – V(s) – averaged over all possible actions in that state
  – The value of a state/action pair Q(s,a)
- Q and V are initially unknown, and learned iteratively as we gain experience

Q-learning

- Values are learned by "backing up" values from the current state to the previous one:

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

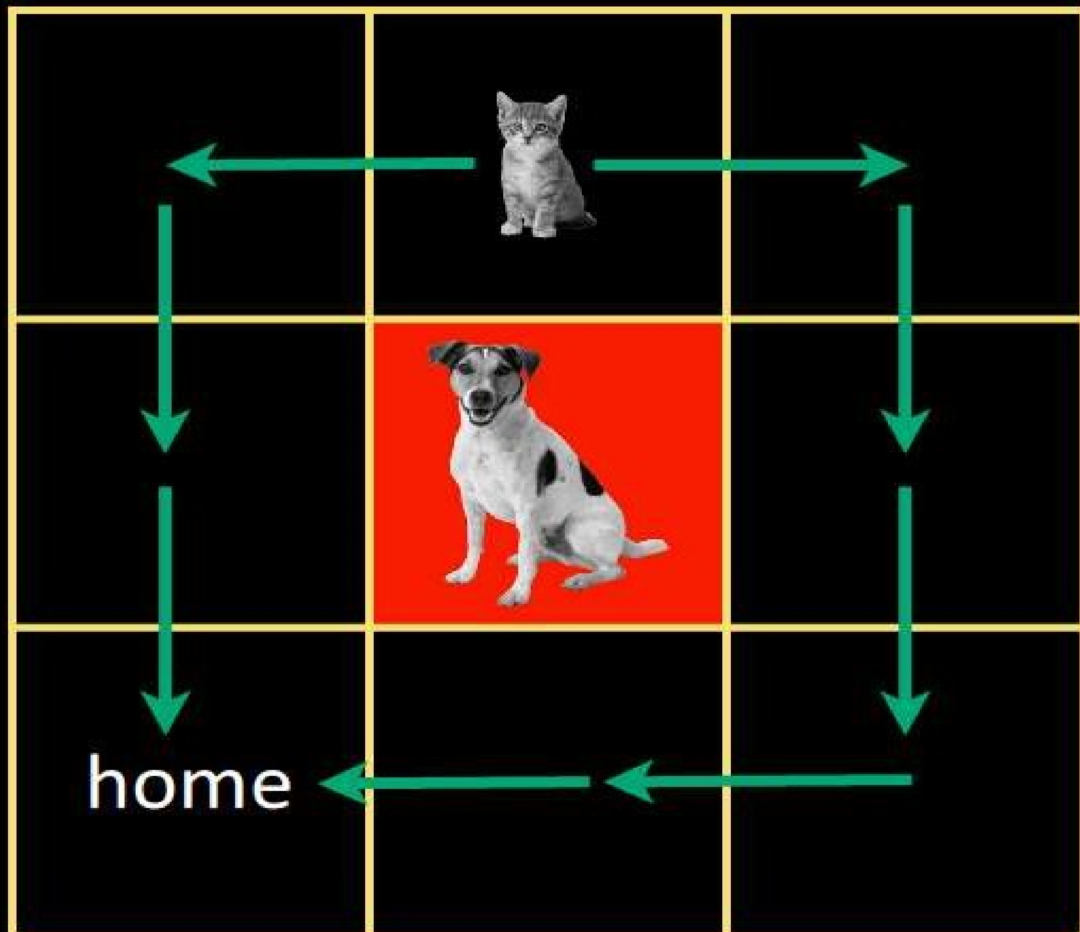
- The same can be done for v-values:

$$V(s_t) \leftarrow V(s_t) + \mu(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$$

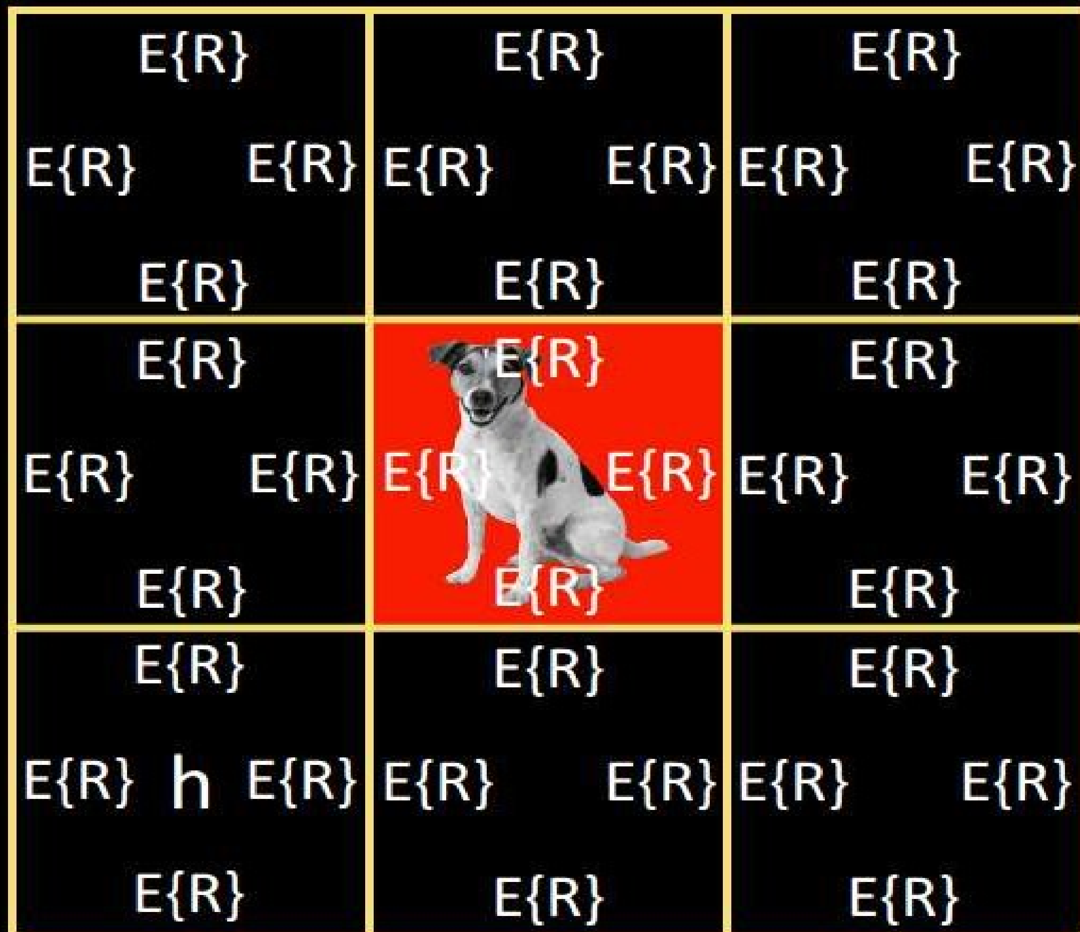
Q-learning example

- Credits: Arjun Chandra

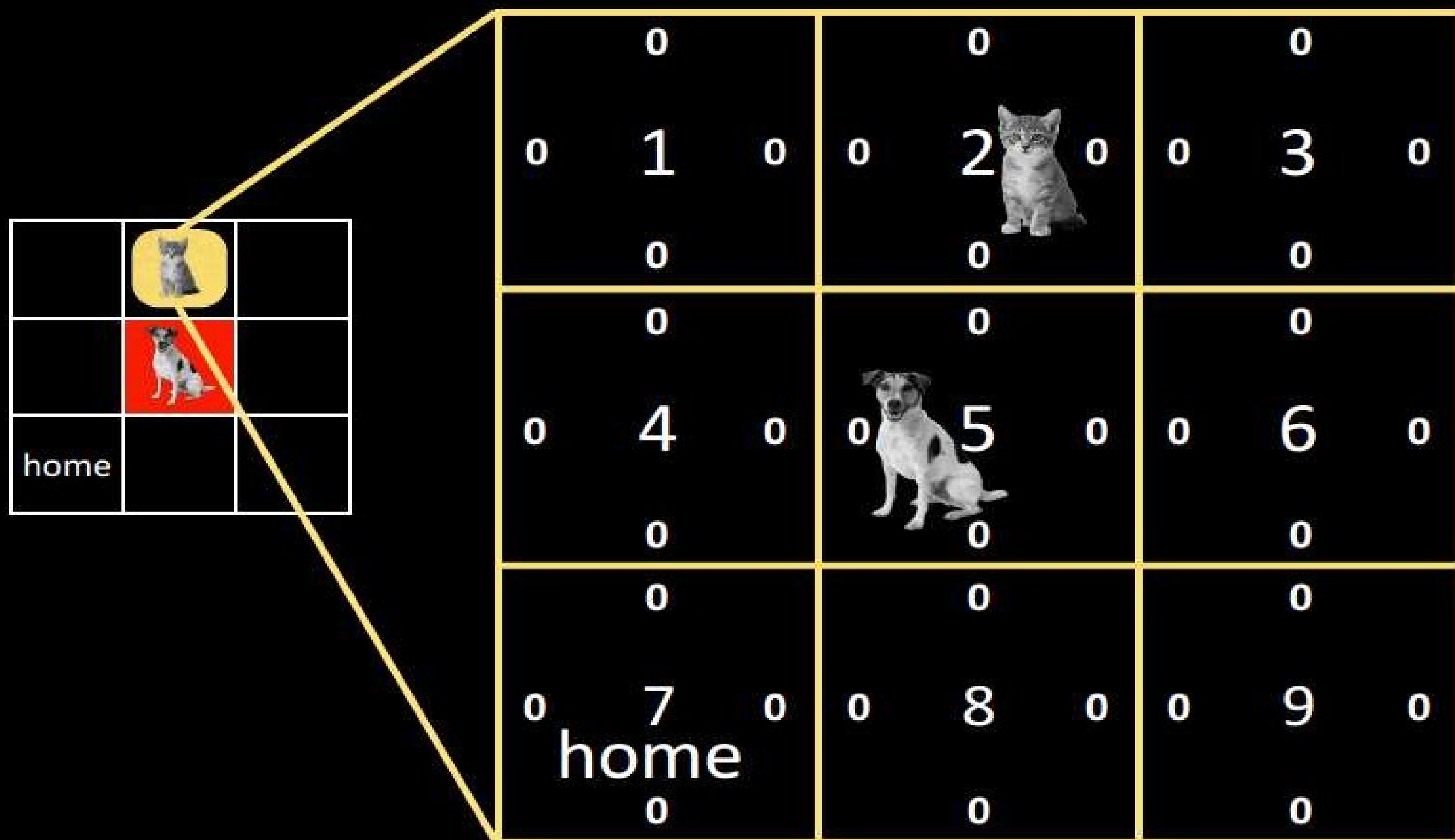
toy problem



expected long term value of taking
some action in each state,
under some action selection scheme?



our toy problem
lookup table



reward structure?

move...

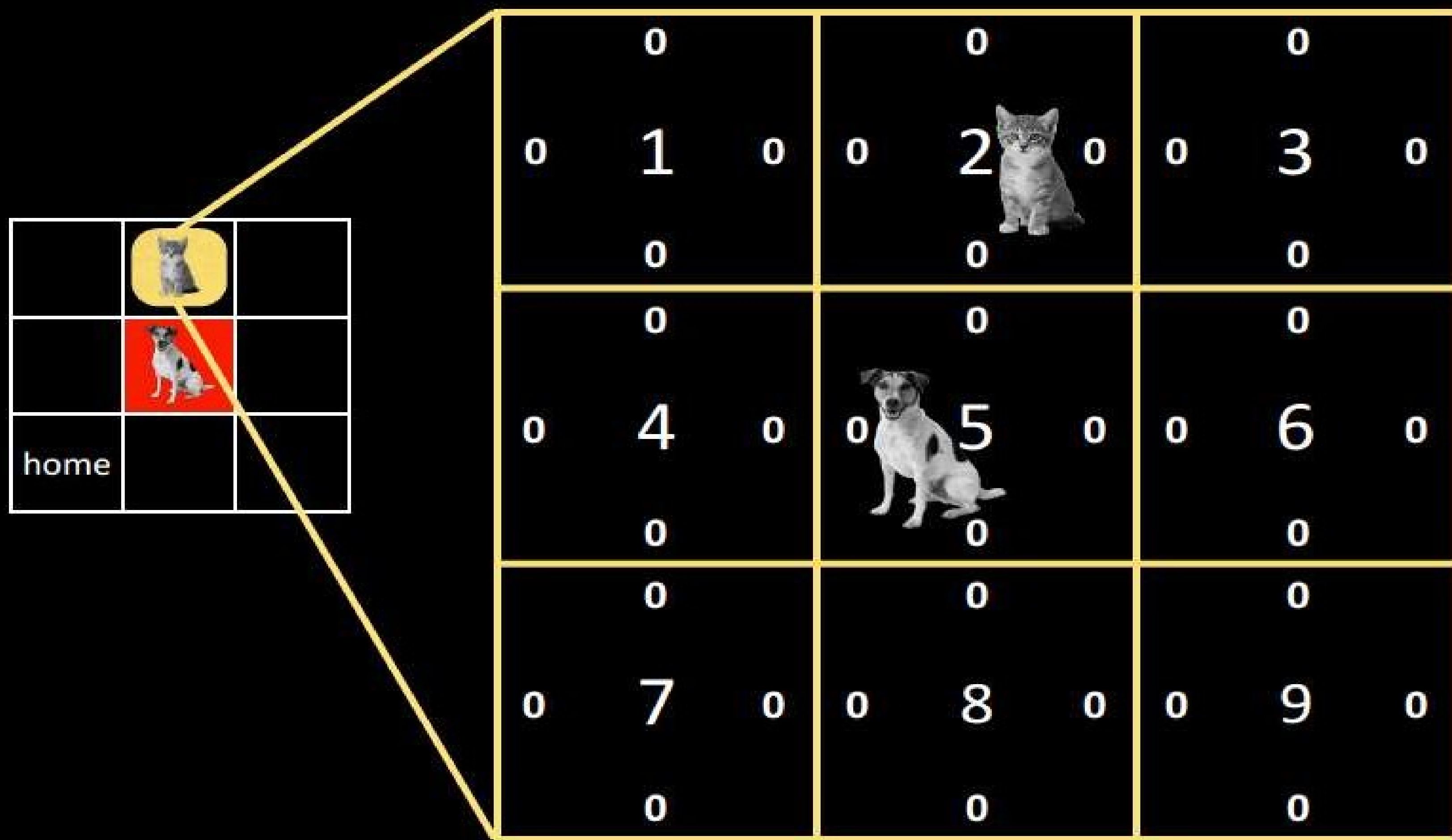
to any cell except 5 and 7:
-1

out of bounds:
-5

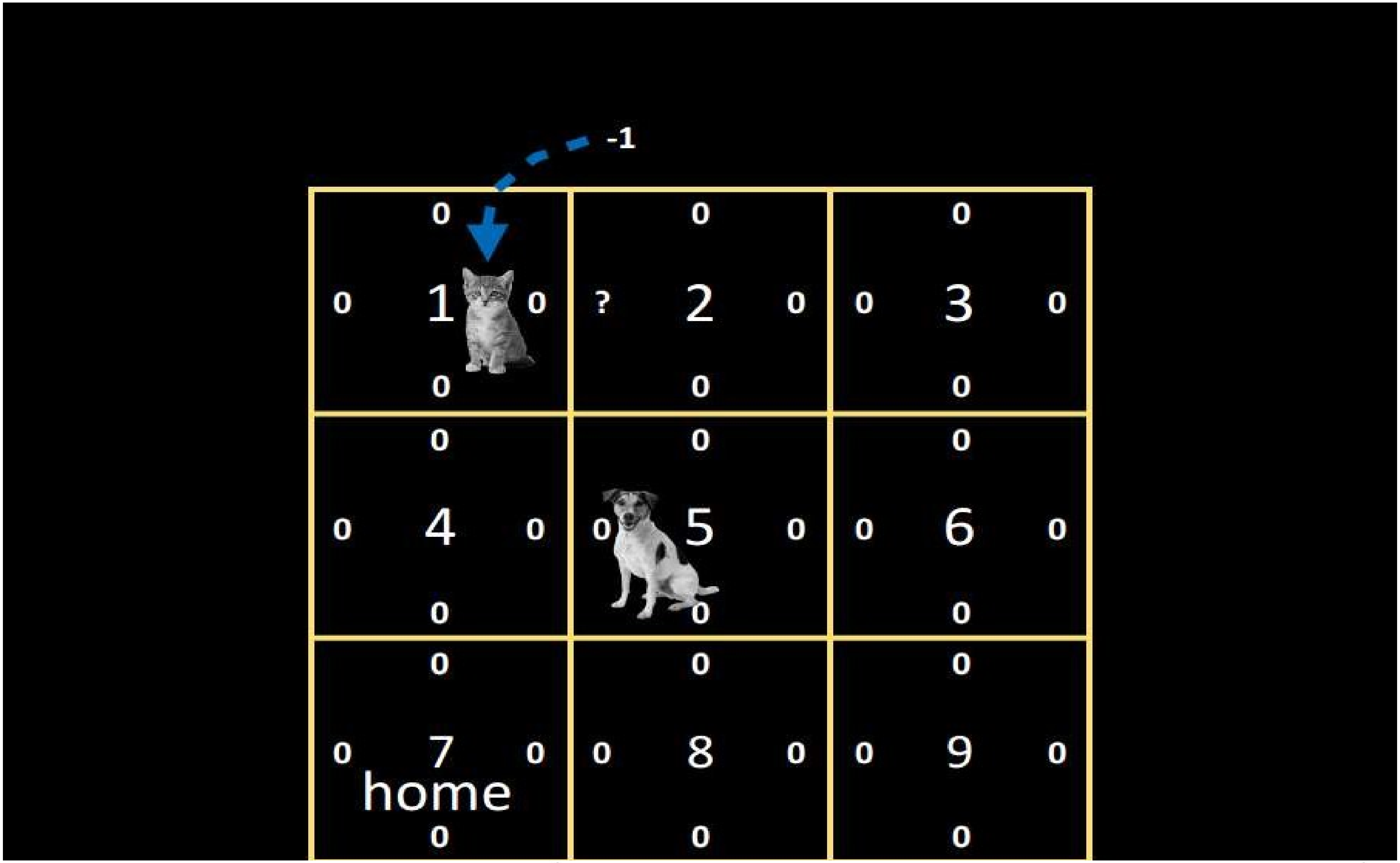
to 5:
-10

to 7/home:
10

let’s fix μ = 0.1, γ = 0.5

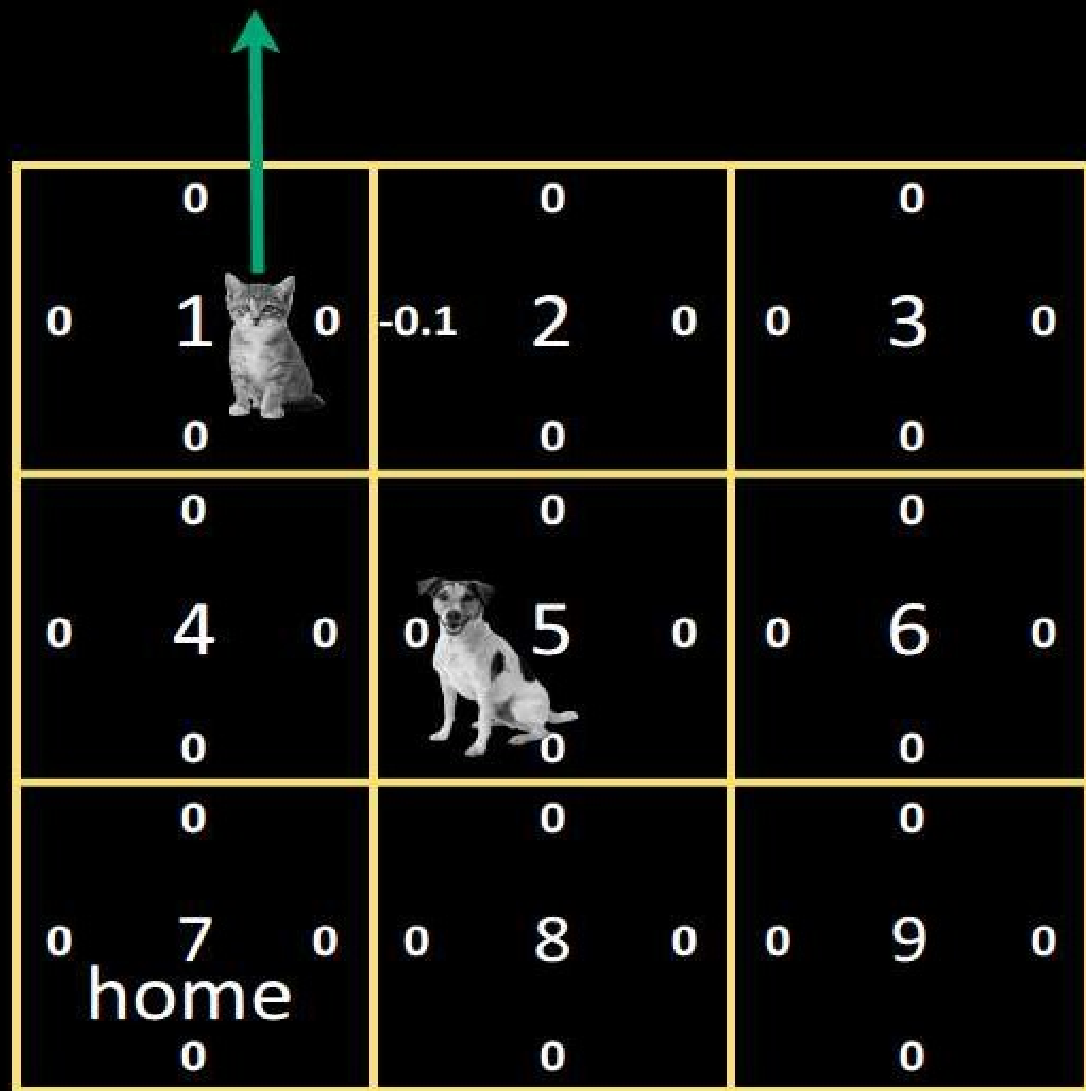


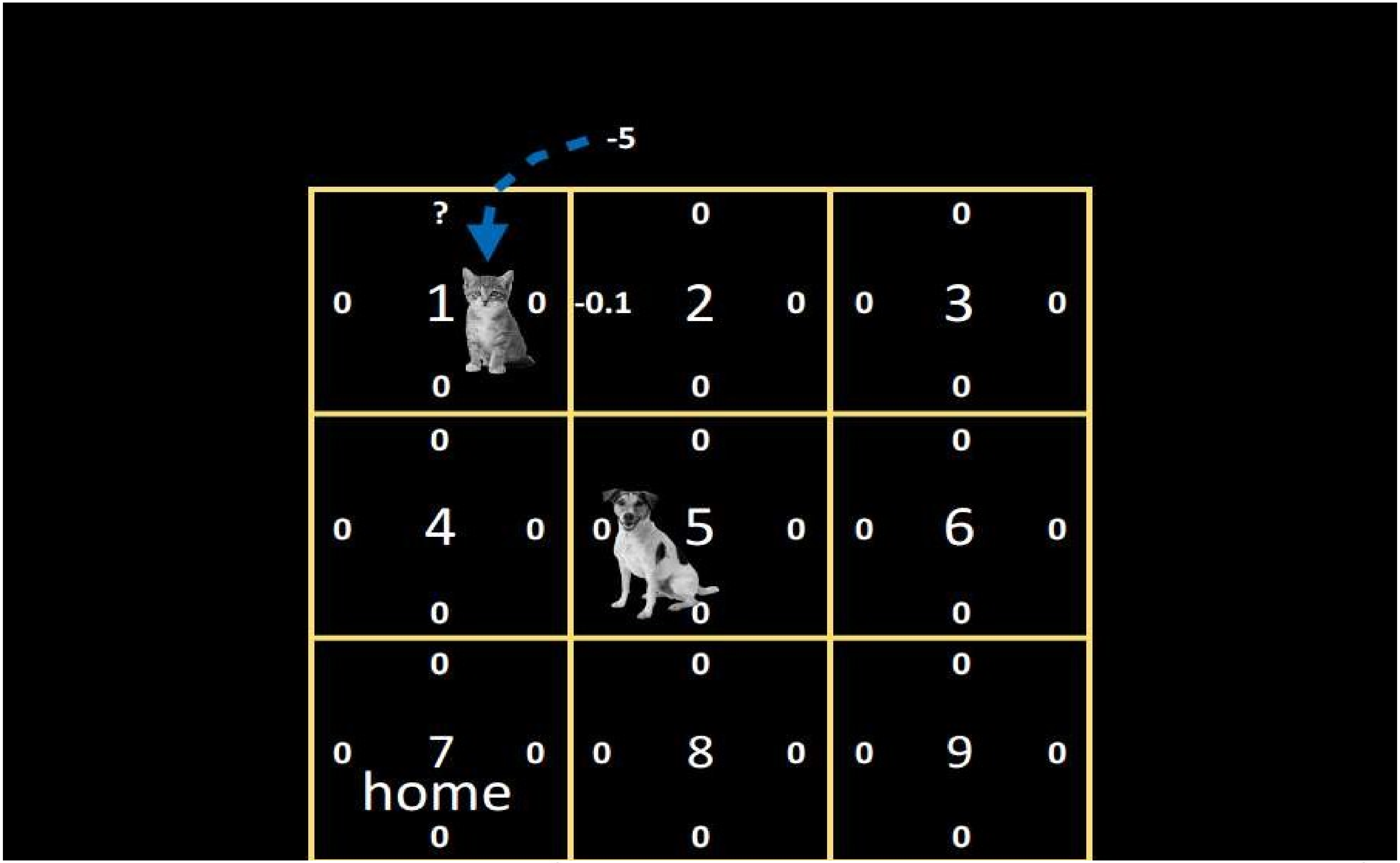
episode 1 begins...



$$Q(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t,a_t)}_{\text{old value}} \right)$$

0
0 1 0
0
0
-0.1 2 0
0
0
0 3 0
0
0
0 4 0
0
0
0 5 0
0
0
0 6 0
0
0
0 7 0
home
0
0
0 8 0
0
0
0 9 0
0

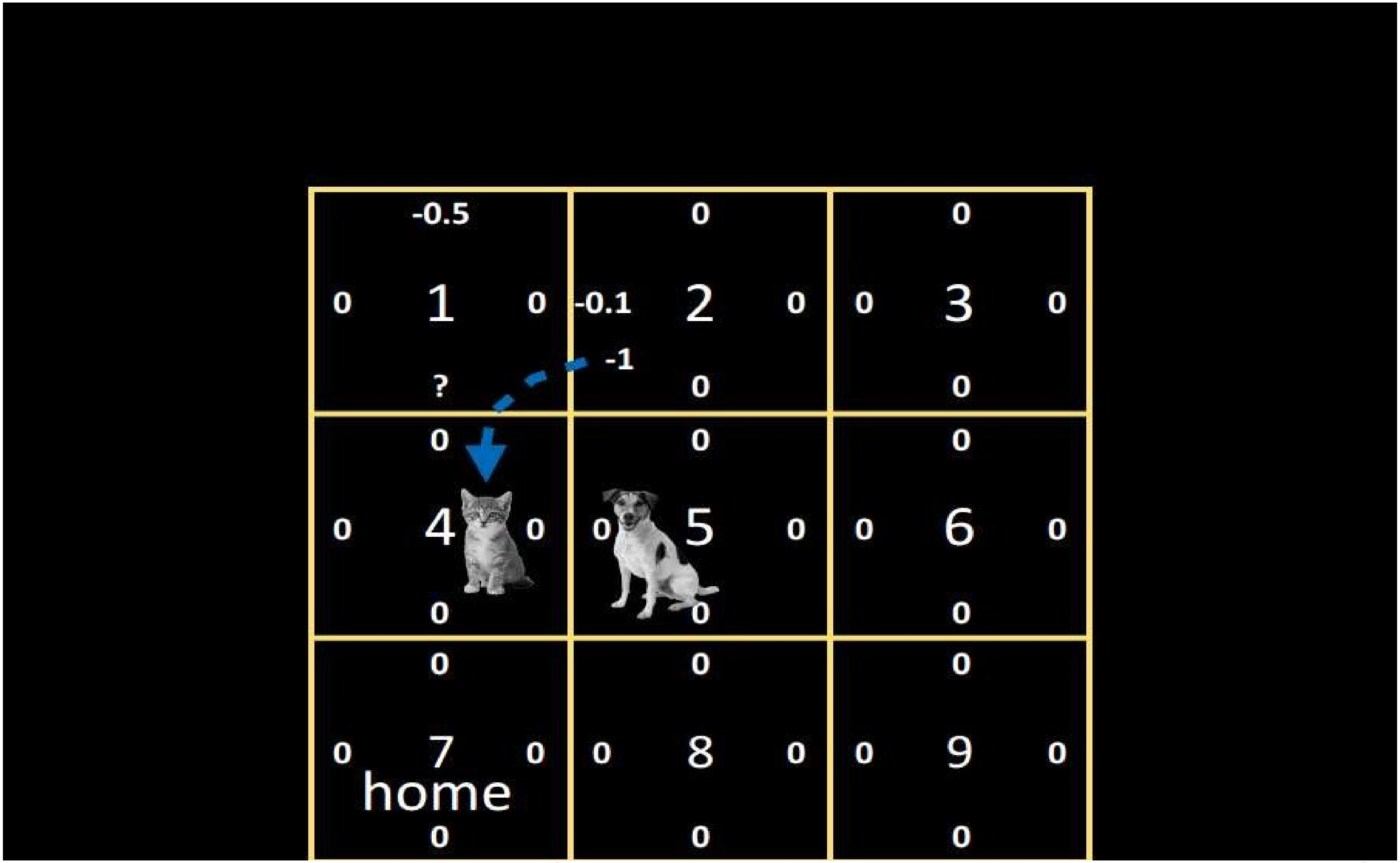

0
0 1 0
0
0
-0.1 2 0
0
0
0 3 0
0
0
0 4 0
0
0
0 5 0
0
0
0 6 0
0
0
0 7 0
home
0
0
0 8 0
0
0
0 9 0
0



$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$


-0.5
0
1
0
0
0
-0.1
2
0
0
0
0
3
0
0
0
0
4
0
0
0
0
5
0
0
0
0
6
0
0
0
0
7
0
home
0
0
0
8
0
0
0
0
9
0
0


-0.5
0
0
0
1
0
-0.1
2
0
0
3
0
0
0
0
0
0
0
0
4
0
0
5
0
0
6
0
0
0
0
0
0
0
0
7
0
0
8
0
0
9
0
home
0
0
0



$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_{a} Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

-0.5
0 1 0
-0.1
0
-0.1 2 0
0
0
0 3 0
0
0
0 4 0
0
0
0 5 0
0
0
0 6 0
0
0
0 7 0
home
0
0
0 8 0
0
0
0 9 0
0

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

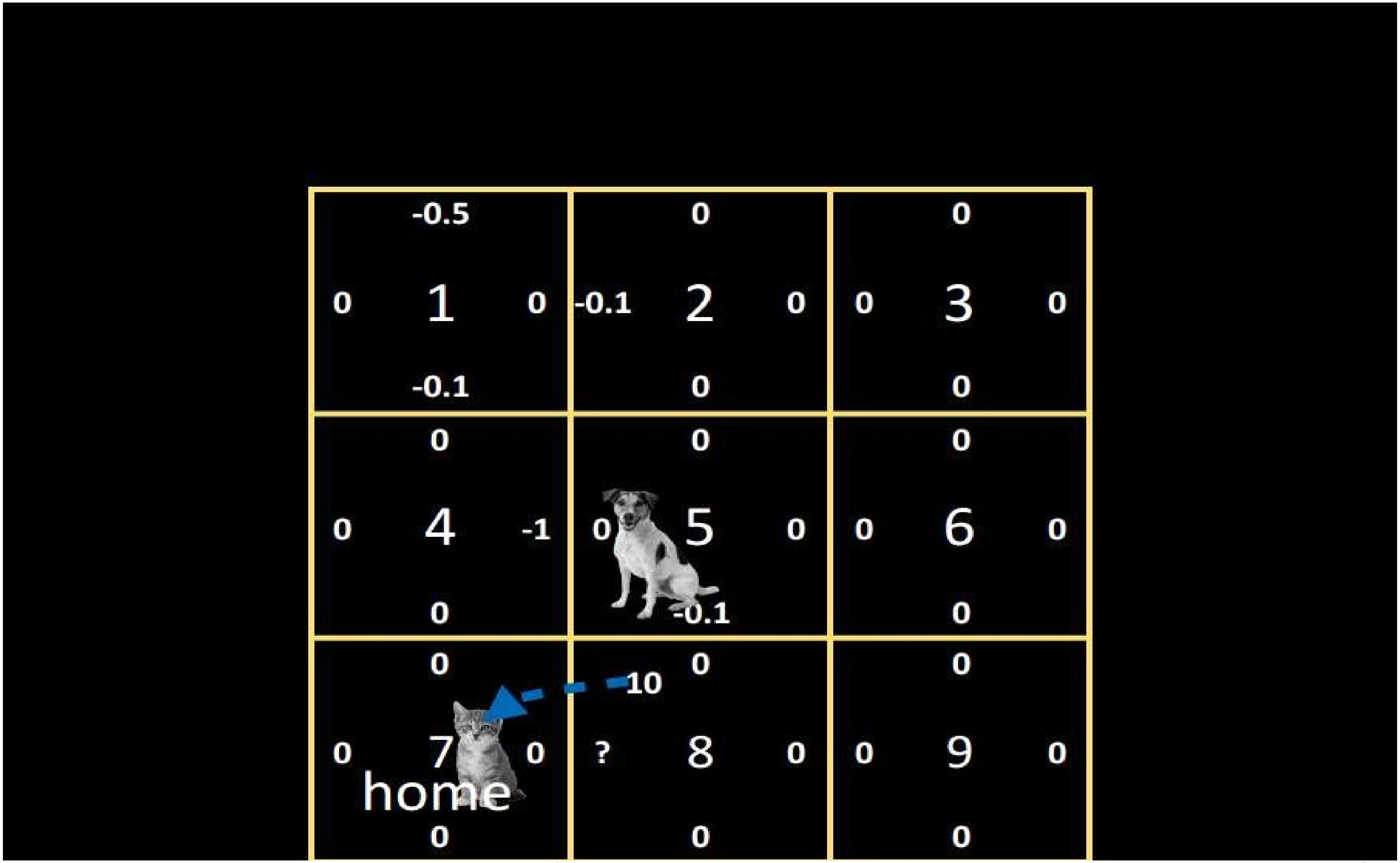
-0.5
0 1 0
-0.1
0
-0.1 2 0
0
0
0 3 0
0
0
0 4 -1
0
0
0 5 0
0
0
0 6 0
0
0
0 7 0
home
0
0
0 8 0
0
0
0 9 0
0

-0.5
0 1 0
-0.1
0
-0.1 2 0
0
0
0 3 0
0
0
0 4 -1
0
0
0 5 0
0
0
0 6 0
0
0
0 7 0
home
0
0
0 8 0
0
0
0 9 0
0

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$


-0.5
0 1 0
-0.1
0
-0.1 2 0
0
0
0 3 0
0
0
0 4 -1
0
0
0 5 0
-0.1
0
0 6 0
0
0
0 7 0
home
0
0
0 8 0
0
0
0 9 0
0


-0.5
0 1 0
-0.1
0
-0.1 2 0
0
0
0 3 0
0
0
0 4 -1
0
0
0 5 0
-0.1
0
0 6 0
0
0
0 7 0
home
0
0
0 8 0
0
0
0 9 0
0

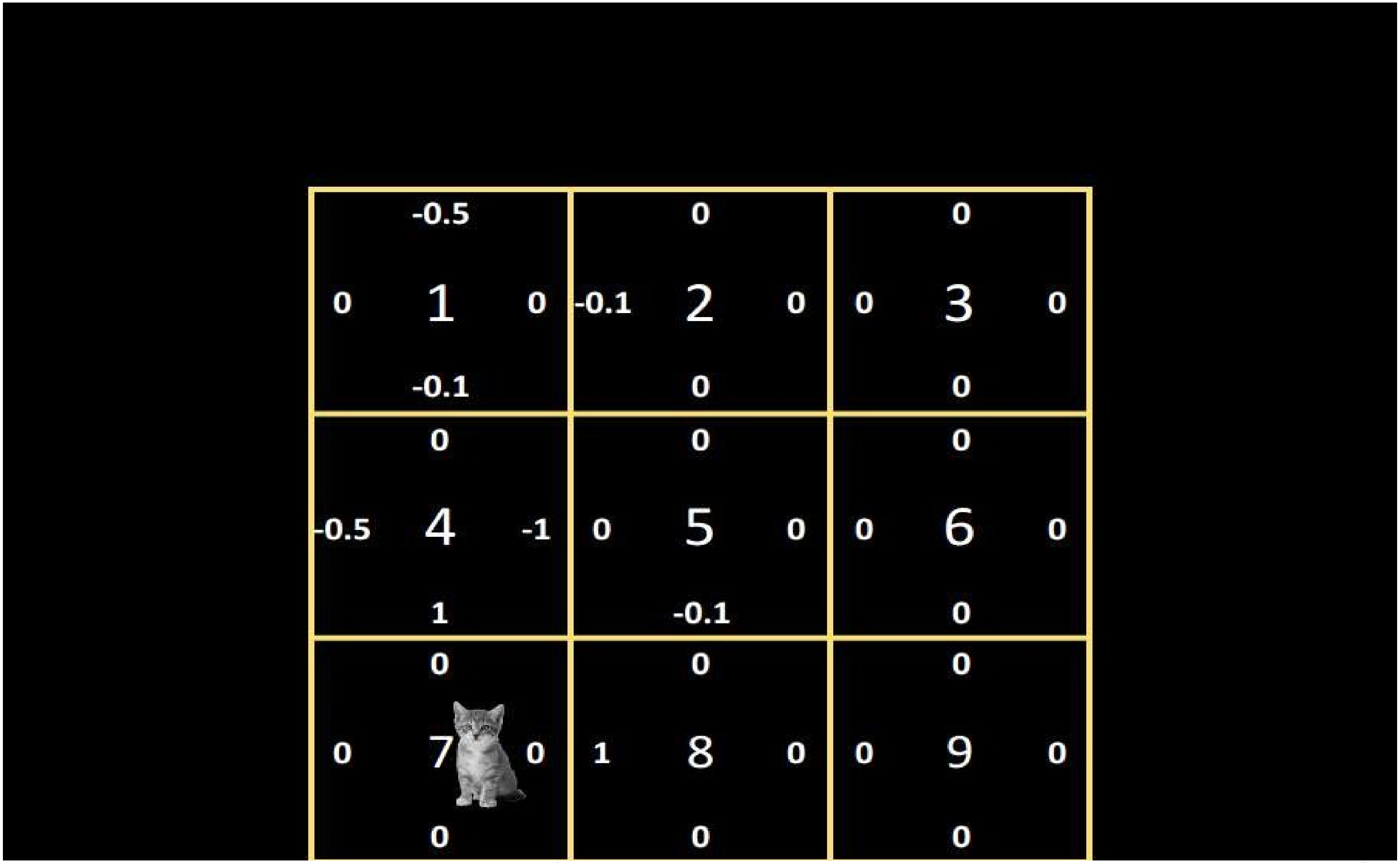


$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

let's work out the next episode, starting at state 4

go WEST and then SOUTH

how does the table change?

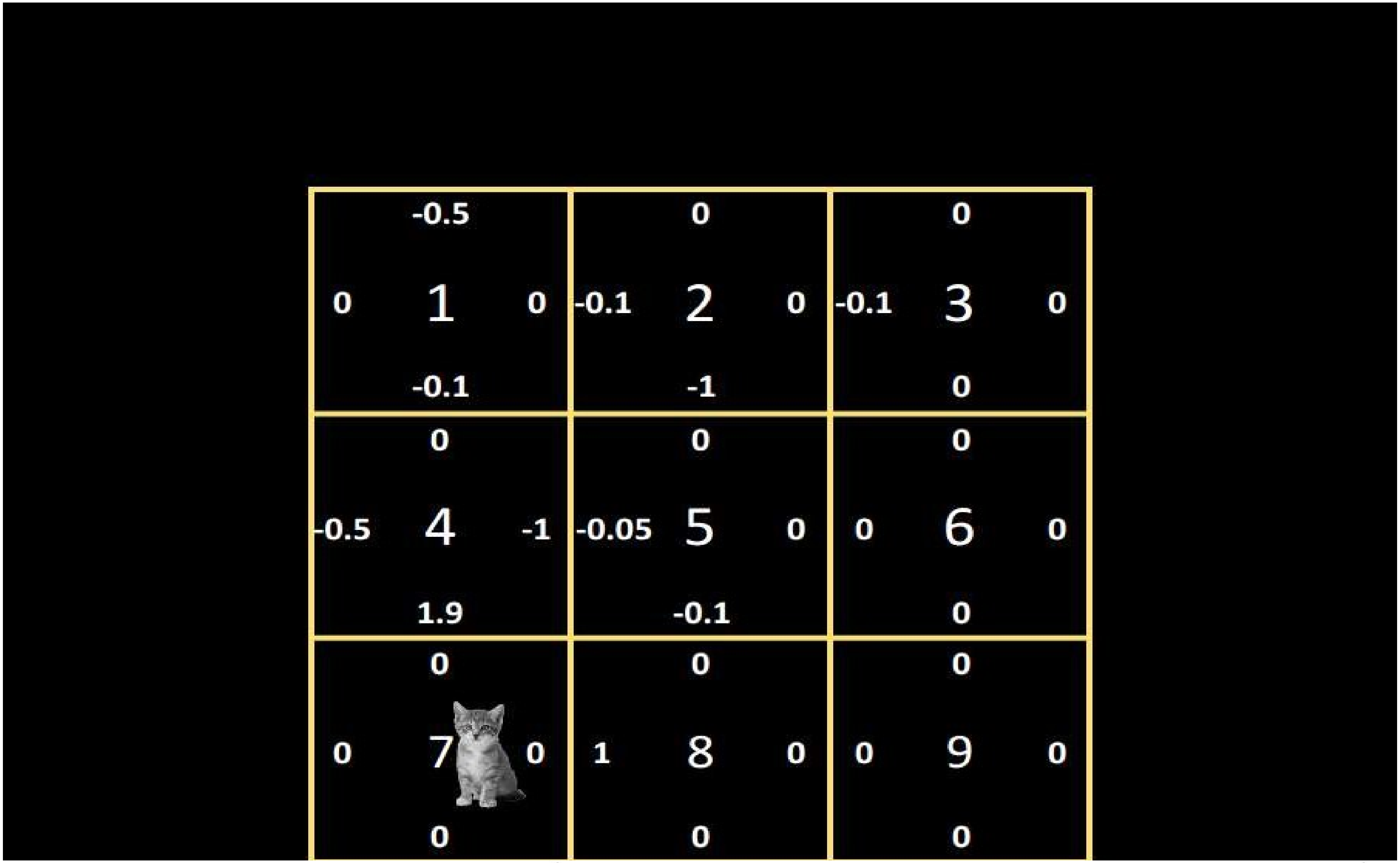


$$Q(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t,a_t)}_{\text{old value}} \right)$$

and the next episode, starting at state 3

go WEST -> SOUTH -> WEST -> SOUTH

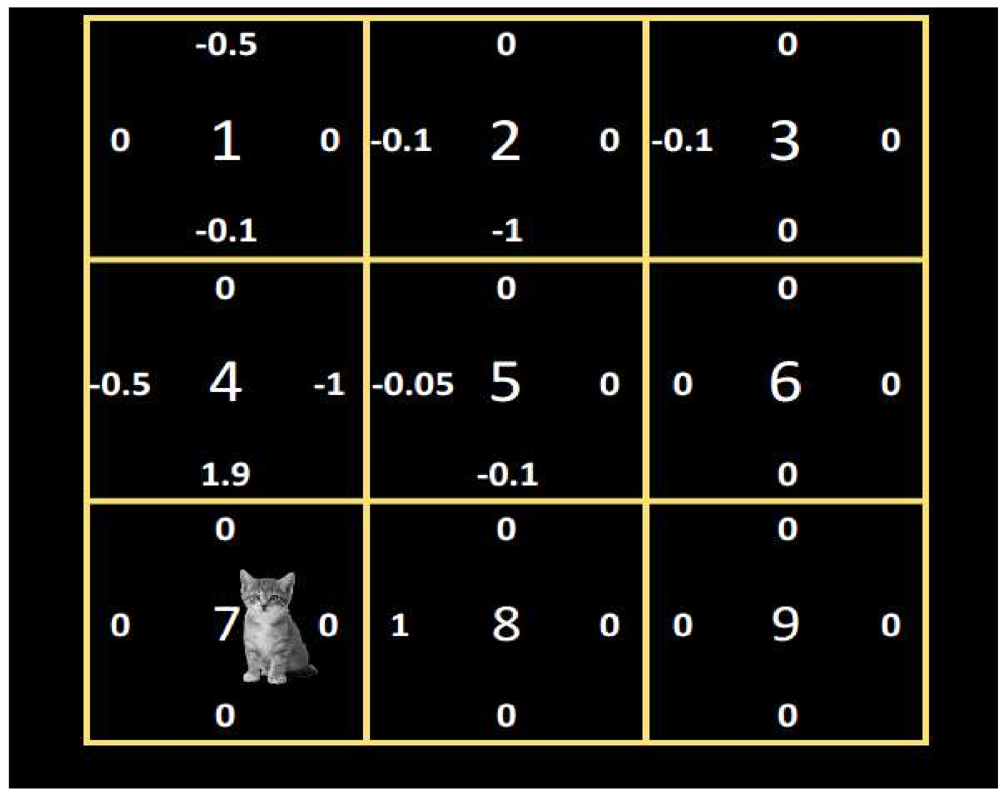
how does the table change?



$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

Action selection

- Estimate the *value* of each action: $Q_{s,t}(a)$
- Decide whether to:
  - Explore, or
  - exploit



Action selection

- The function deciding which action to take in each state is called the policy, π. Examples:
  - Greedy: Always choose most valuable action
  - ϵ-greedy: Greedy, except small probability (ϵ) of choosing the action at random
- The q-learning we just saw is an example of *off-policy learning*:

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

On-policy vs off-policy learning

- Q-learning (off-policy):

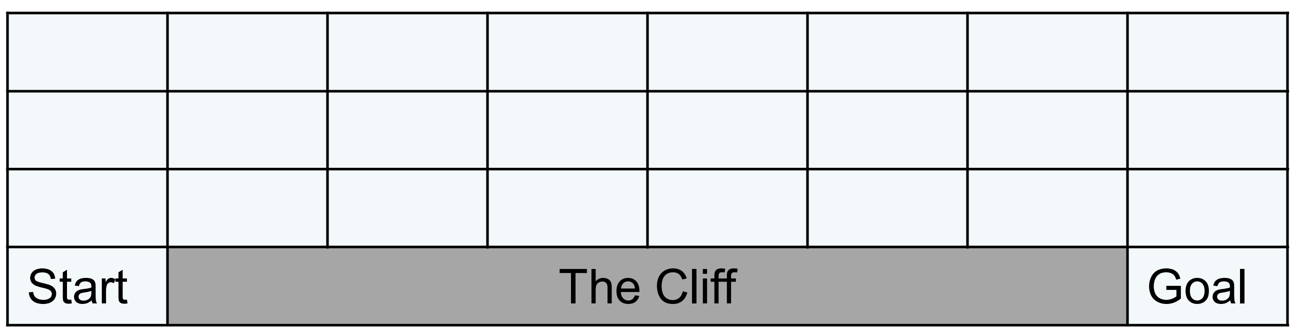
$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

- Sarsa (on-policy):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \mu[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

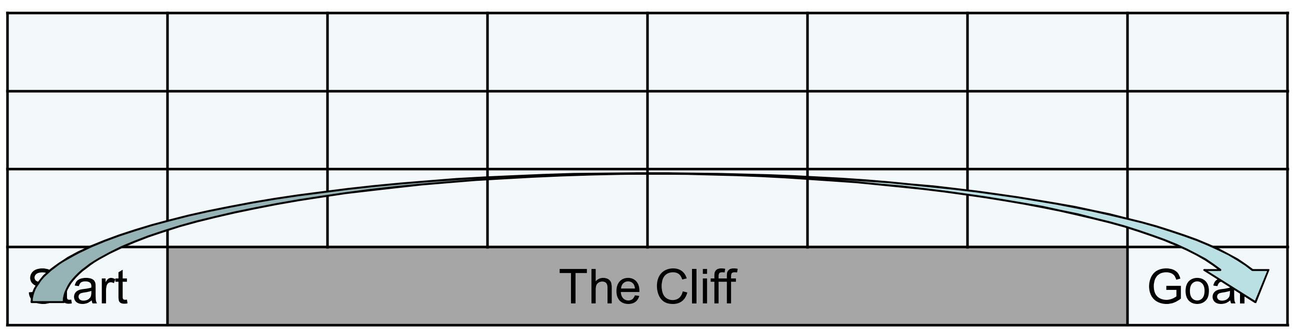
On-policy vs off-policy learning

- Reward structure: Each move: -1. Move to cliff: -100.
- Policy: 90% chance of choosing best action (exploit). 10% chance of choosing random action (explore).



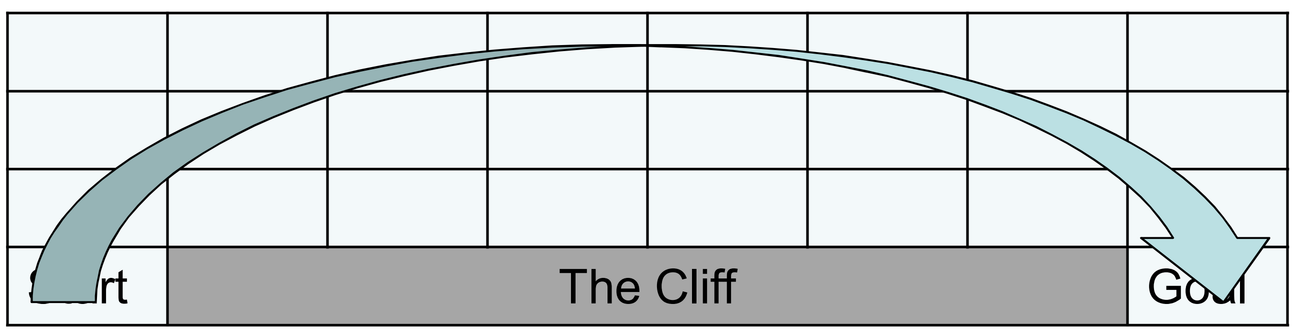
On-policy vs off-policy learning: Q-learning

- Always assumes optimal action -> does not visit cliff often while learning. Therefore, does not learn that cliff is dangerous.
- Resulting path is efficient, but risky.



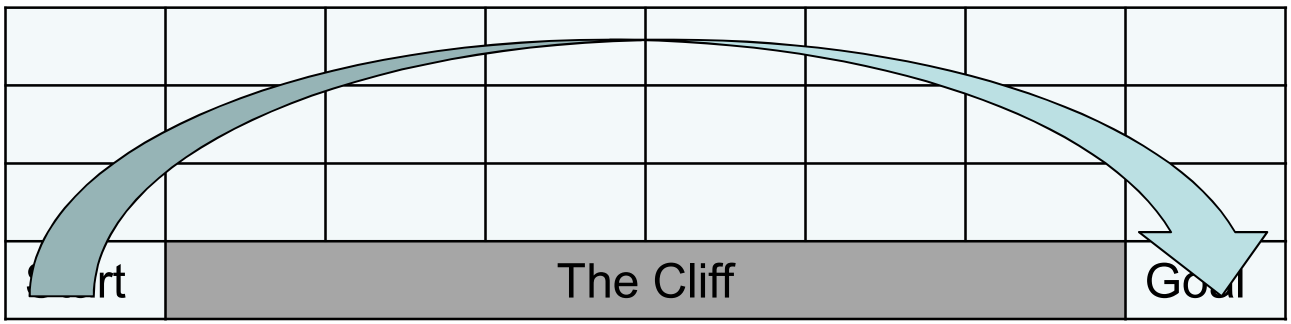
On-policy vs off-policy learning: sarsa

- During learning, we more frequently end up outside the cliff (due to the 10% chance of exploring in our policy).
- That info propagates to all states, generating a safer plan.



Which plan is better?

- sarsa (on-policy):



- Q-learning (off-policy):