Reinforcement Learning

Rob Platt
Northeastern University

Some images and slides are used from:
AIMA
CS188 UC Berkeley

Reinforcement Learning (RL)

Previous session discussed sequential decision making problems where the transition model and reward function were known

In many problems, the model and reward are *not known* in advance

Agent must learn how to act through *experience* with the world

This session discusses *reinforcement learning* (*RL*) where an agent receives a reinforcement signal

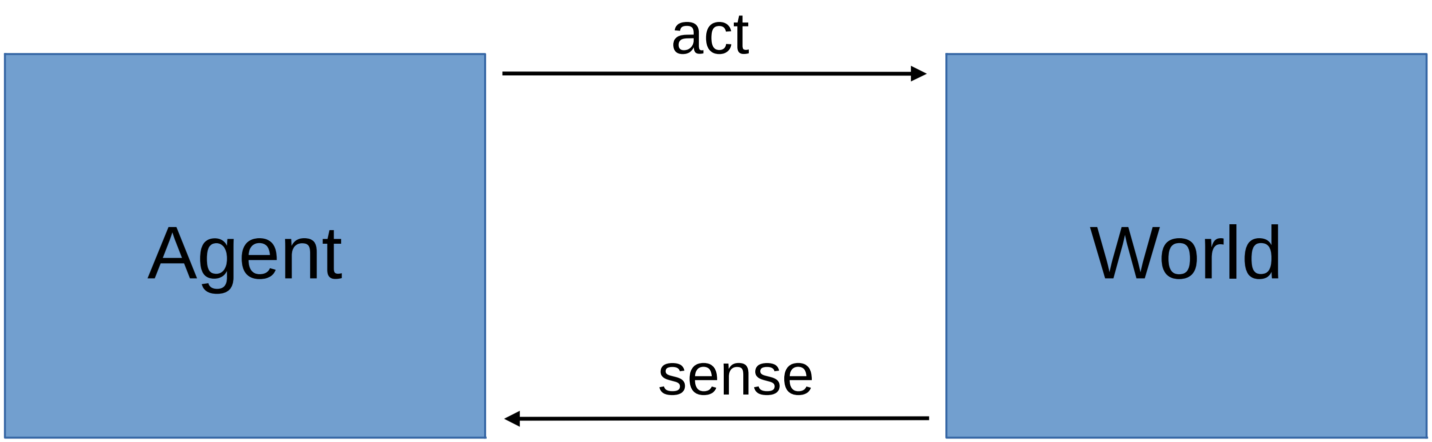
Challenges in RL

Exploration of the world must be balanced with *exploitation* of knowledge gained through experience

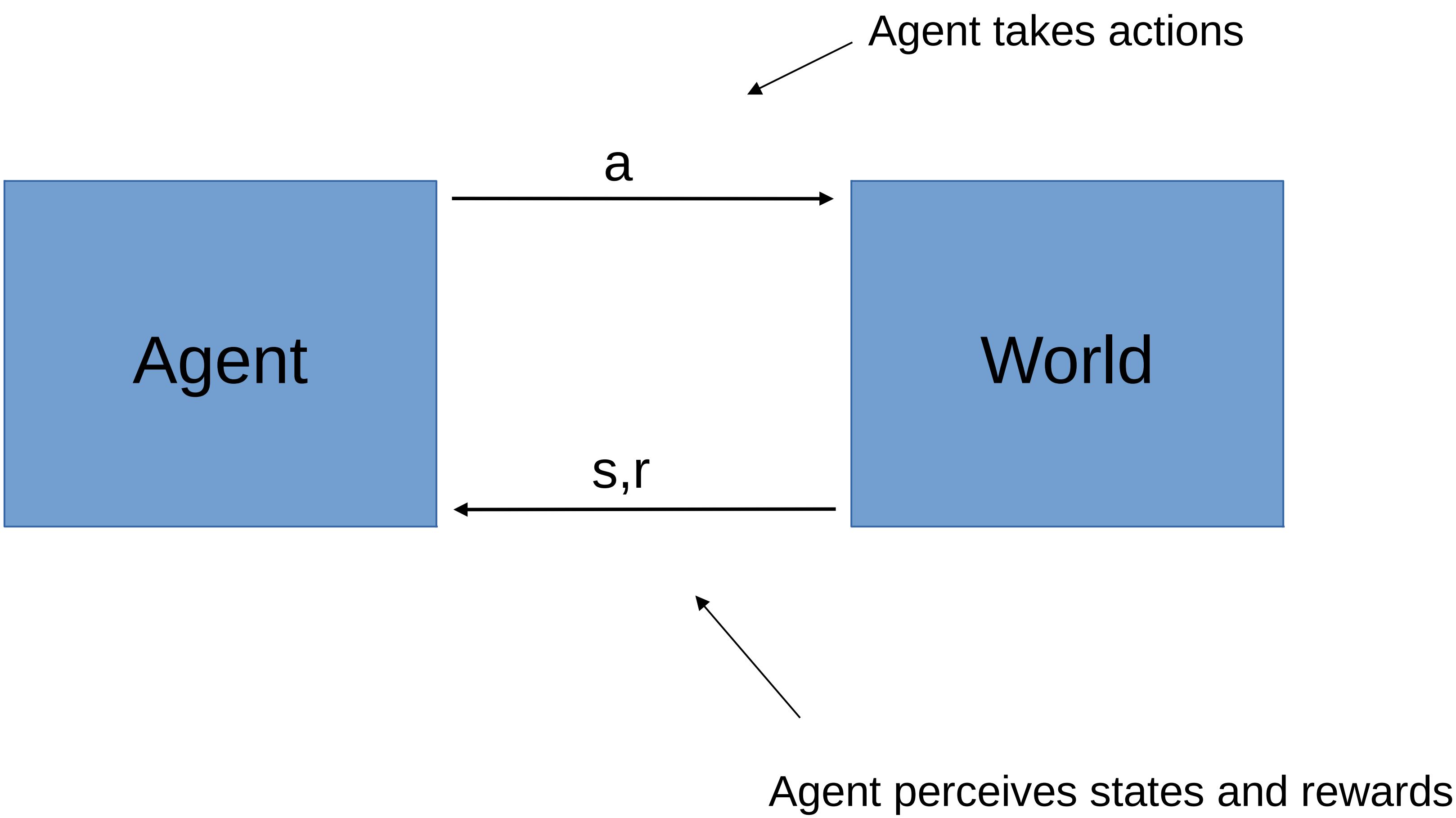
Reward may be received long after the important choices have been made, so *credit* must be assigned to earlier decisions

Must *generalize* from limited experience

Conception of agent



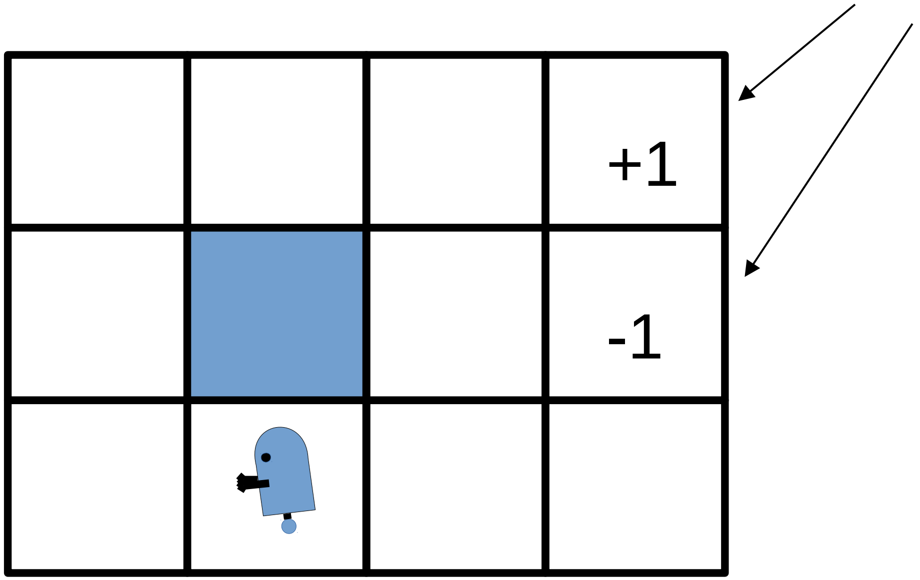
RL conception of agent



Transition model and reward function are initially unknown to the agent! – value iteration assumed knowledge of these two things...

Value iteration

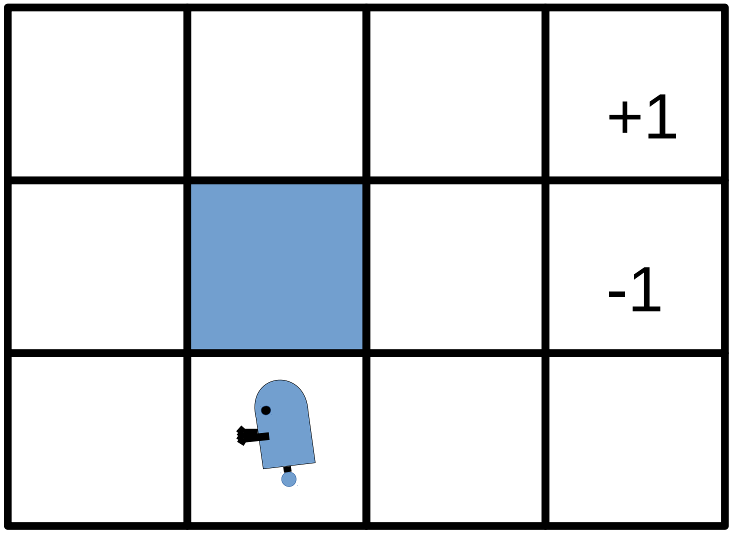
We know the reward function



We know the probabilities of moving in each direction when an action is executed

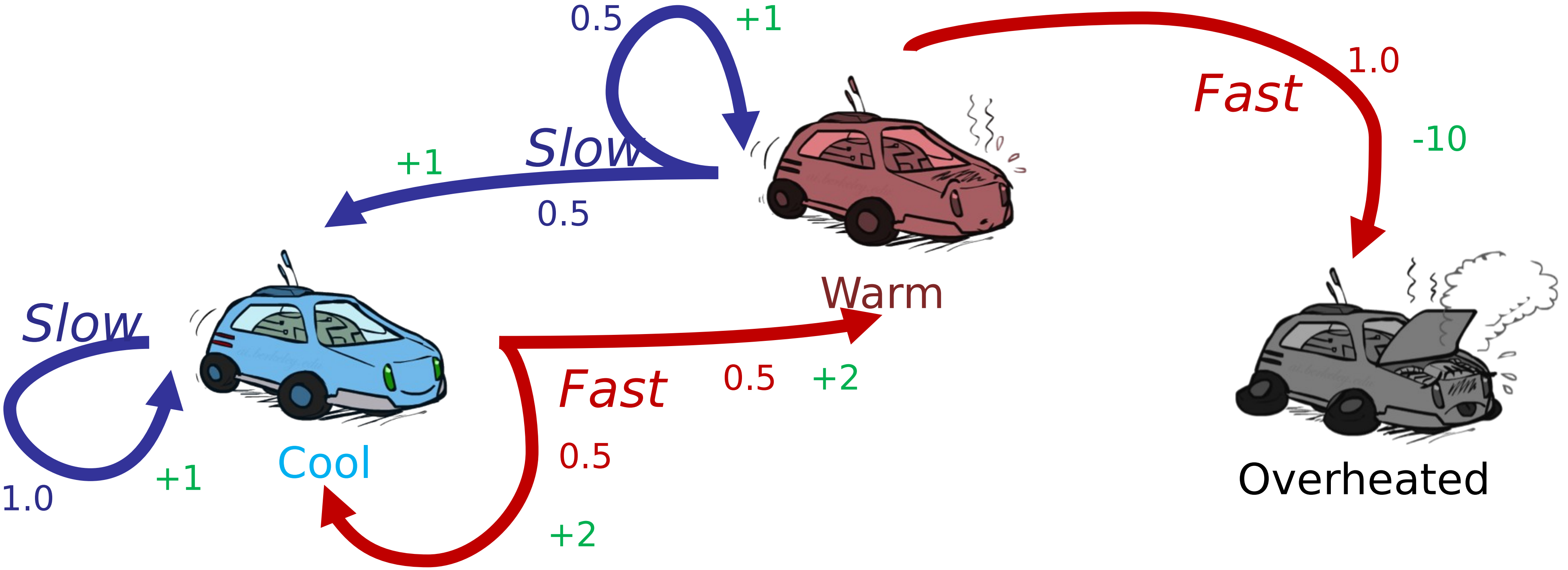
Value iteration

We know the reward function



We know the probabilities of moving in each direction when an action is executed

Value iteration vs RL



RL still assumes that we have an MDP

Value iteration vs RL

RL still assumes that we have an MDP
– we know S and A
– we still want to calculate an optimal policy

<u>BUT:</u>
– we do not know T or R
– we need to figure our T and R by trying out actions and seeing what happens

Example: Learning to Walk

Initial

A Learning Trial

After Learning
[1K Trials]

[Kohl and Stone, ICRA 2004]

Example: Learning to Walk

Initial

[Kohl and Stone, ICRA 2004]

Example: Learning to Walk

Training

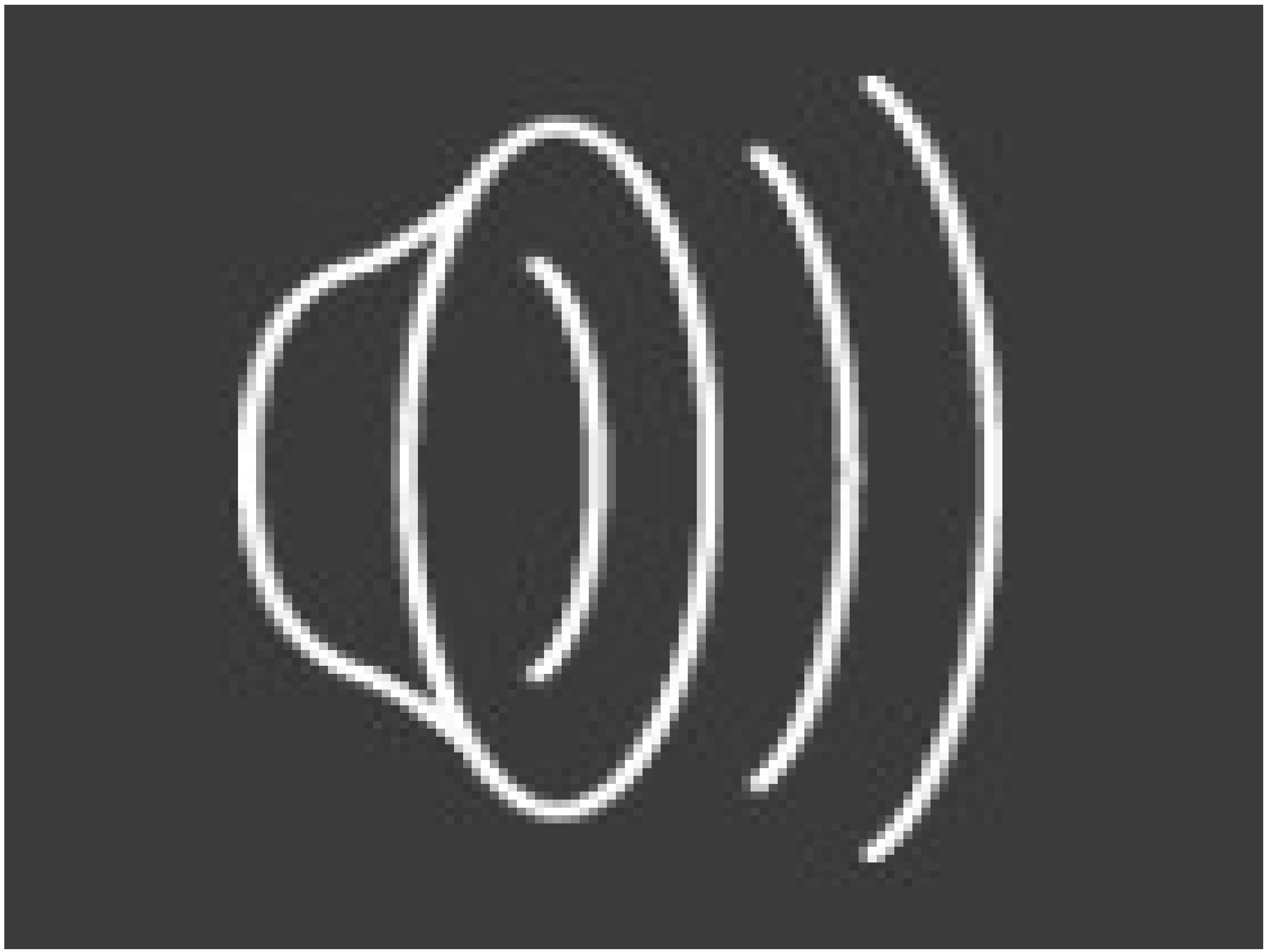
[Kohl and Stone, ICRA 2004]

Example: Learning to Walk

Finished

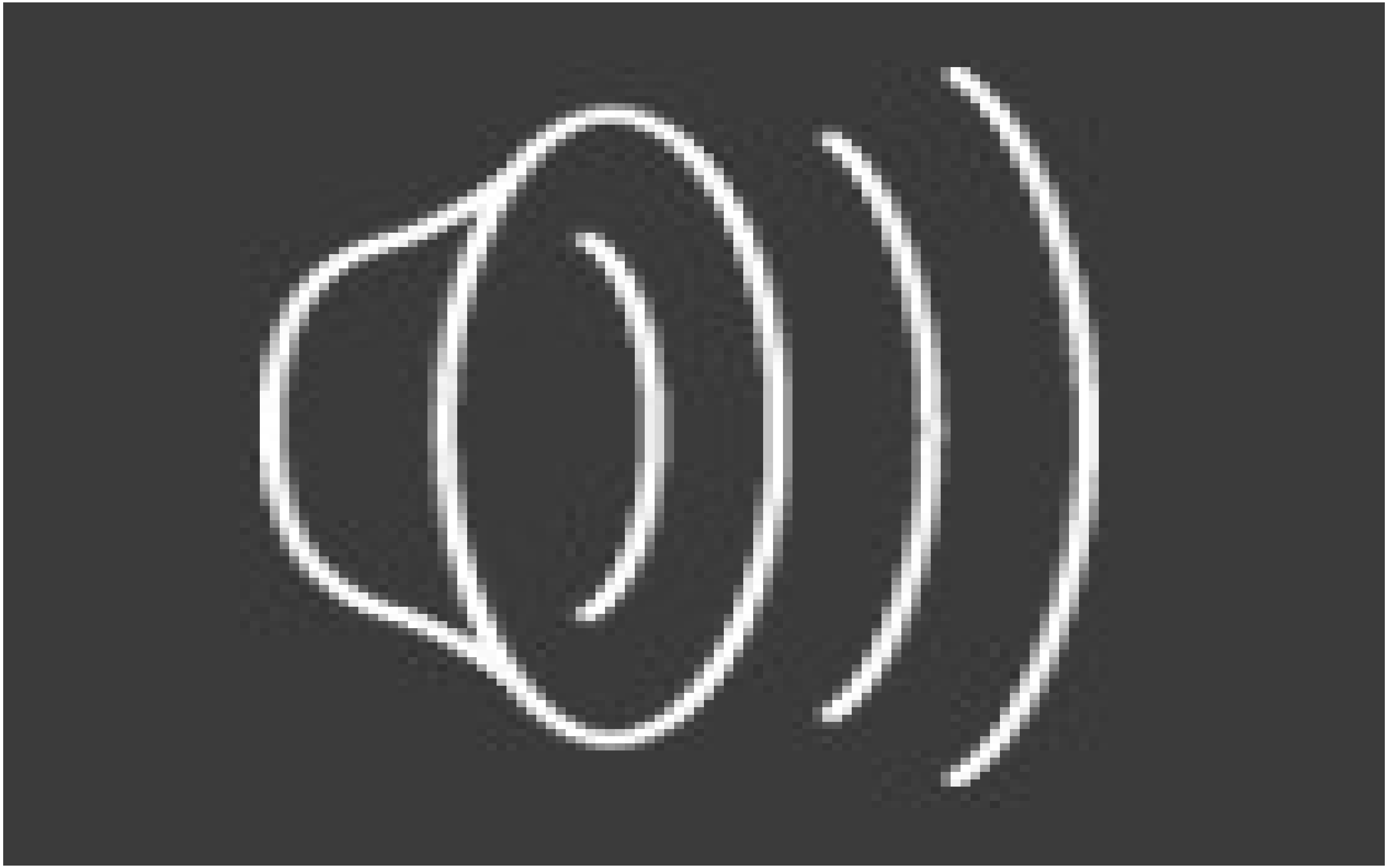
[Kohl and Stone, ICRA 2004]

Toddler robot uses RL to learn to walk



Tedrake et al., 2005

The next homework assignment!



Model-based RL

1. estimate T, R by averaging experiences

2. solve for policy in MDP (e.g., value iteration)

a. choose an exploration policy
   – policy that enables agent to explore all relevant states

b. follow policy for a while

c. estimate T and R

Model-based RL

1. estimate T, R by averaging experiences

2. solve for policy in MDP (e.g., value iteration)

a. choose an exploration policy
   – policy that enables agent to explore all relevant states

b. follow policy for a while

c. estimate T and R

$N_{s,a,s'} \equiv$ Number of times agent reached *s'* by taking *a* from *s*

$R_{s,a,s'} \equiv$ Set of rewards obtained when reaching *s'* by taking *a* from *s*

$$T(s, a, s') \approx \frac{N_{s,a,s'}}{\sum_{s'} N_{s,a,s'}} \qquad R(s, a, s') \approx \frac{1}{N_{s,a,s'}} R_{s,a,s'}$$

Model-based RL

1. estimate T, R by averaging experiences

a. choose an exploration policy – policy that enables agent to explore all relevant states

a while

2. solve for p
(e.g., value ite

R

What is a downside of this approach?

$N_{s,a,s'} \equiv$

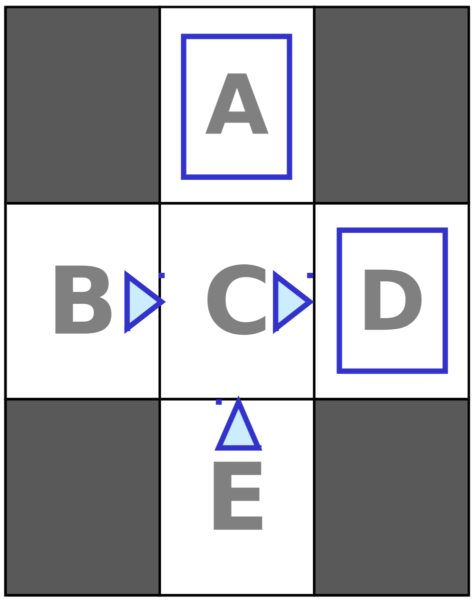
s

$R_{s,a,s'} \equiv$

a from s

$$T(s,a,s') \approx \frac{N_{s,a,s'}}{\sum_{s'} N_{s,a,s'}} \qquad R(s,a,s') \approx \frac{1}{N_{s,a,s'}} R_{s,a,s'}$$

Example: Model-based RL



Blue arrows denote policy

<u>States</u>: a,b,c,d,e

<u>Actions</u>: l, r, u, d

<u>Observations</u>:

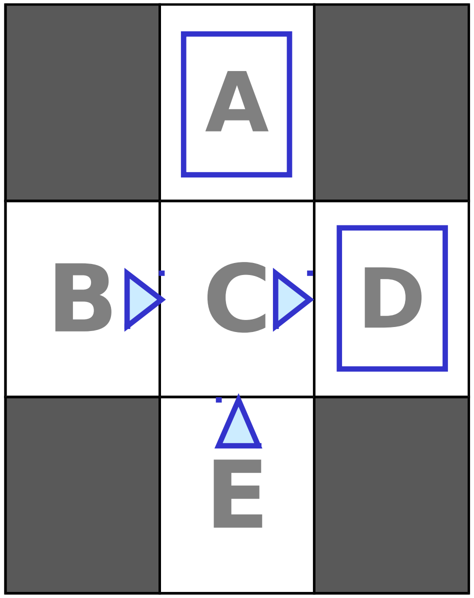
1. b,r,c
2. e,u,c
3. c,r,d
4. b,r,a
5. b,r,c

6, e,u,c

7, e,u,c

?

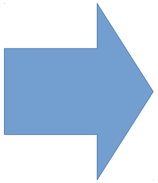
Example: Model-based RL



Blue arrows denote policy

States: a,b,c,d,e
Actions: l, r, u, d

Observations:
1. b,r,c
2. e,u,c
3. c,r,d
4. b,r,a
5. b,r,c
6, e,u,c
7, e,u,c



Estimates:

P(c|e,u) = 1
P(c|b,r) = 0.66
P(a|b,r) = 0.33
P(d|c,r) = 1

Model-based vs Model-free

Suppose you want to calculate average age in this class room

Method 1: $$\mathbb{E}(a) = \sum_a P(a)a$$

where: $$P(a) = \frac{\text{num people of age } a}{\text{total num people}}$$

Method 2: $$\mathbb{E}(a) \approx \sum_{i=1}^{n} a_i$$

where: a_i is a the age of a randomly sampled person

Model-based vs Model-free

Suppose you want to calculate average age in this class room

Model based (why?)

Method 1: $\mathbb{E}(a) = \sum_a P(a)a$

where: $P(a) = \frac{\text{num people of age } a}{\text{total num people}}$

Model free (why?)

Method 2: $\mathbb{E}(a) \approx \sum_{i=1}^{n} a_i$

where: a_i is a the age of a randomly sampled person

Model-free estimate of the value function

Remember this equation?

$$V_{i+1}(s) = \max_a \sum_{s'} T(s, a, s') \left[r(s, a) + \gamma V_i(s') \right]$$

Is this model-based or model-free?

Model-free estimate of the value function

Remember this equation?

$$V_{i+1}(s) = \max_a \sum_{s'} T(s, a, s') \left[r(s, a) + \gamma V_i(s') \right]$$

Is this model-based or model-free?

How do you make it model-free?

Model-free estimate of the value function

Remember this equation?

$$V_{i+1}(s) = \max_a \sum_{s'} T(s,a,s') \left[r(s,a) + \gamma V_i(s') \right]$$

Let's think about this equation first:

$$V_{i+1}^{\pi}(s) = \sum_{s'} T(s,a,s') \left[r(s,a) + \gamma V_i^{\pi}(s') \right]$$

Model-free estimate of the value function

$$\mathbb{E}(a) = \sum_a P(a)a \;\Rightarrow\; \mathbb{E}(a) \approx \sum_{i=1}^{n} a_i$$

$$V_{i+1}^{\pi}(s) = \sum_{s'} T(s,a,s')\left[r(s,a) + \gamma V_i^{\pi}(s')\right]$$

Expectation

Thing being estimated

Model-free estimate of the value function

$$\mathbb{E}(a) = \sum_a P(a)a \Rightarrow \mathbb{E}(a) \approx \sum_{i=1}^{n} a_i$$

$$V_{i+1}^{\pi}(s) = \sum_{s'} T(s,a,s') \left[r(s,a) + \gamma V_i^{\pi}(s') \right]$$

Expectation

Thing being estimated

$$V_{i+1}^{\pi}(s) \approx \frac{1}{n} \sum_{i=1}^{n} r(s,a) + \gamma V_i^{\pi}(s')$$

Sample-based estimate

Model-free estimate of the value function

$$V^{\pi}_{i+1}(s) \approx \frac{1}{n}\sum_{i=1}^{n} r(s,a) + \gamma V^{\pi}_{i}(s')$$

How would we use this equation?

– get a bunch of samples of (s, a, s', r)

– for each sample, calculate $r + \gamma V^{\pi}_{i}(s')$

– average the results...

Weighted moving average

Suppose we have a random variable *X* and we want to estimate the mean from samples $x_1,\ldots,x_k$

After k samples $$\hat{x}_k = \frac{1}{k}\sum_{i=1}^{k} x_i$$

Can show that $$\hat{x}_k = \hat{x}_{k-1} + \frac{1}{k}(x_k - \hat{x}_{k-1})$$

Can be written $$\hat{x}_k = \hat{x}_{k-1} + \alpha(k)(x_k - \hat{x}_{k-1})$$

Learning rate $\alpha(k)$ can be functions other than 1, loose k conditions on learning rate to ensure convergence to mean

If learning rate is constant, weight of older samples decay exponentially at the rate $(1 - \alpha)$

Forgets about the past (distant past values were wrong anyway)

Update rule $$\hat{x} \neg \ \hat{x} + \alpha(x - \hat{x})$$

Weighted moving average

Suppose we have a random variable *X* and we want to estimate the mean from samples $x_1,\ldots,x_k$

After k samples $$\hat{x}_k = \frac{1}{k}\sum_{i=1}^{k} x_i$$

Can show that $$\hat{x}_k = \hat{x}_{k-1} + \frac{1}{k}(x_k - \hat{x}_{k-1})$$

Can be written $$\hat{x}_k = \hat{x}_{k-1} + \alpha(k)(x_k - \hat{x}_{k-1})$$

After several samples

$$V_{i+1}^{\pi}(s) \approx \frac{1}{n}\sum_{i=1}^{n} r(s,a) + \gamma V_i^{\pi}(s')$$

$$\approx V_i^{\pi}(s) + \alpha\left[r(s,a) + \gamma V_i^{\pi}(s') - V_i^{\pi}(s)\right]$$

or just drop the subscripts...

$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha\left[r + \gamma V^{\pi}(s') - V^{\pi}(s)\right]$$

Weighted moving average

Suppose we have a random variable *X* and we want to estimate the mean from samples $x_1, \ldots, x_k$

After k samples $$\hat{x}_k = \frac{1}{k}\sum_{i=1}^{k} x_i$$

Can show that $$\hat{x}_k = \hat{x}_{k-1} + \frac{1}{k}(x_k - \hat{x}_{k-1})$$

<u>This is called TD Value learning</u>

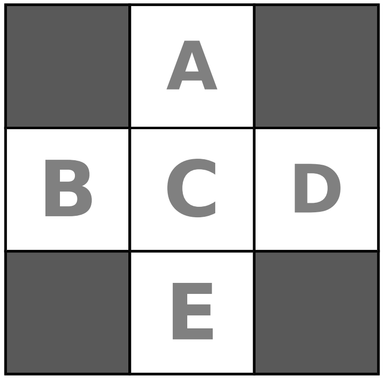
– thing inside the square brackets is called the “TD error”

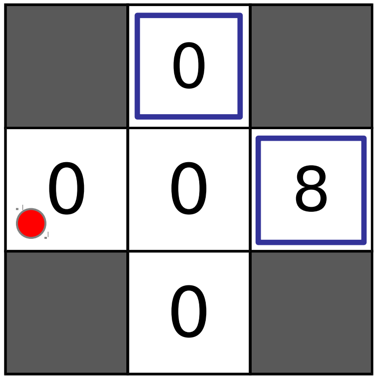
$$\approx V_i(s) + \alpha\,[r(s,a) + \gamma \ldots (s)]$$

or just drop the subscripts...

$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha\left[r + \gamma V^{\pi}(s') - V^{\pi}(s)\right]$$

TD Value Learning: example

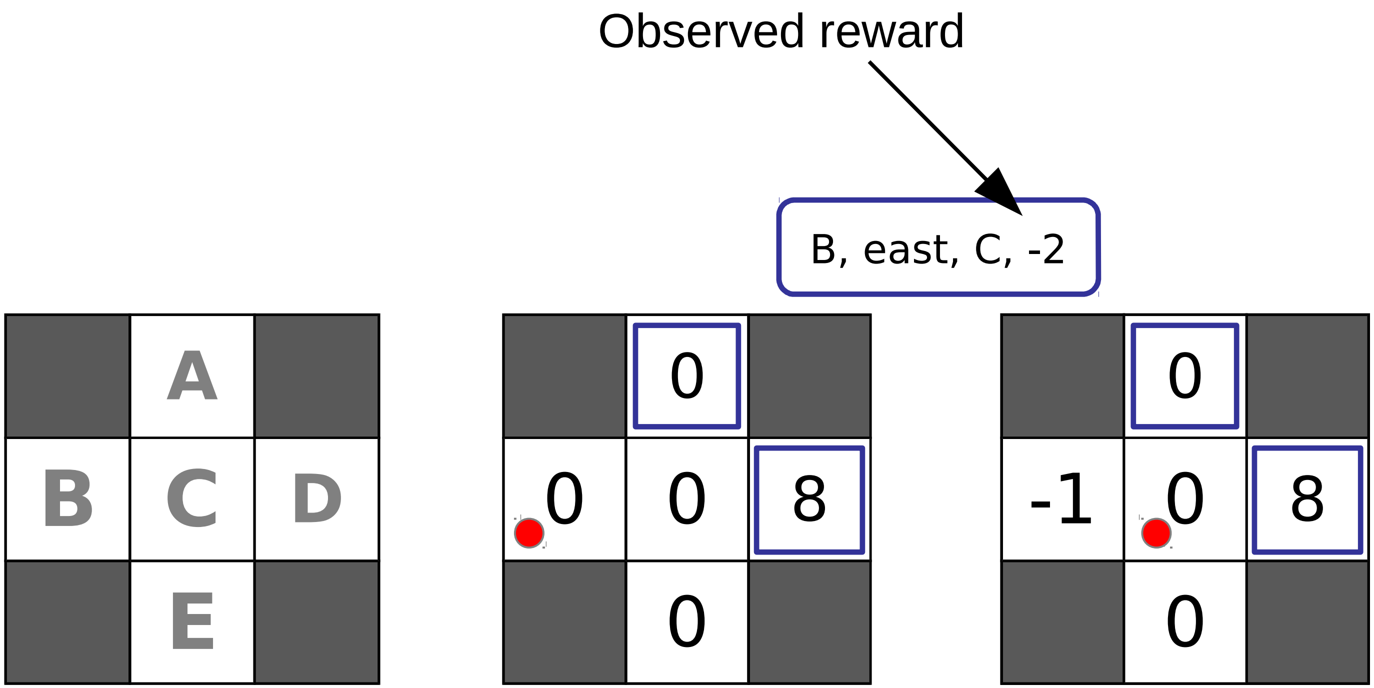




$\gamma = 1, \alpha = 0.5$

$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha \left[r + \gamma V^{\pi}(s') - V^{\pi}(s) \right]$$

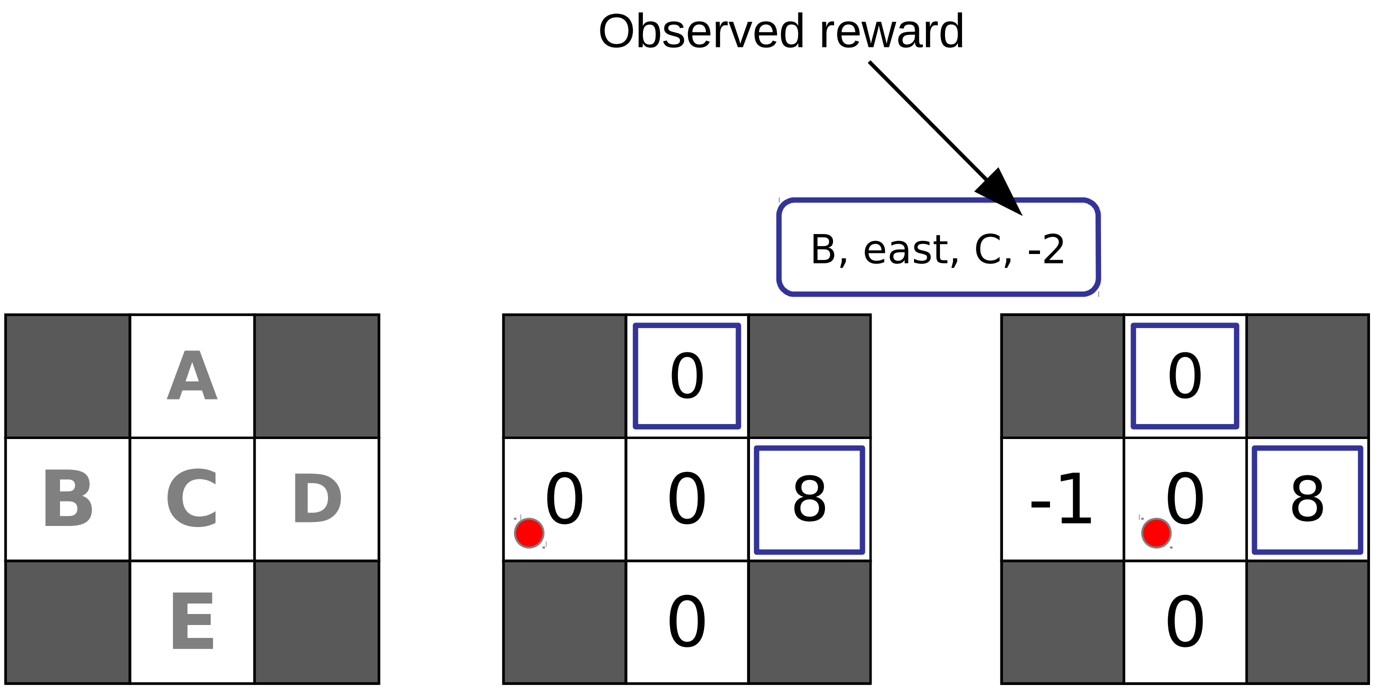
TD Value Learning: example



$$\gamma = 1, \alpha = 0.5$$

$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha \left[r + \gamma V^{\pi}(s') - V^{\pi}(s) \right]$$

TD Value Learning: example

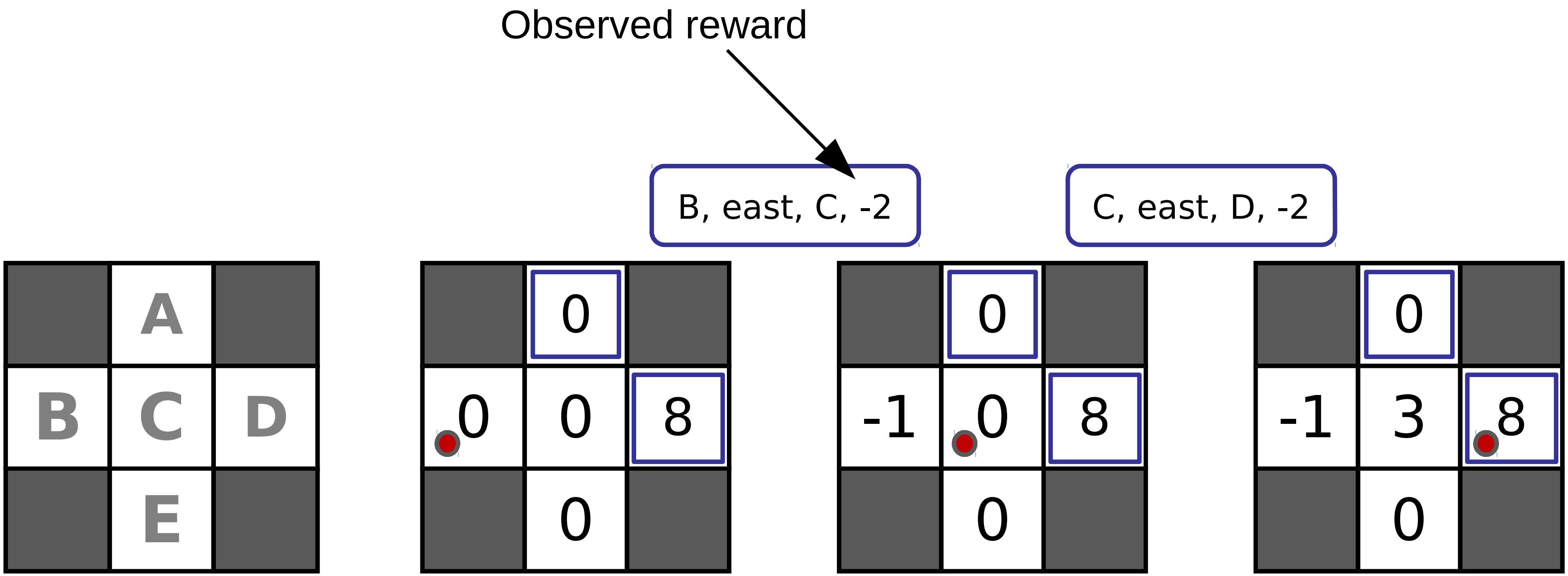


$$\gamma = 1, \alpha = 0.5$$

$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha \left[r + \gamma V^{\pi}(s') - V^{\pi}(s) \right]$$

$$V^{\pi}(s) \leftarrow 0 + 0.5 \left[-2 + 0 - 0 \right]$$

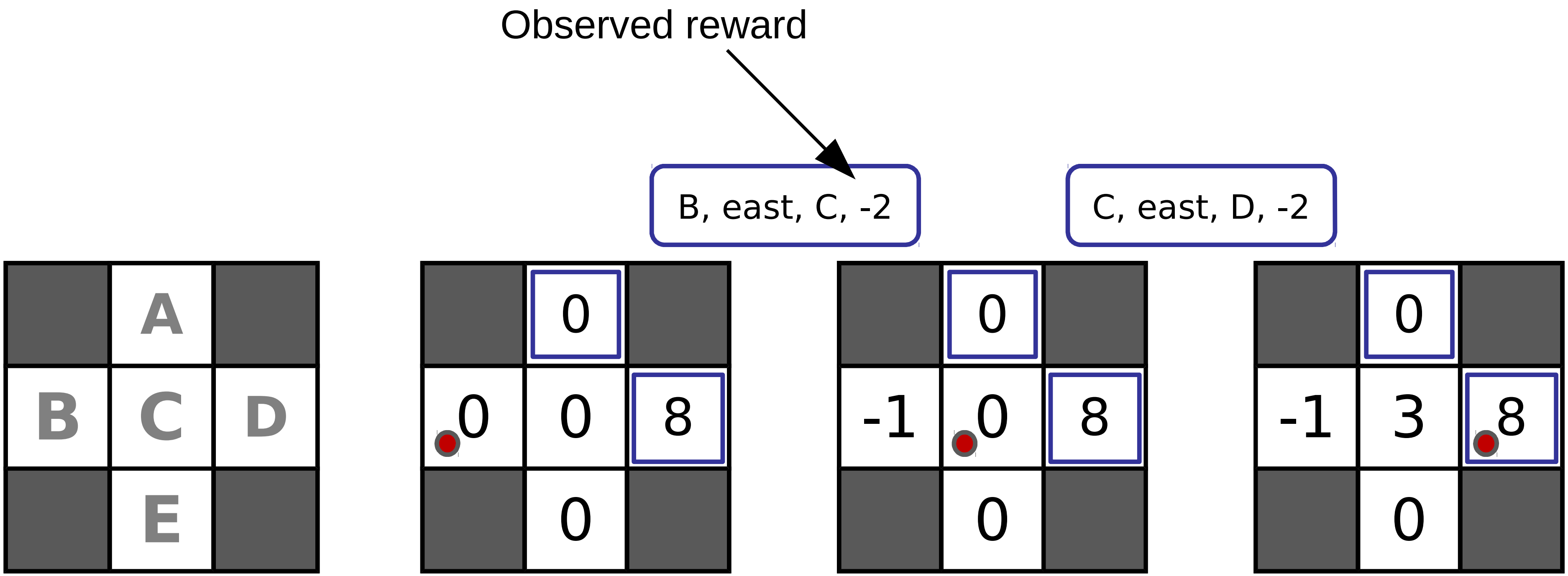
TD Value Learning: example



$$\gamma = 1, \alpha = 0.5$$

$$V^\pi(s) \leftarrow V^\pi(s) + \alpha \left[r + \gamma V^\pi(s') - V^\pi(s) \right]$$

TD Value Learning: example



$\gamma = 1, \alpha = 0.5$

$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha \left[r + \gamma V^{\pi}(s') - V^{\pi}(s) \right]$$

$$V^{\pi}(s) \leftarrow 0 + 0.5 \left[-2 + 8 - 0 \right]$$

What's the problem w/ TD Value Learning?

What's the problem w/ TD Value Learning?

Can't turn the estimated value function into a policy!

This is how we did it when we were using value iteration:

$$\pi^*(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

Why can't we do this now?

What's the problem w/ TD Value Learning?

Can't turn the estimated value function into a policy!

This is how we did it when we were using value iteration:

$$\pi^*(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

Why can't we do this now?

<u>Solution: Use TD value learning to estimate Q*, not V*</u>

How do we estimate Q?

$V(s)$ ← Value of being in state s and acting optimally

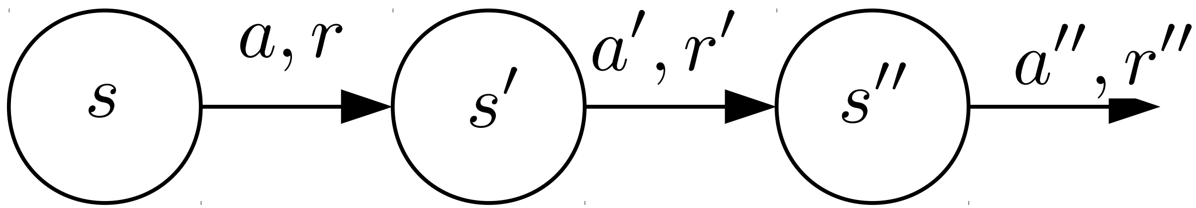
$Q(s,a)$ ← Value of taken action a from state s and then acting optimally

$$Q_{i+1}(s,a) = \sum_{s'} T(s,a,s') \left[r(s,a) + \gamma V_i(s') \right]$$

$$= \sum_{s'} T(s,a,s') \left[r(s,a) + \gamma \max_{a'} Q_i(s',a') \right]$$

Use this equation inside of the value iteration loop we studied last lecture...

Model-free reinforcement learning



Life consists of a sequence of tuples like this: (s,a,s',r')

Use these updates to get an estimate of Q(s,a)

How?

Model-free reinforcement learning

Here's how we estimated V:

$$V^\pi(s) \leftarrow V^\pi(s) + \alpha \left[r + \gamma V^\pi(s') - V^\pi(s)\right]$$

So do the same thing for Q:

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

Model-free reinforcement learning

Here's how we estimated V:

$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha \left[r + \gamma V^{\pi}(s') - V^{\pi}(s) \right]$$

So do the same thing for Q:

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

This is called Q-Learning
Most famous type of RL

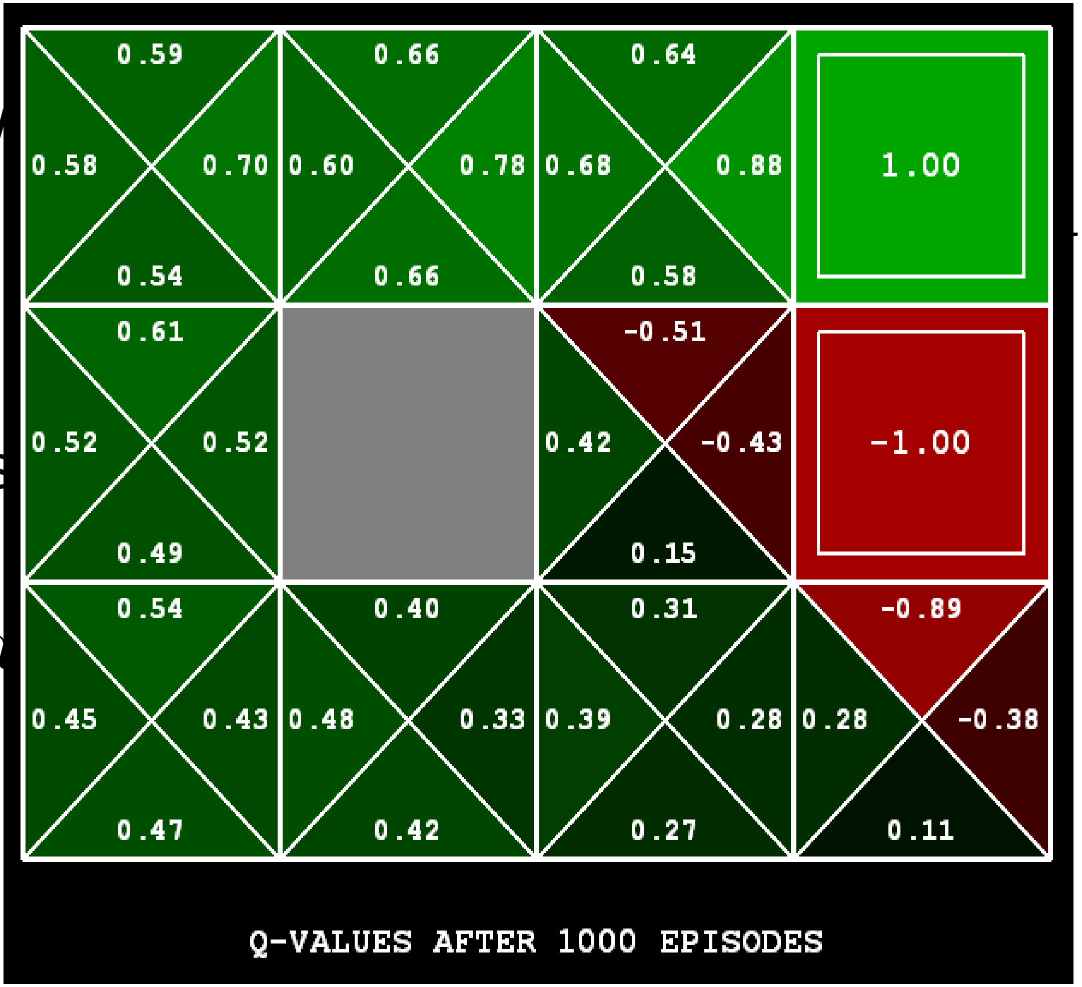
Model-free reinforcement learning

Here's how

$V^{\pi}(s$ … $(s)]$

So do the s

$Q(s, a$ … $a') - Q(s,a)\Big]$



Q-values learned using Q-Learning

Q-Learning

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ϵ-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
    until S is terminal

Q-Learning: properties

Q-learning converges to optimal Q-values if:

1. it explores every s, a, s' transition sufficiently often

2. the learning rate approaches zero (eventually)

Key insight: Q-value estimates converge even if experience is obtained using a suboptimal policy.

This is called off-policy learning

SARSA

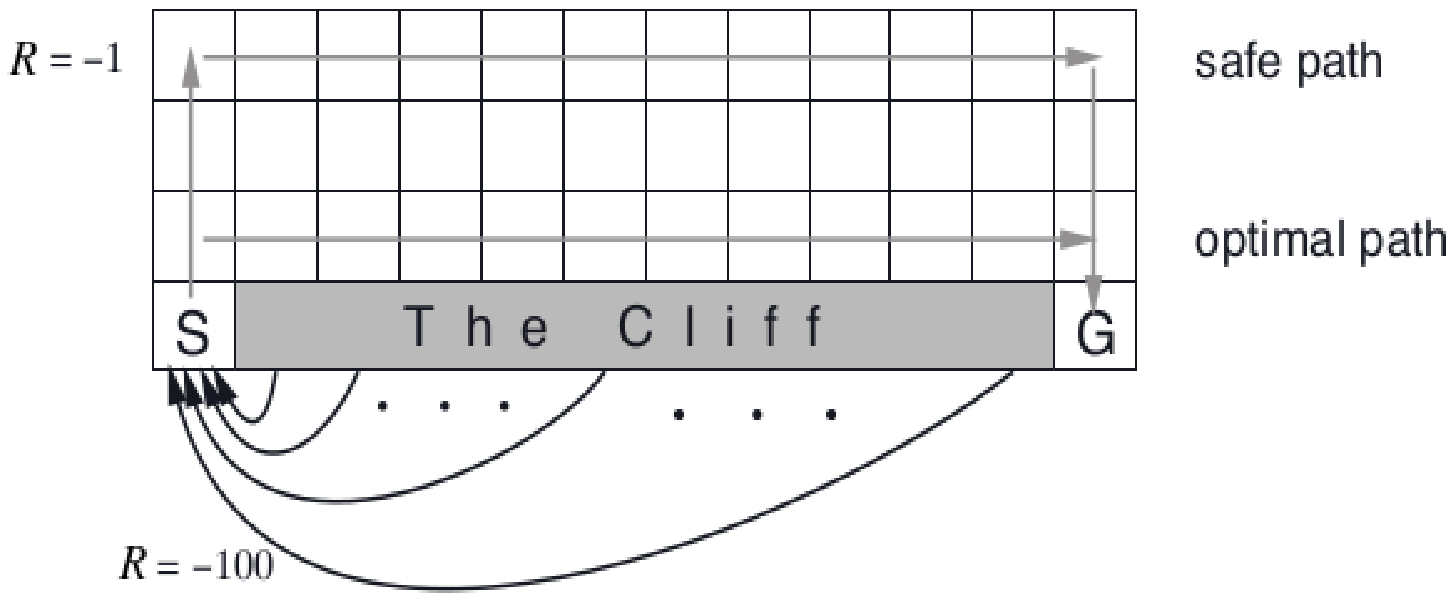
Q-learning

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until S is terminal

SARSA

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

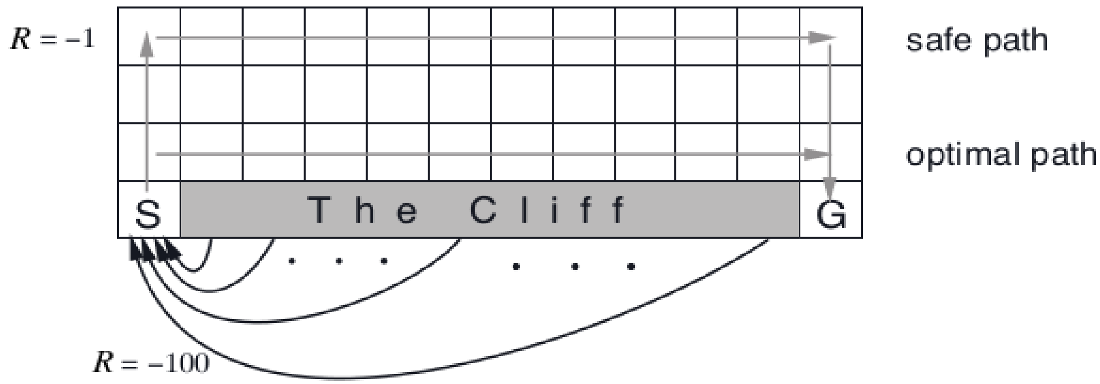
Q-learning vs SARSA

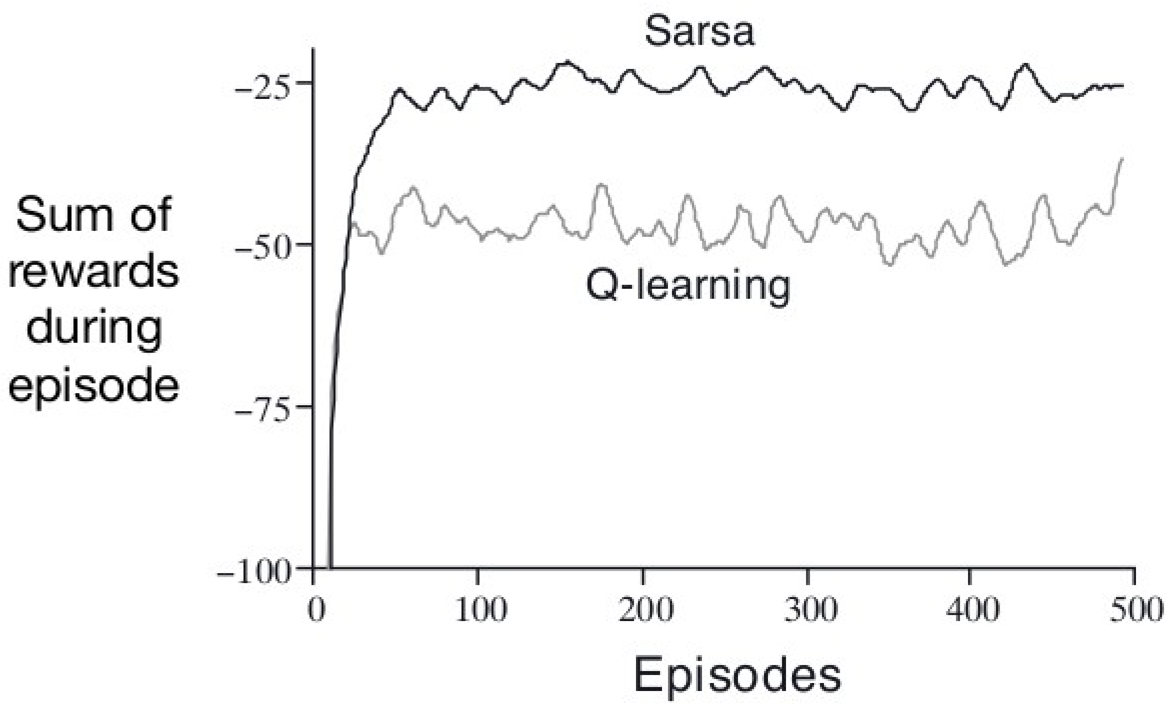


Which path does SARSA learn?

Which one does q-learning learn?

Q-learning vs SARSA





Exploration vs exploitation

Think about how we choose actions:

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until S is terminal

$$a = \arg\max_a Q(s,a)$$

<u>But: if we only take “greedy” actions, then how do we explore?</u>
– if we don't explore new states, then how do we learn anything new?

Exploration vs exploitation

Think about how we choose actions:

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take
    $Q($
    $S \leftarrow$
  until S is

Taking only greedy actions makes it more likely that you get stuck in local minimia in the policy space

$$\leftarrow \arg\max_a Q(s,a)$$

But: if we only take "greedy" actions, then how do we explore?
– if we don't explore new states, then how do we learn anything new?

Exploration vs exploitation

Choose a random action e% of the time.
OW, take the greedy action

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q($ ... $tate, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ϵ-greedy)
        Take action A, observe R, S'
        $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
        $S \leftarrow S'$
    until S is terminal

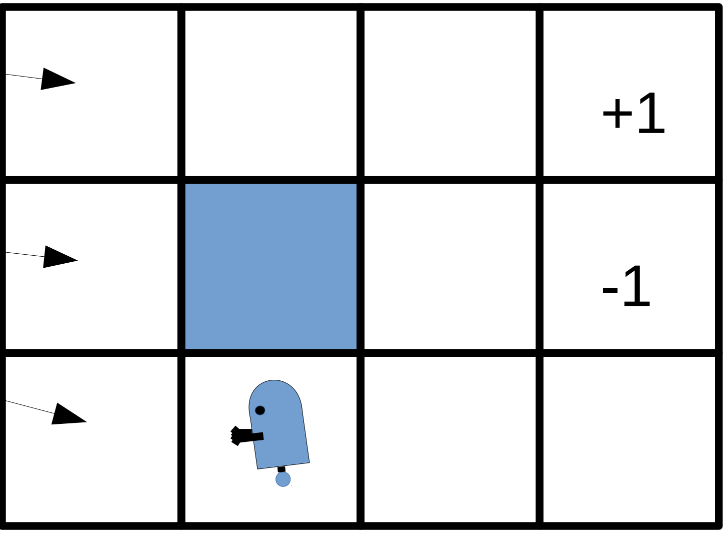
$$a = \arg\max_a Q(s,a)$$

<u>But: if we only take “greedy” actions, then how do we explore?</u>
– if we don't explore new states, then how do we learn anything new?

Function approximation

So far, the policy is distinct for each state

– knowing something about this state tells us nothing about what to do in other states.

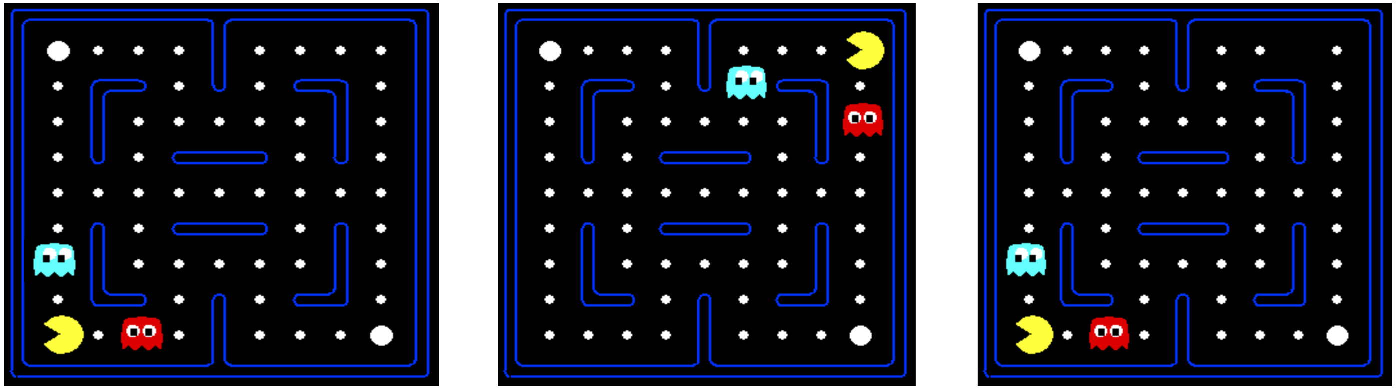


Function approximation

So far, the policy is distinct for each state

– knowing something about this state tells us nothing about what to do in other states.

But, what if you have a large state space?

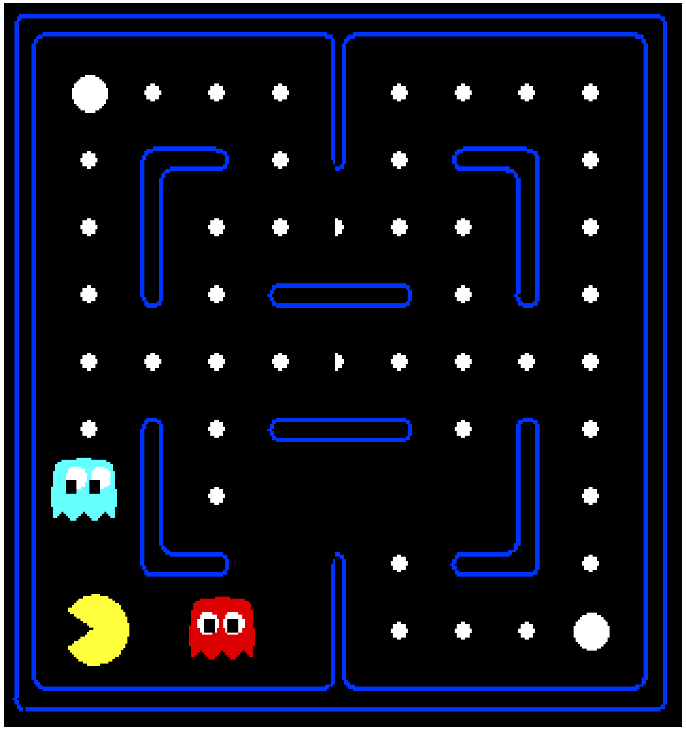


How should these states generalize?

Feature-based representations

Solution: describe a state using a vector of features (properties)

- Features are functions from states to real numbers (often 0/1) that capture important properties of the state
- Example features:
  - Distance to closest ghost
  - Distance to closest dot
  - Number of ghosts
  - 1 / (dist to dot)^2
  - Is Pacman in a tunnel? (0/1)
  - …… etc.
- Can also describe a q-state (s, a) with features (e.g. action moves closer to food)



Linear value functions

Using a feature representation, we can write a q function (or value function) for any state using a few weights:

$$V(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

Advantage: our experience is summed up in a few powerful numbers

Disadvantage: states may share features but actually be very different in value!

Approximate Q-learning

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

Q-learning with linear Q-functions:

$$\text{transition } = (s, a, r, s')$$

$$\text{difference} = \left[r + \gamma \max_{a'} Q(s', a')\right] - Q(s,a)$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha \,[\text{difference}]$$ Exact Q's

$$w_i \leftarrow w_i + \alpha \,[\text{difference}]\, f_i(s,a)$$ Approximate Q's

Intuitive interpretation:

Adjust weights of active features

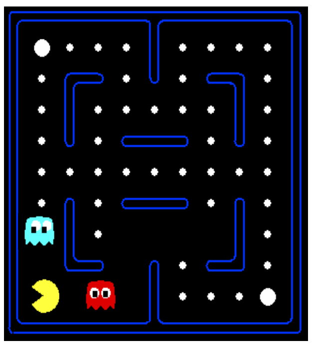
E.g., if something unexpectedly bad happens, blame the features that were on: disprefer all states with that state's features

Formal justification: online least squares

Example: Q-Pacman

$$Q(s,a) = 4.0 f_{DOT}(s,a) - 1.0 f_{GST}(s,a)$$

s



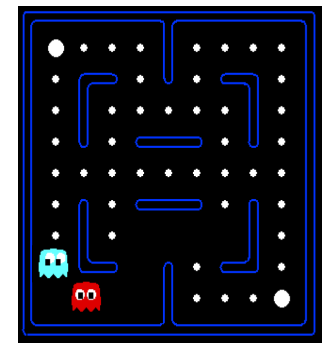
$f_{DOT}(s, \text{NORTH}) = 0.5$

$f_{GST}(s, \text{NORTH}) = 1.0$

$a = \text{NORTH}$

$r = -500$

s'



$Q(s, \text{NORTH}) = +1$

$Q(s', \cdot) = 0$

$$r + \gamma \max_{a'} Q(s', a') = -500 + 0$$

difference $= -501$

$$w_{DOT} \leftarrow 4.0 + \alpha\,[-501]\,0.5$$

$$w_{GST} \leftarrow -1.0 + \alpha\,[-501]\,1.0$$

$$Q(s,a) = 3.0 f_{DOT}(s,a) - 3.0 f_{GST}(s,a)$$