Reinforcement Learning

George Konidaris
gdk@cs.brown.edu

Fall 2021

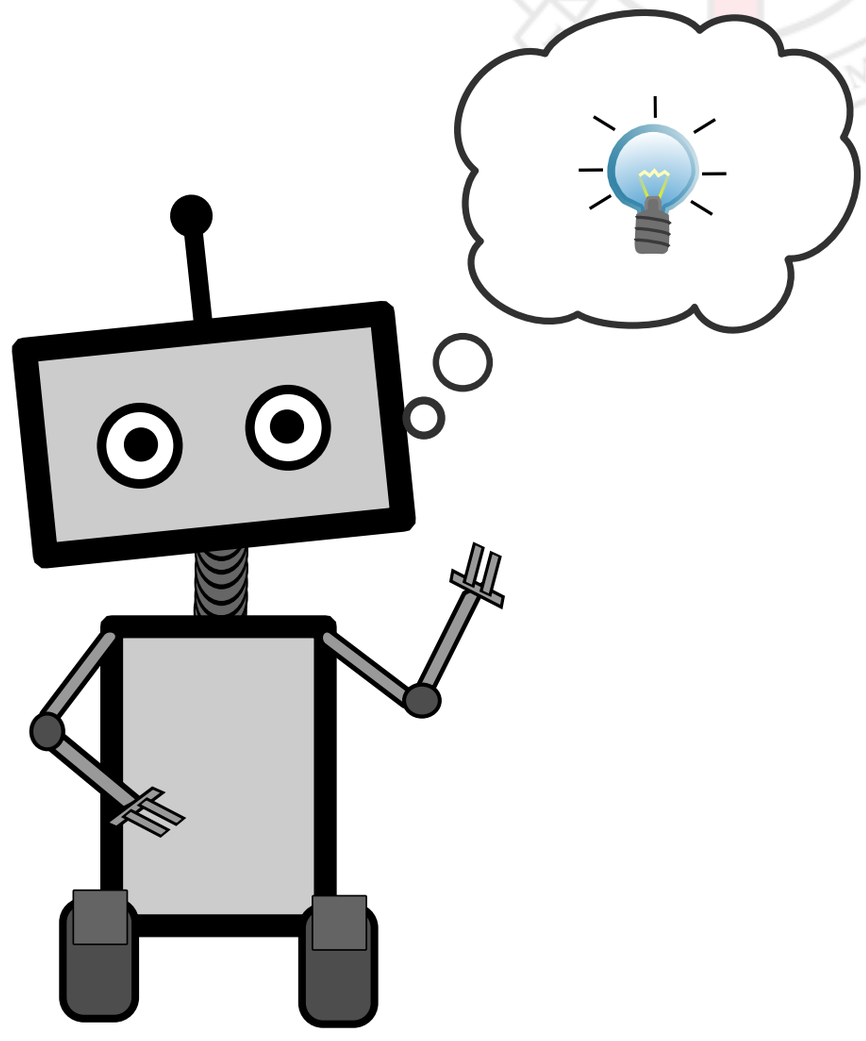
Machine Learning

Subfield of AI concerned with *learning from data*.

Broadly, using:

- ***Experience***
- To Improve ***Performance***
- On Some ***Task***

(Tom Mitchell, 1997)



vs . . .

ML

vs

Statistics

vs

Data Mining

Why?

Developing effective learning methods has proved difficult.
Why bother?

Autonomous discovery

- We don't know something, want to find out.

Hard to program

- Easier to specify task, collect data.

Adaptive behavior

- Our agents should adapt to new data, unforeseen circumstances.

Types of Machine Learning

Depends on *feedback available*:

Labeled data:

- Supervised learning

No feedback, just data:

- Unsupervised learning.

Sequential data, weak labels:

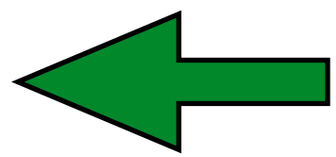
- Reinforcement learning

Supervised Learning

Input:

$X = \{x_1, \ldots, x_n\}$ inputs

$Y = \{y_1, \ldots, y_n\}$ labels



training data

Learn to *predict new labels*.

Given x: y?

Unsupervised Learning

Input:

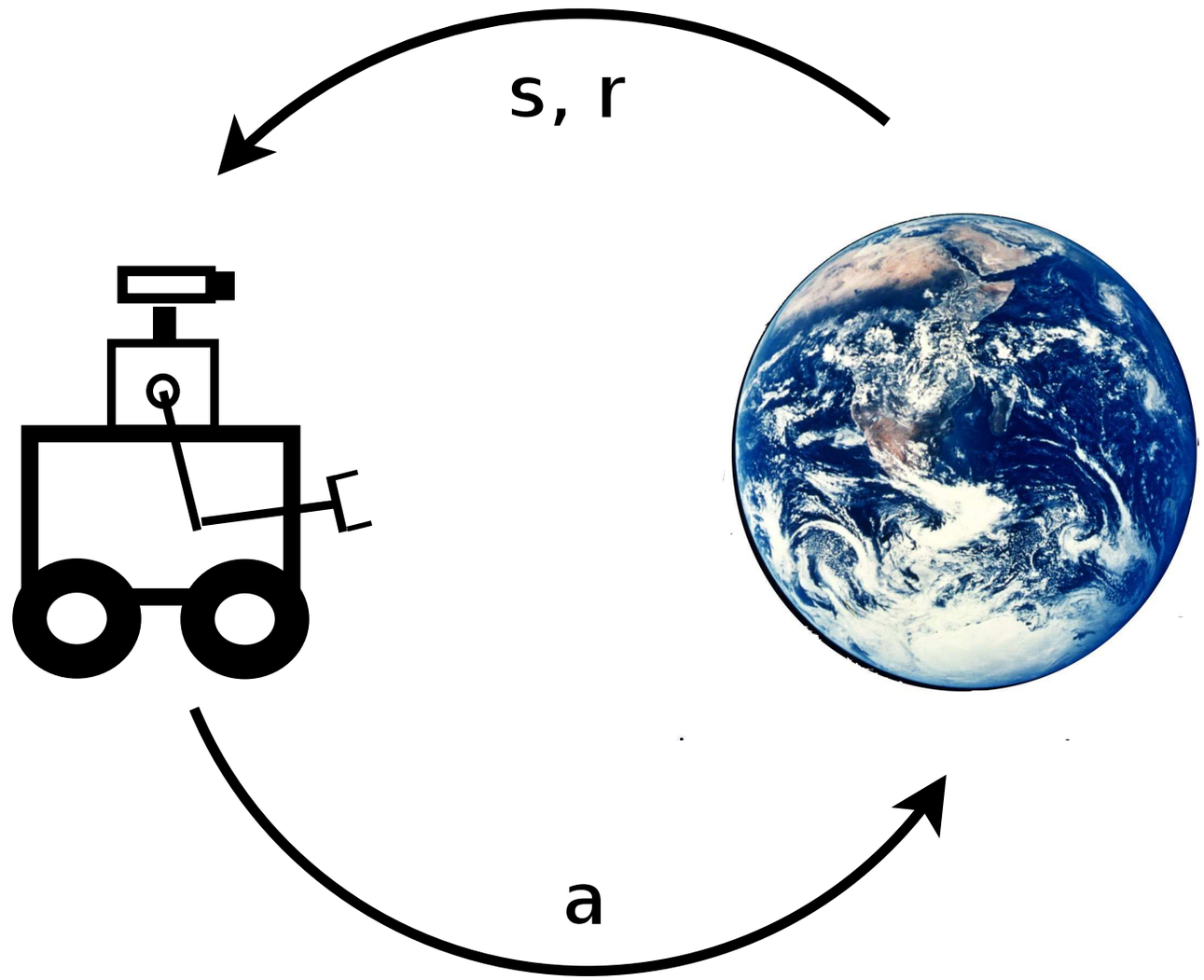
$X = \{x_1, \ldots, x_n\}$ inputs

Try to understand the *structure of the data.*

E.g., how many types of cars?
How can they vary?

Reinforcement Learning

Learning counterpart of planning.



$$\pi : S \rightarrow A$$

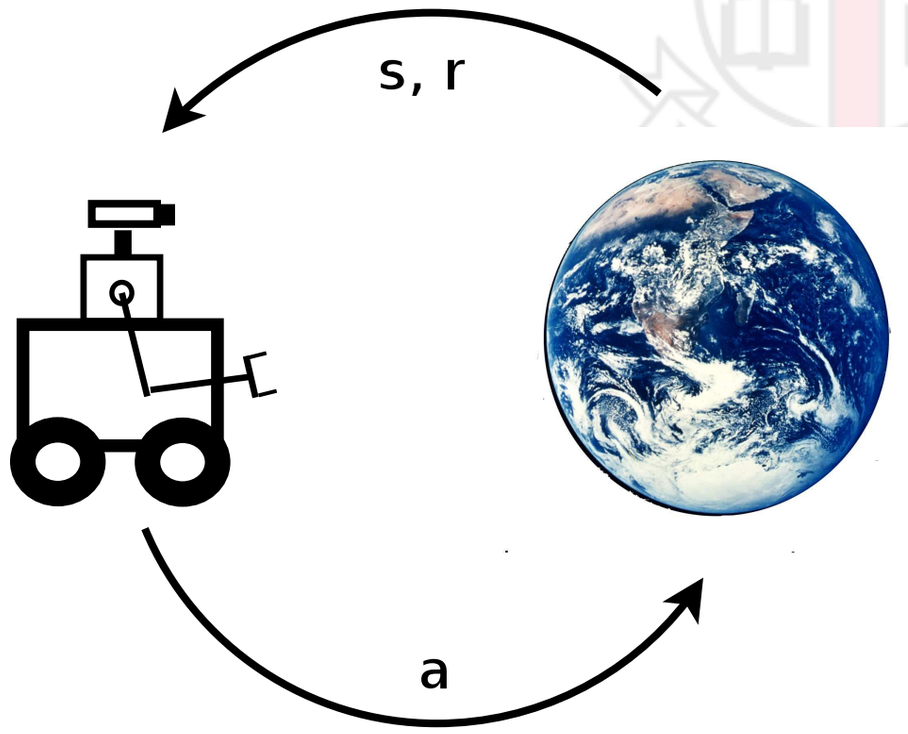
$$\max_{\pi} R = \sum_{t=0}^{\infty} \gamma^t r_t$$

MDPs

Agent interacts with an environment

At each time t:

- Receives sensor signal s_t
- Executes action a_t
- *Transition*:
  - new sensor signal s_{t+1}
  - reward r_t



Goal: find policy π that maximizes expected return (sum of discounted future rewards):

$$\max_{\pi} \mathbb{E}\left[R = \sum_{t=0}^{\infty} \gamma^t r_t\right]$$

Markov Decision Processes

S : set of *states*
A : set of actions
γ : discount factor

$$< S, A, \gamma, R, T >$$

R : reward function

$R(s, a, s')$ is the reward received taking action a from state s and transitioning to state s'.

T : transition function

$T(s'|s, a)$ is the probability of transitioning to state s' after taking action a in state s.

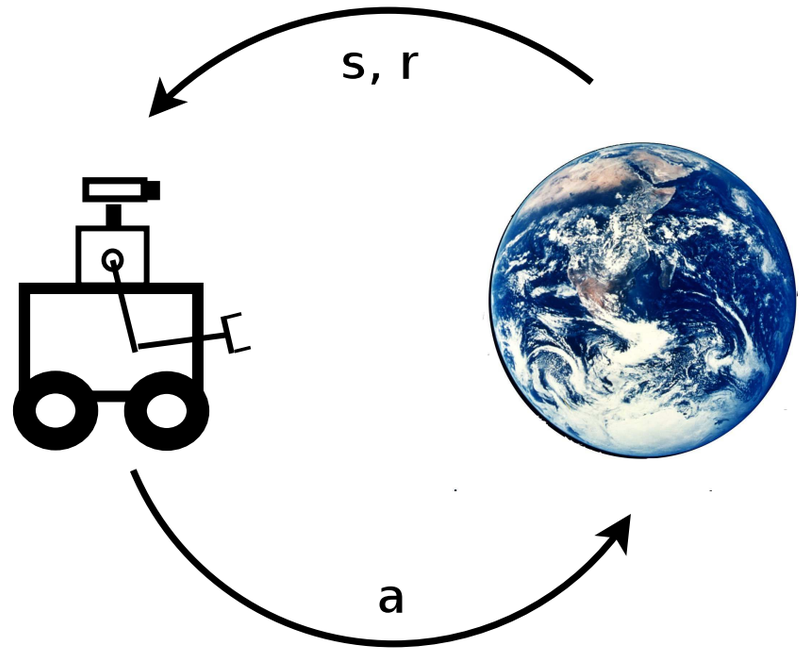
RL vs Planning

In planning:

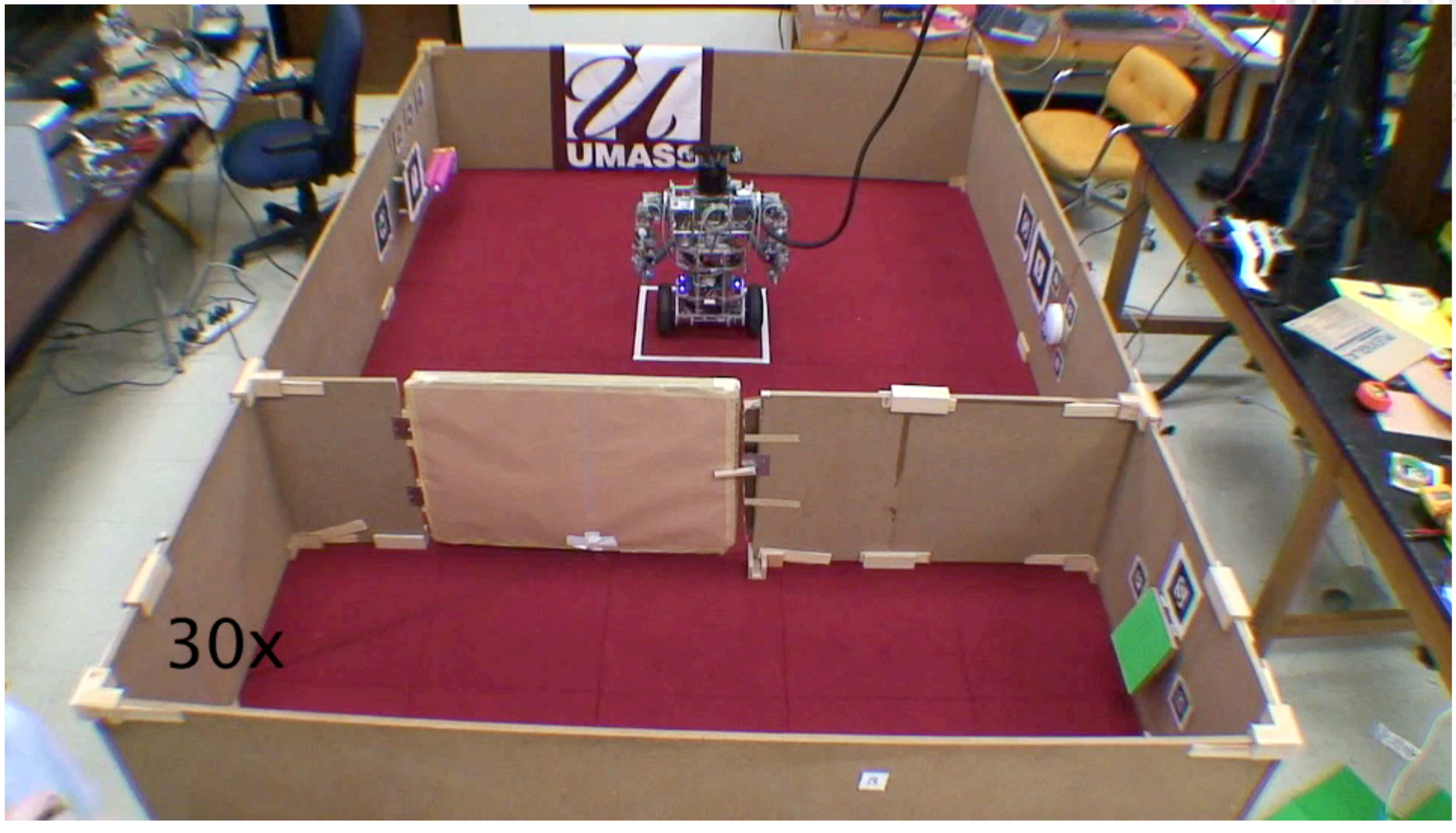
- Transition function (*T*) known.
- Reward function (*R*) known.
- Computation "offline".

In reinforcement learning:

- One or both of T, R unknown.
- *Action in the world* only source of data.
- Transitions are *executed* not *simulated*.

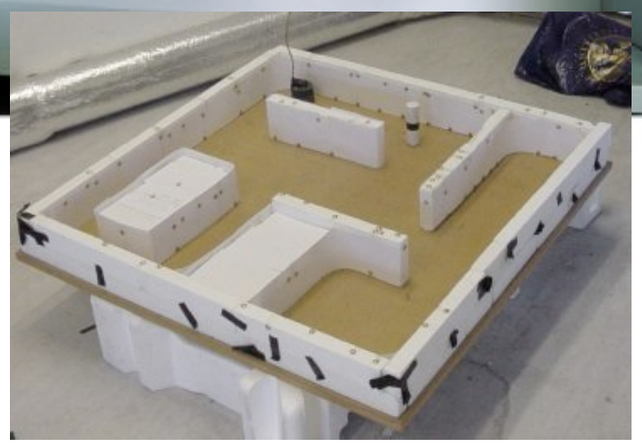


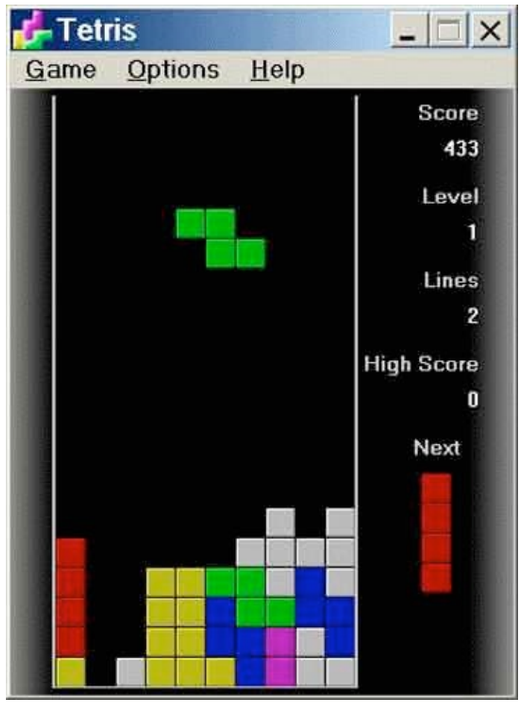
Reinforcement Learning



RL

This formulation is general enough to encompass a wide variety of learned control problems.





MDPs

As before, our target is a **policy**:

$$\pi : S \rightarrow A$$

A policy maps *states* to *actions*.

The **optimal policy** maximizes:

$$\max_{\pi} \forall s, \mathbb{E}\left[R(s) = \sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s\right]$$

This means that we wish to find a policy that maximizes the **return** from **every state.**

Planning via Policy Iteration

In planning, we used policy iteration to find an optimal policy.

1. Start with a policy π
2. Estimate V^π
3. Improve π
   a. $\pi(s) = \max_a \mathbb{E}\left[r + \gamma V^\pi(s')\right], \forall s$

Repeat

More precisely, we use a value function:

$$V^\pi(s) = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r_i\right]$$

can't do this anymore

… then we would update π by computing:

$$\pi(s) = \text{argmax}_a \sum_{s'} T(s,a,s')\left[r(s,a,s') + \gamma V[s']\right]$$

Value Functions

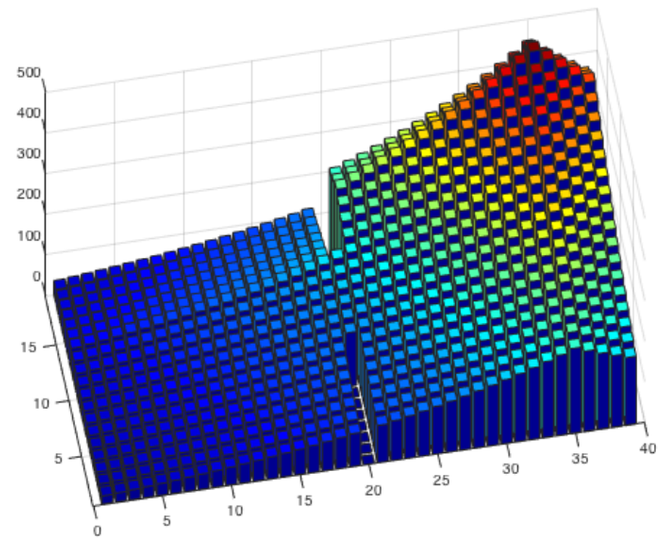
For learning, we use a *state-action value function* as follows:

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, a_0 = a\right]$$

This is the value of executing a in state s, *then following* π.

Note that $V^{\pi}(s) = Q^{\pi}(s, \pi(s))$.

|A| x



Policy Iteration

This leads to a general policy improvement framework:

1. Start with a policy π
2. Learn Q^π
3. Improve π
   a. $\pi(s) = \max_a Q(s, a), \forall s$

Repeat

Steps 2 and 3 can be interleaved as rapidly as you like. Usually, perform 3a *every time step*.

Value Function Learning

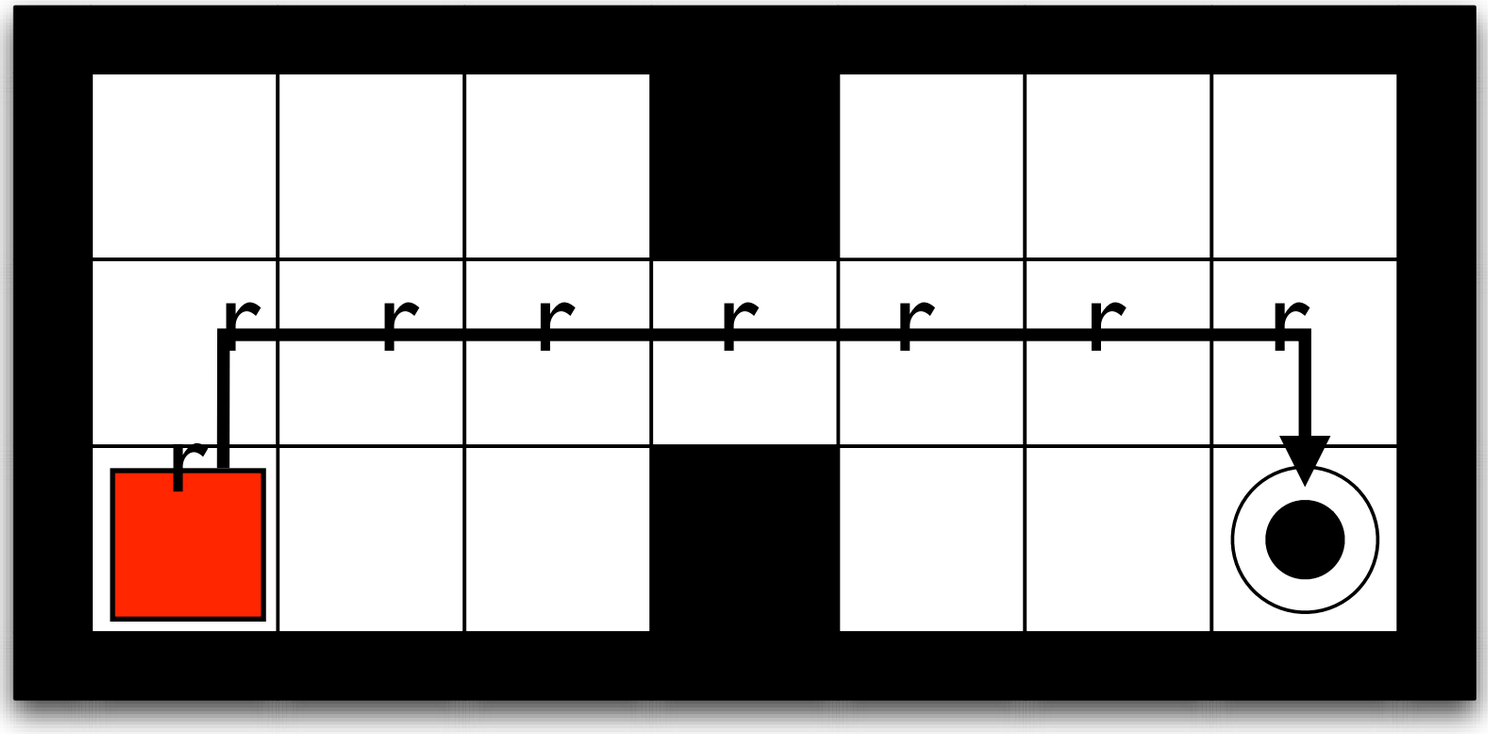
Learning proceeds by gathering samples of $Q(s, a)$.

Methods differ by:

- How you get the samples.
- How you use them to update Q.

Monte Carlo

Simplest thing you can do: sample $R(s)$.



Do this repeatedly, average values:

$$Q(s,a) = \frac{R_1(s) + R_2(s) + ... + R_n(s)}{n}$$

Temporal Difference Learning

Where can we get more (immediate) samples?

Idea: *use the Bellman equation.*

$$Q^{\pi}(s,a) = \mathbb{E}_{s'}\left[r(s,a,s') + \gamma Q^{\pi}(s', \pi(s'))\right]$$

value of this state

reward

value of next state

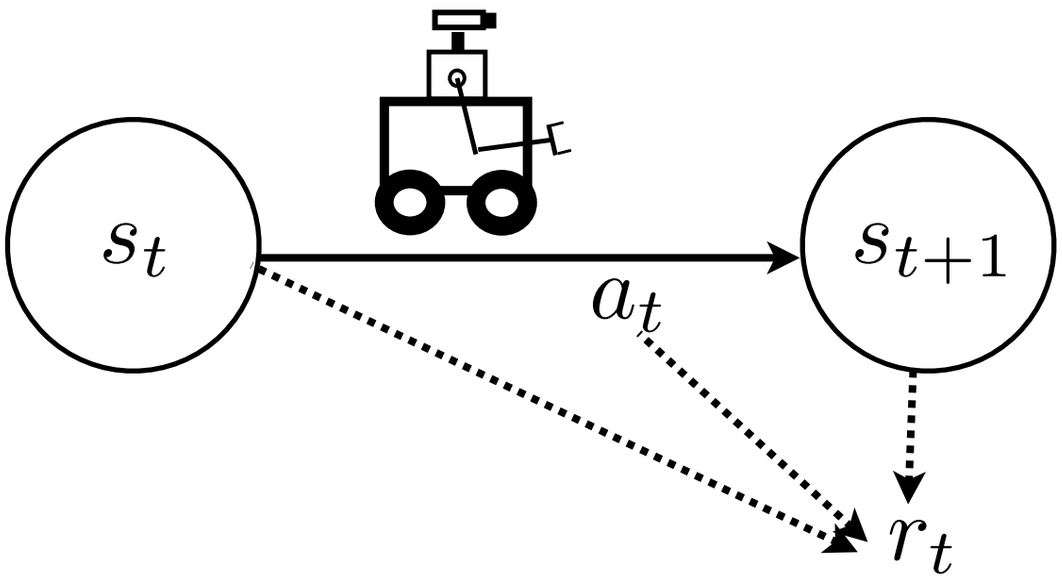
TD Learning

Ideally and in expectation:

$$r_i + \gamma Q(s_{i+1}, a_{i+1}) - Q(s_i, a_i) = 0$$

Q is correct if this holds in expectation for all states.

When it does not: *temporal difference error.*



$$Q(s_t, a_t) \leftarrow r_t + \gamma Q(s_{t+1}, a_{t+1})$$

Sarsa

Sarsa: very simple algorithm

1. Initialize Q[s][a] = 0
2. For *n* episodes
    - observe state *s*
    - select *a* = argmax$_a$ Q[s][a]
    - observe transition (s, a, r, s', a')
        - compute TD error $\delta = r + \gamma Q(s', a') - Q(s, a)$
        - update Q: $Q(s, a) = Q(s, a) + \alpha\delta$
    - if not end of episode, repeat

zero by def.
if *s* is absorbing

Exploration vs. Exploitation

Always $\max_a Q(s, a)$?

- Exploit current knowledge.

What if your current knowledge is wrong?
How are you going to find out?

- Explore to gain new knowledge.

Exploration is ***mandatory*** if you want to find the optimal solution, but every exploratory action may sacrifice reward.

Exploration vs. Exploitation - when to try new things? Consistent theme of RL.

Exploration vs. Exploitation

How to balance?

Simplest, most popular approach:

Instead of *always* being greedy:

- $\max_a Q(s, a)$

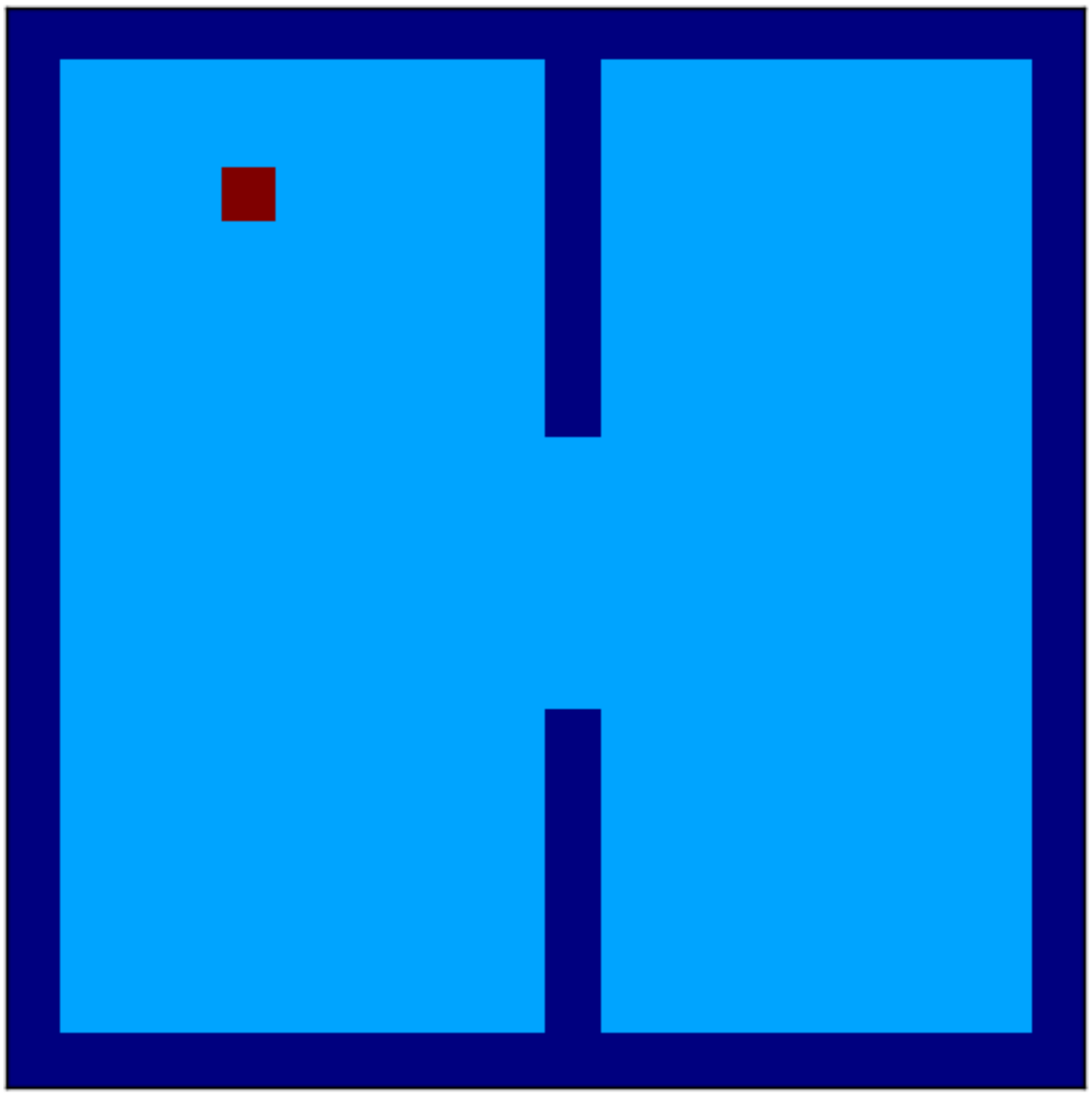
Explore with probability ϵ:

- $\max_a Q(s, a)$ with probability $(1 - \epsilon)$.
- random action with probability ϵ.

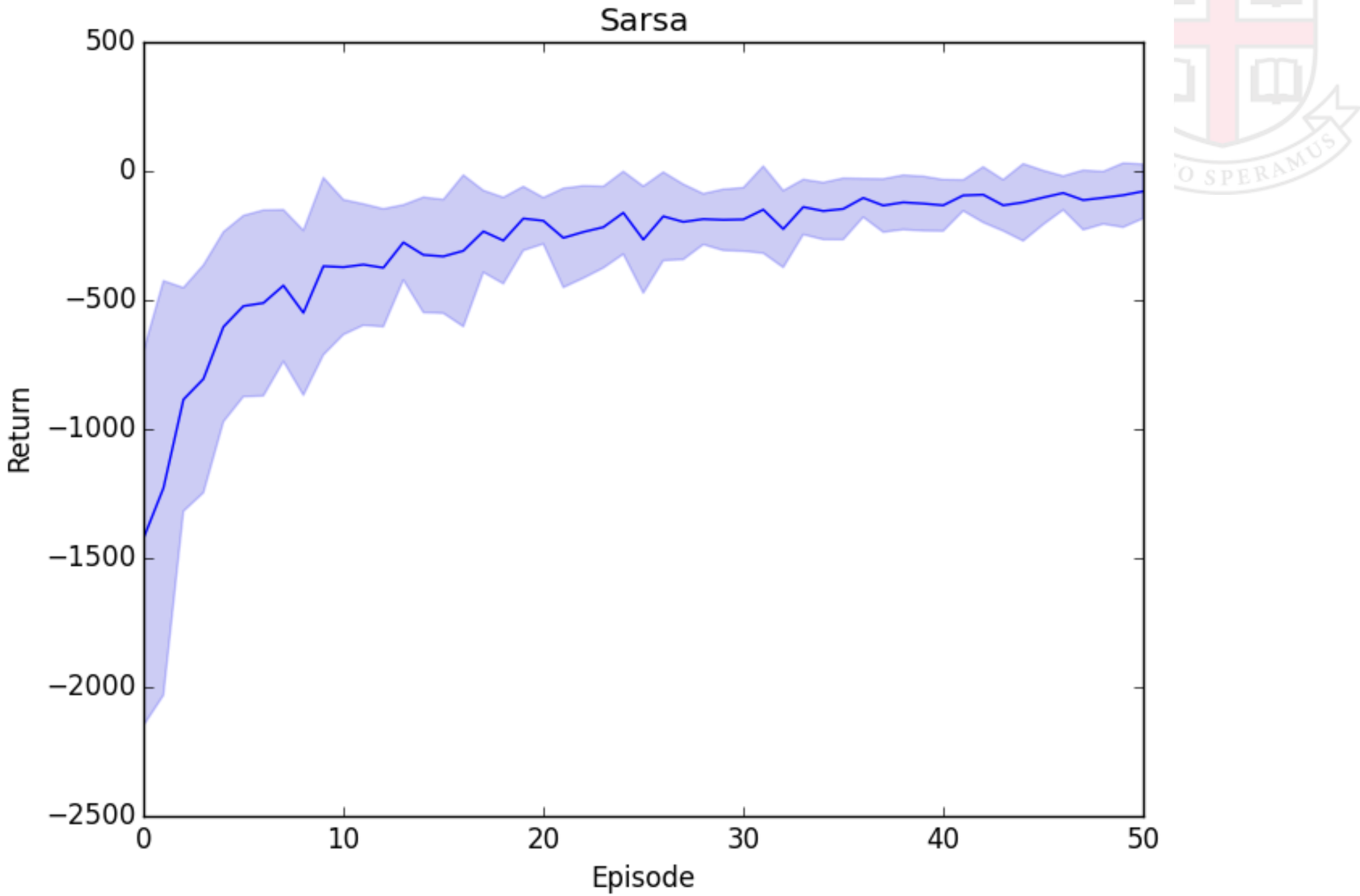
ϵ - greedy exploration $(\epsilon \approx 0.1)$

- Very simple
- Ensures asymptotic coverage of state space

Sarsa

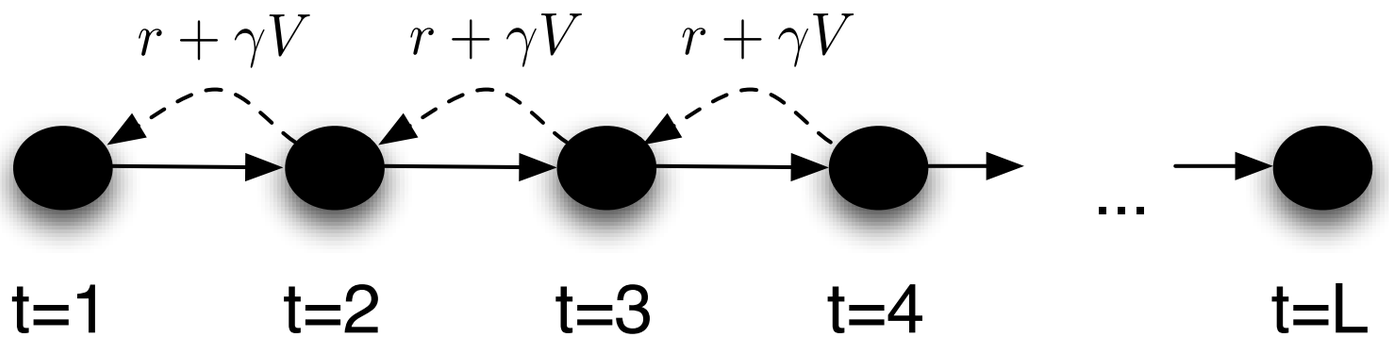


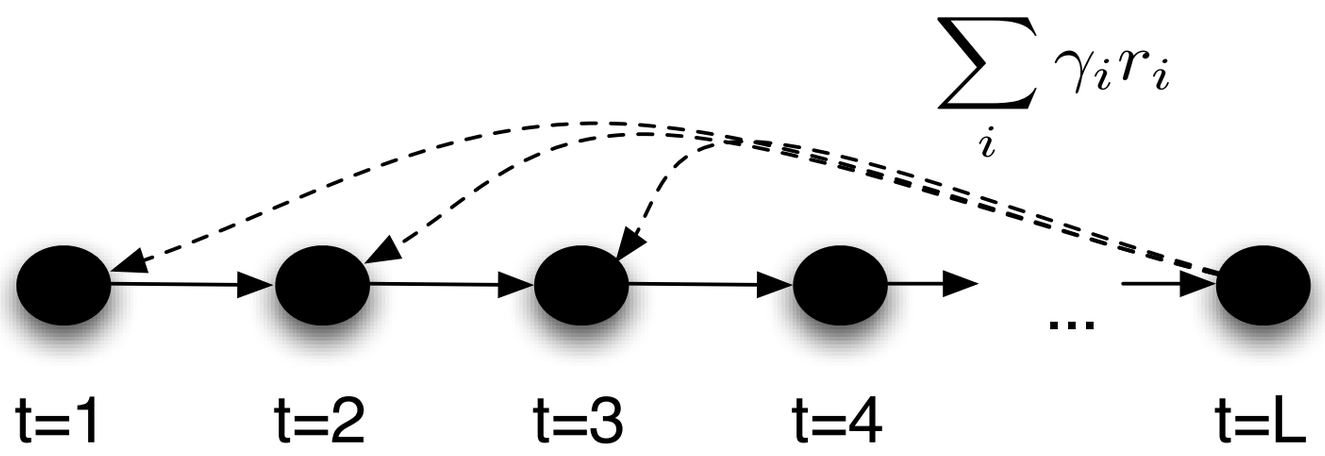
Sarsa


Sarsa
500
0
−500
−1000
−1500
−2000
−2500
Return
0
10
20
30
40
50
Episode

TD vs. MC

TD and MC two extremes of obtaining samples of Q:





Generalizing TD

We can generalize this to the idea of an n-step rollout:

$$R_{s_t}^{(1)} = r_t + \gamma Q(s_{t+1}, a_{t+1})$$

$$R_{s_t}^{(2)} = r_t + \gamma r_{t+1} + \gamma^2 Q(s_{t+2}, a_{t+2})$$

.

.

.

$$R_{s_t}^{(n)} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... + \gamma^{n-1} r_{t+n-1} + \gamma^n Q(s_{t+n}, a_{t+n})$$

Each tells us something about the value function.

- We can combine *all* n-step rollouts.
- This is known as a *complex backup*.

TD(λ)

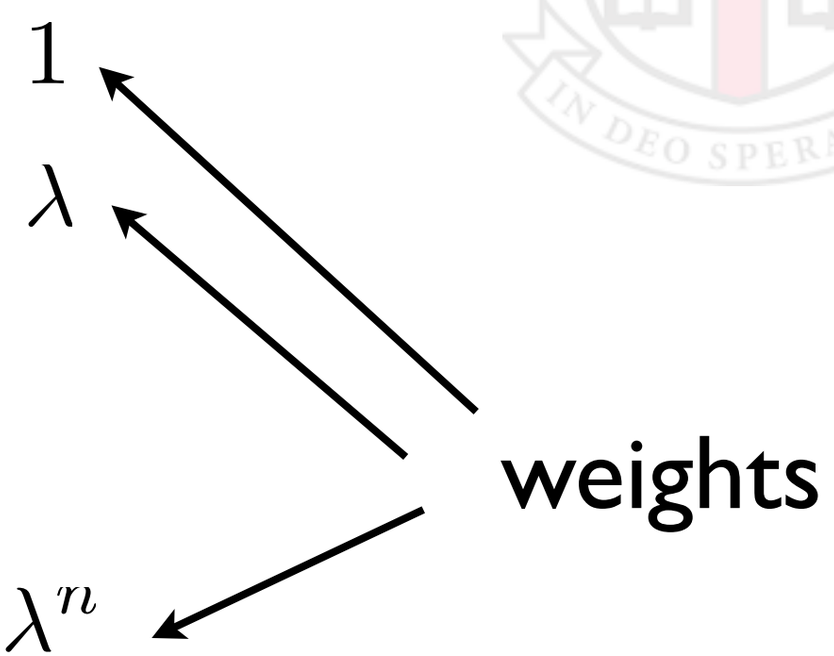
Weighted sum:

$$R^{(1)} = r_0 + \gamma Q(s_1, a_1)$$

$$R^{(2)} = r_0 + \gamma r_1 + \gamma^2 Q(s_2, a_2)$$

$$\vdots$$

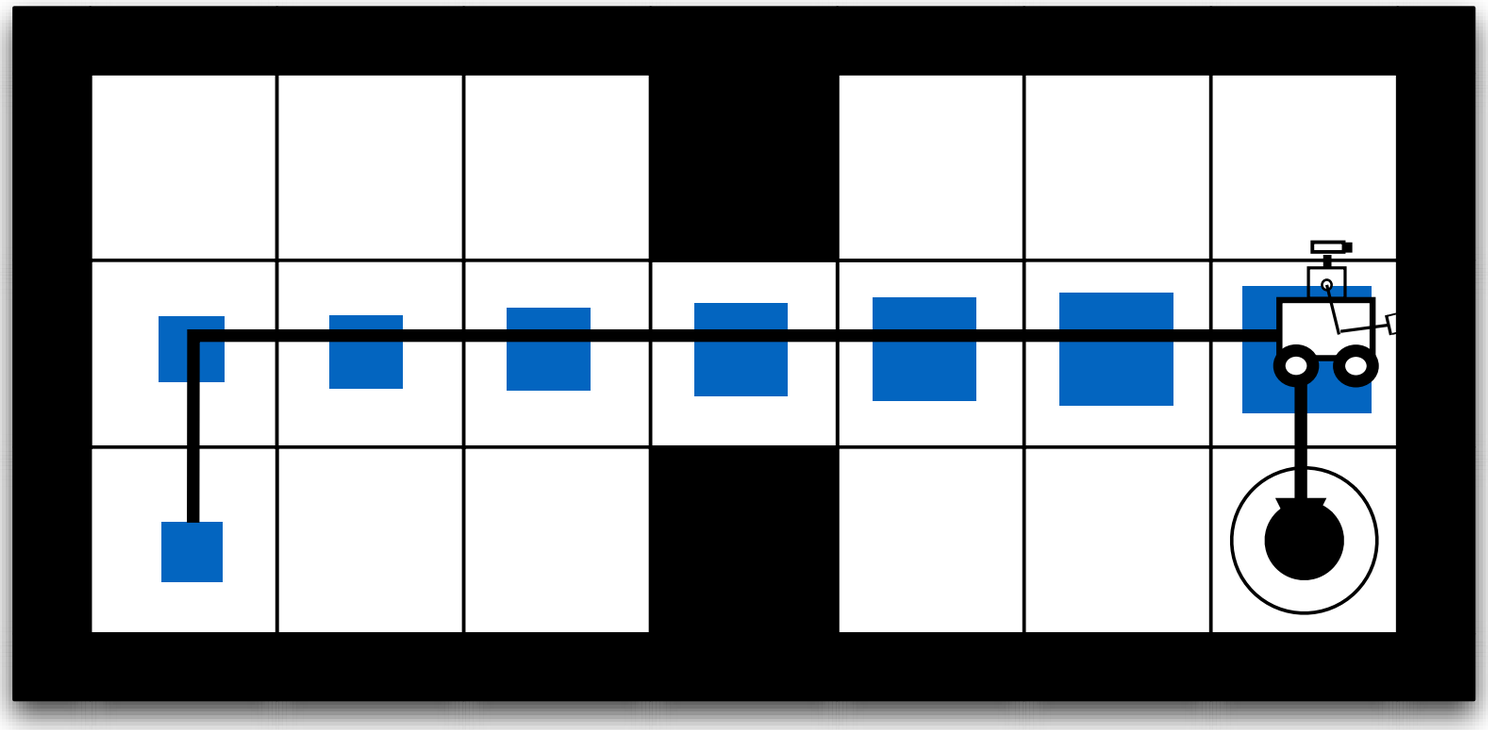
$$R^{(n)} = \sum_{i=0}^{n-1} \gamma^i r_i + \gamma^n Q(s_n, a_n)$$



Estimator:

$$R_{s_t}^{\lambda} = (1 - \lambda) \sum_{n=0}^{\infty} \lambda^n R_{s_t}^{(n+1)}$$

Sarsa(λ)



This is called the λ-return.

- At λ=0 we get Sarsa, at λ=1 we get MC.
- Intermediate values of λ usually best.
- TD(λ) family of algorithms

Sarsa(λ): Implementation

Each state has eligibility trace e(s, a).

At time t:

$e(s_t, a_t) = 1$

$e(s, a) = \gamma\lambda e(s, a)$, for all other *(s, a)* pairs.

At end of episode:

e(s, a) = 0, for all *(s, a)* pairs.

When updating:

- Compute δ as before
- $Q(s, a) = Q(s, a) + \alpha\delta e(s, a)$, for each *(s, a)* pair.

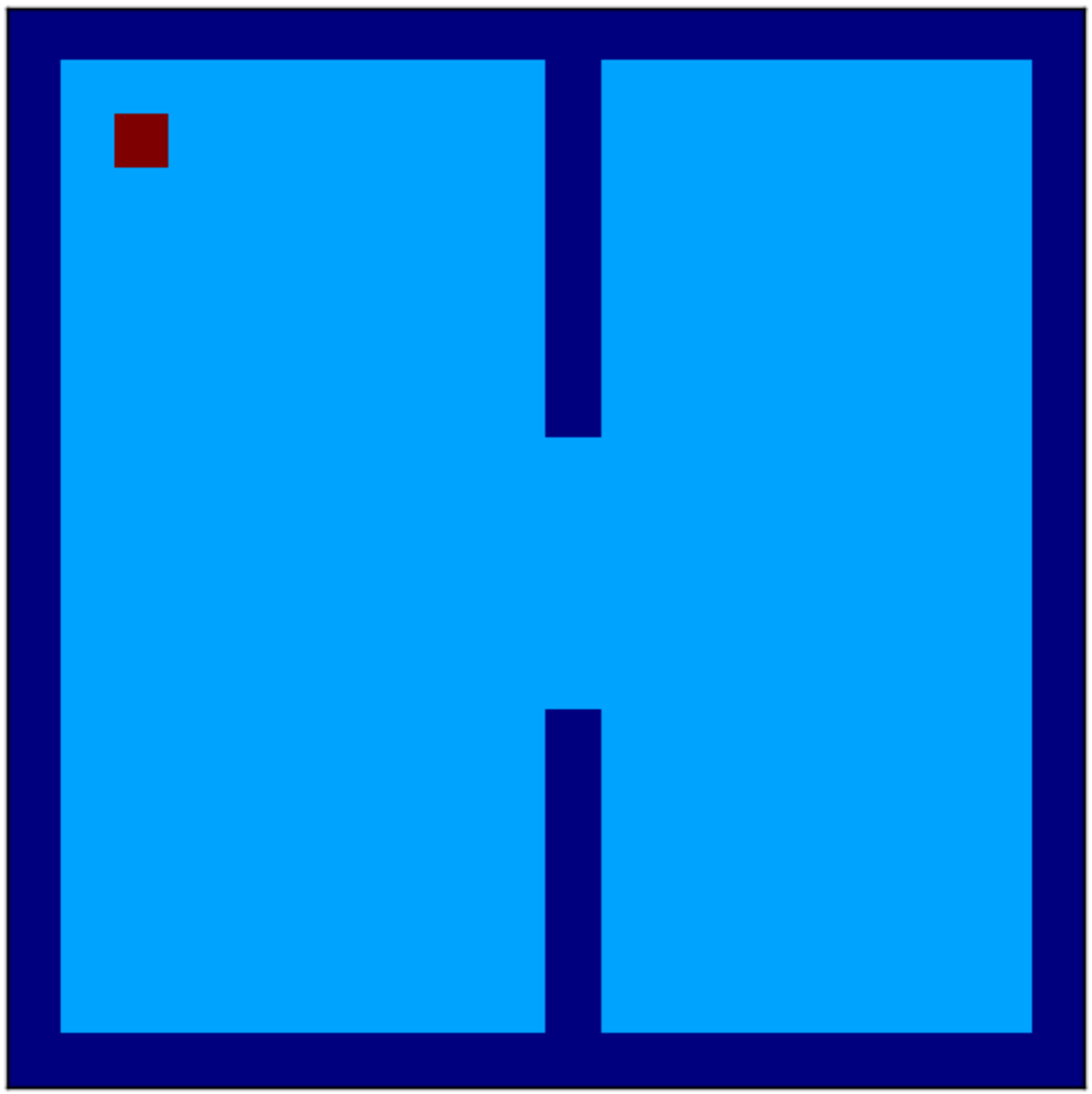
Sarsa(λ): Implementation

1. Initialize Q[s][a] = 0, for all (s, a)
2. Initialize e[s][a] = 0, for all (s, a)
2. For *n* episodes
   - observe state s_t
   - select a_t = argmax$_a$ Q(s$_t$, a)
   - observe transition $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$
     - compute TD error $\delta = r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$
     - e(s$_t$, a$_t$) = 1; other e(s, a) = γλe(s, a)
   - update Q: $Q(s,a) = Q(s,a) + \alpha\delta e(s,a), \forall s, a$
   - if not end of episode, repeat
   - if end of episode, e[s][a] = 0 for all (s,a)

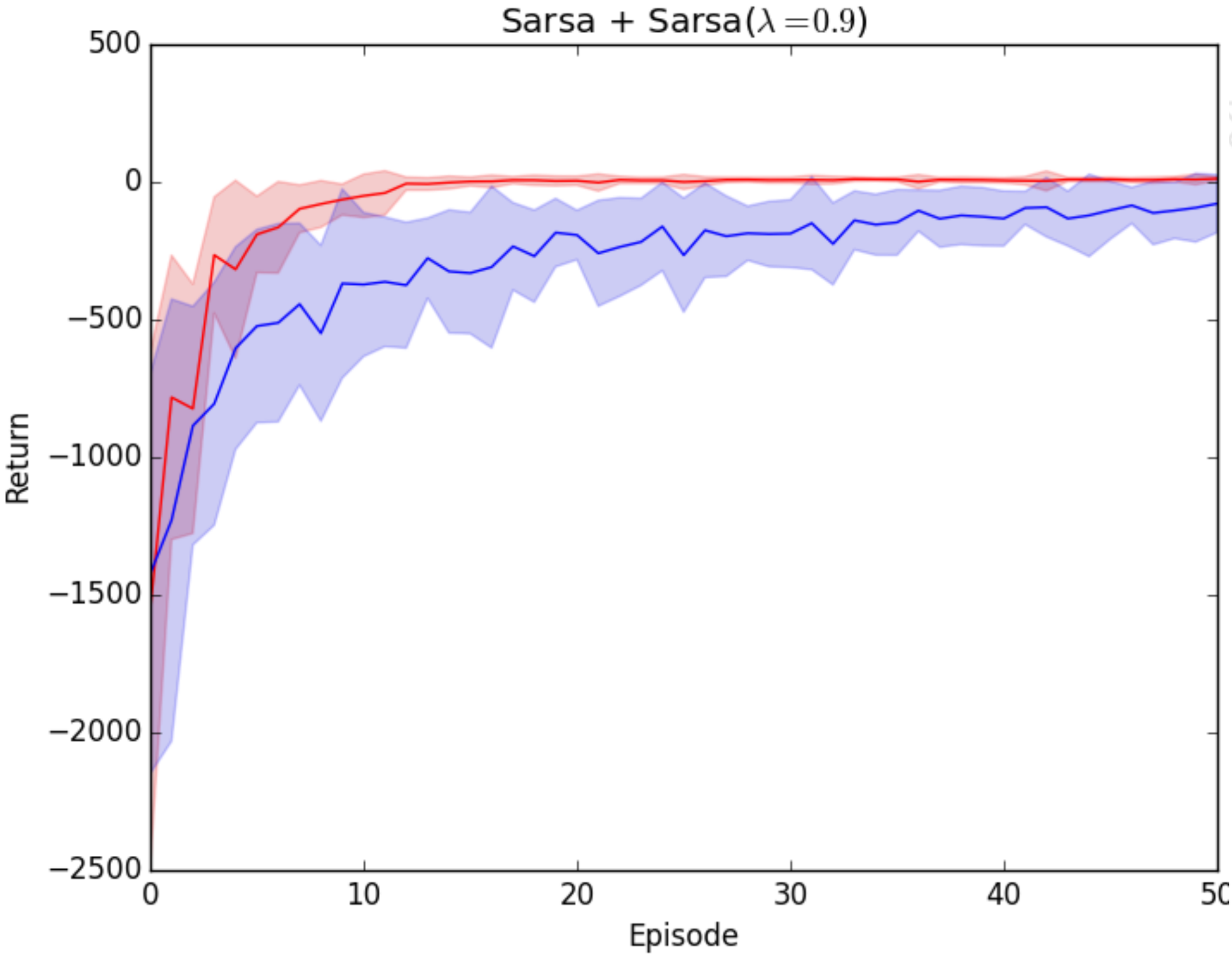
zero by def.
if *s* is absorbing

don't forget!

Sarsa(λ)



Sarsa(λ)


Sarsa + Sarsa($\lambda = 0.9$)
500
0
−500
−1000
−1500
−2000
−2500
Return
0
10
20
30
40
50
Episode

Next Week: More Realism