Reinforcement Learning I: Theory

Quentin Huys

Translational Neuromodeling Unit, ETH and University of Zurich
University Hospital of Psychiatry Zurich

Advanced Course in Computational Neuroscience, Bedlewo, Poland, August 2013

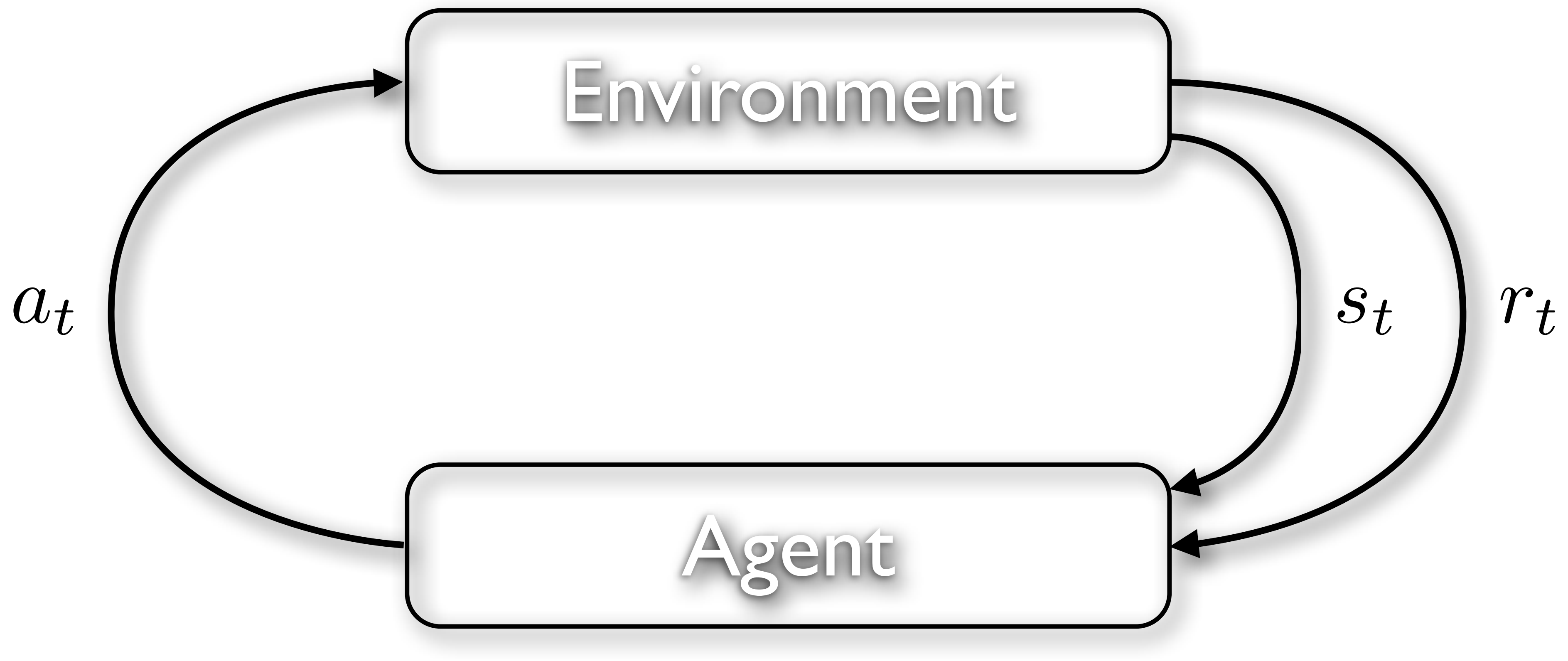
Overview

- Reinforcement learning: rough overview
  - mainly following Sutton & Barto 1998
- Learning theory
  - classical & instrumental conditioning
- Dopamine
  - prediction errors and more
- Fitting behaviour with RL models
  - some applied tips & tricks

Types of learning

- Supervised
- Unsupervised
- Reinforcement learning

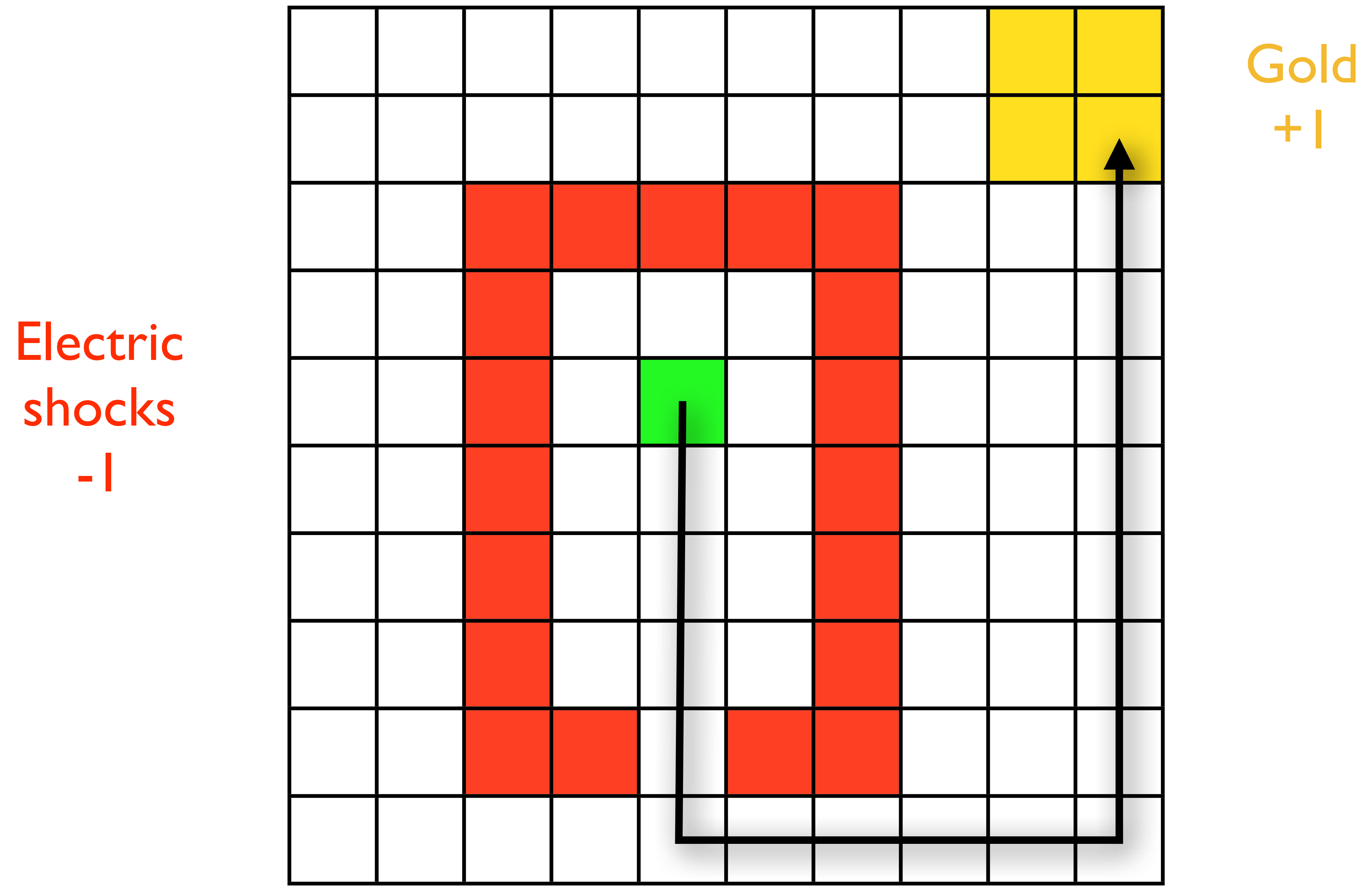
Setup



$$\{a_t\} \leftarrow \underset{\{a_t\}}{\operatorname{argmax}} \sum_{t=1}^{\infty} r_t$$

After Sutton and Barto 1998

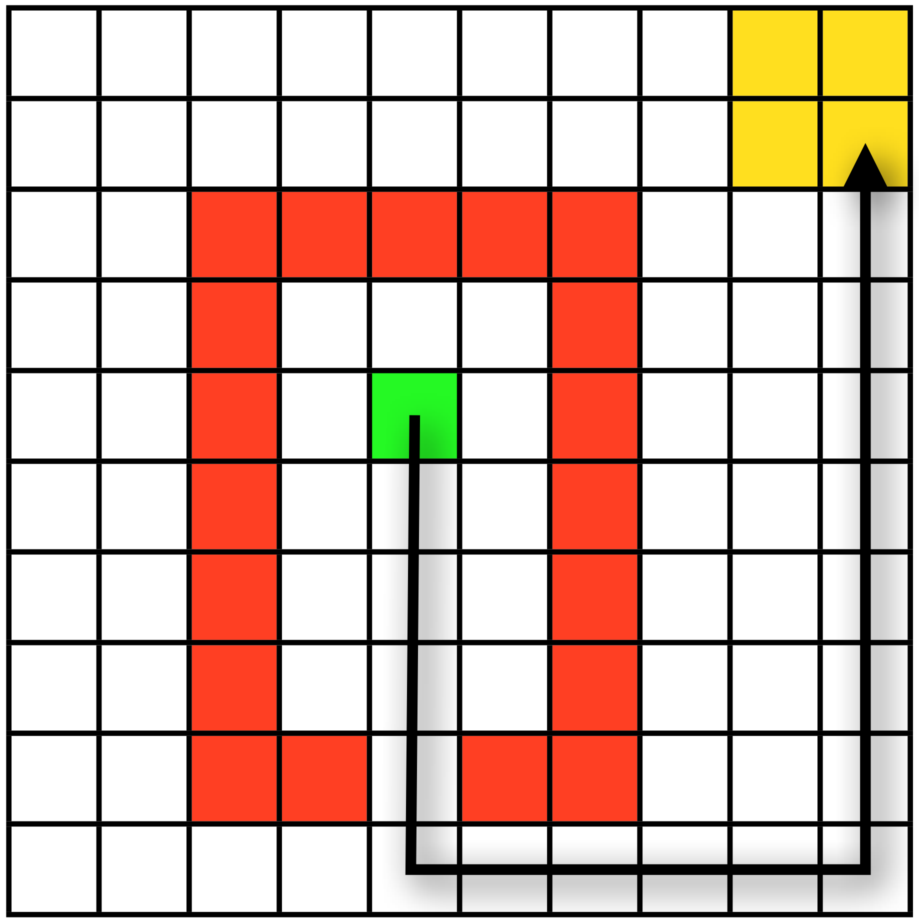
State space



A Markov Decision Problem

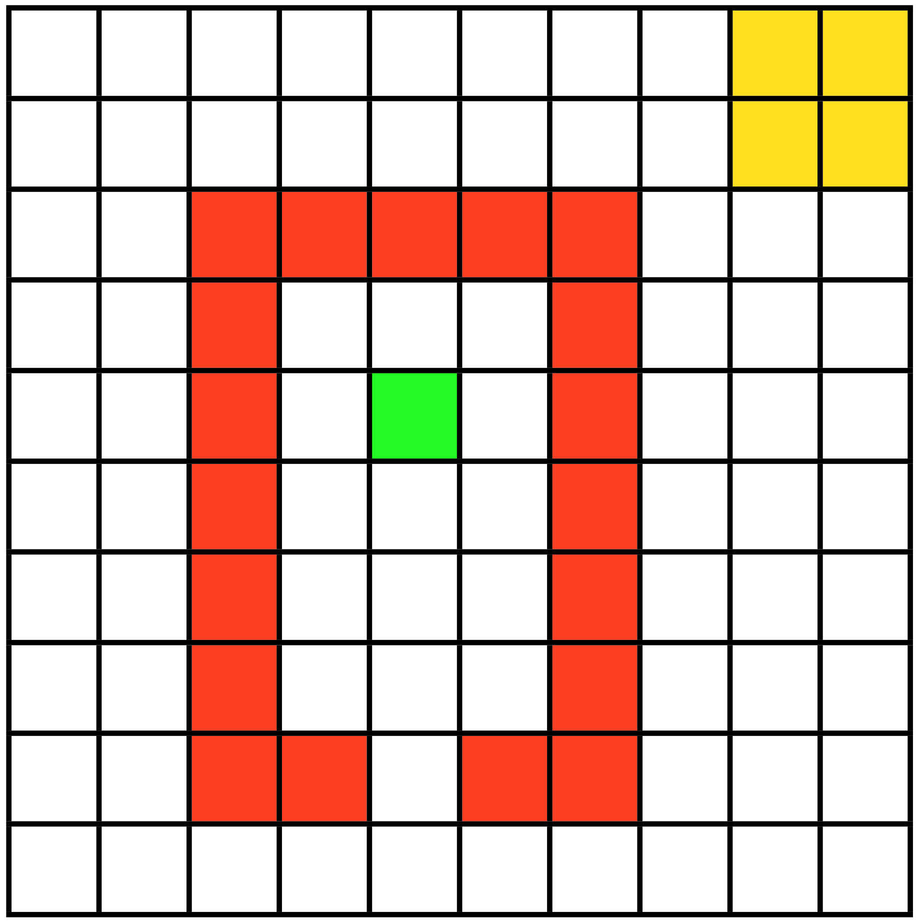
$$\boxed{s_t \in \mathcal{S}}$$

$$
\begin{aligned}
a_t &\in \mathcal{A} \\
\mathcal{T}^a_{ss'} &= p(s_{t+1}|s_t, a_t) \\
r_t &\sim \mathcal{R}(s_{t+1}, a_t, s_t) \\
\pi(a|s) &= p(a|s)
\end{aligned}
$$



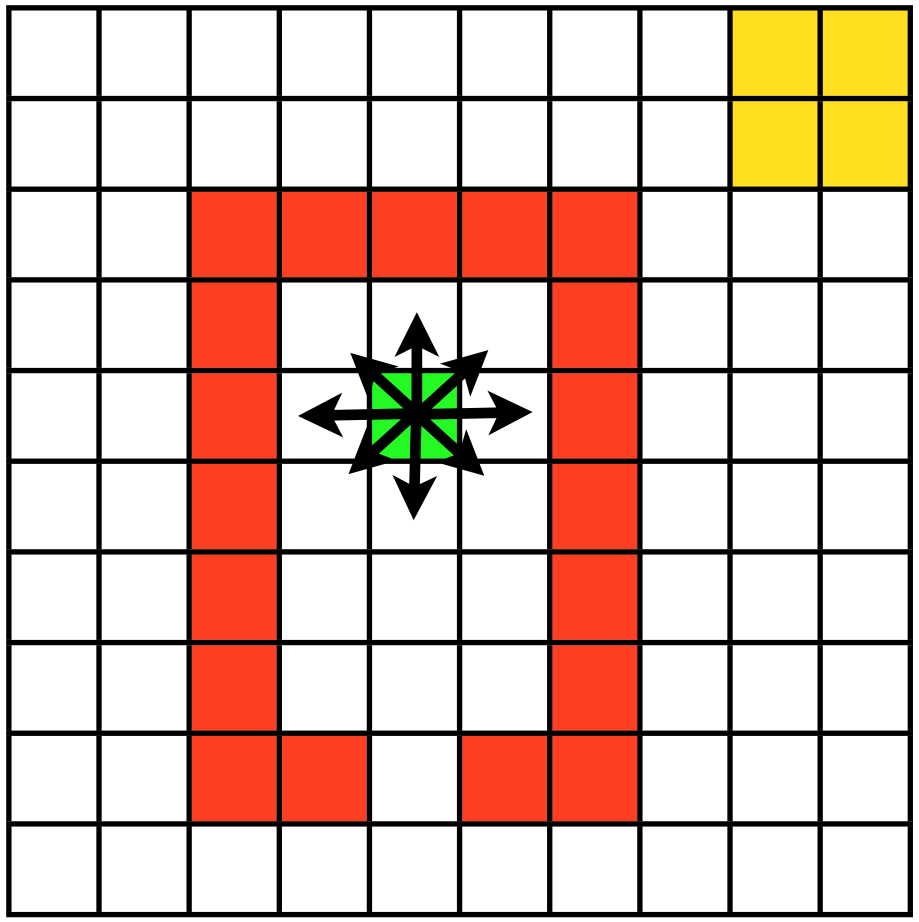
MDP

$$
\begin{aligned}
s_t &\in \mathcal{S} \\
\boxed{\begin{aligned} a_t &\in \mathcal{A} \\ \mathcal{T}^a_{ss'} &= p(s_{t+1}|s_t, a_t) \end{aligned}} \\
r_t &\sim \mathcal{R}(s_{t+1}, a_t, s_t) \\
\pi(a|s) &= p(a|s)
\end{aligned}
$$

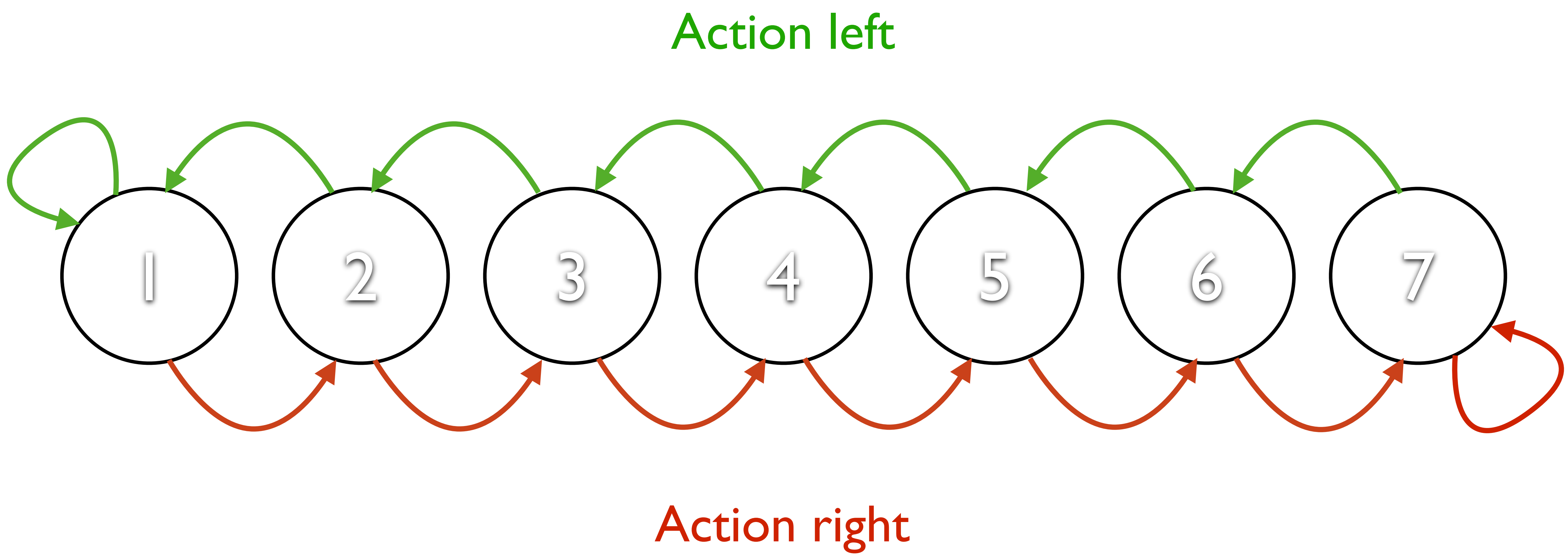


MDP

$$\begin{aligned}
s_t &\in \mathcal{S} \\
\boxed{\begin{aligned} a_t &\in \mathcal{A} \\ \mathcal{T}^a_{ss'} &= p(s_{t+1}|s_t, a_t) \end{aligned}} \\
r_t &\sim \mathcal{R}(s_{t+1}, a_t, s_t) \\
\pi(a|s) &= p(a|s)
\end{aligned}$$

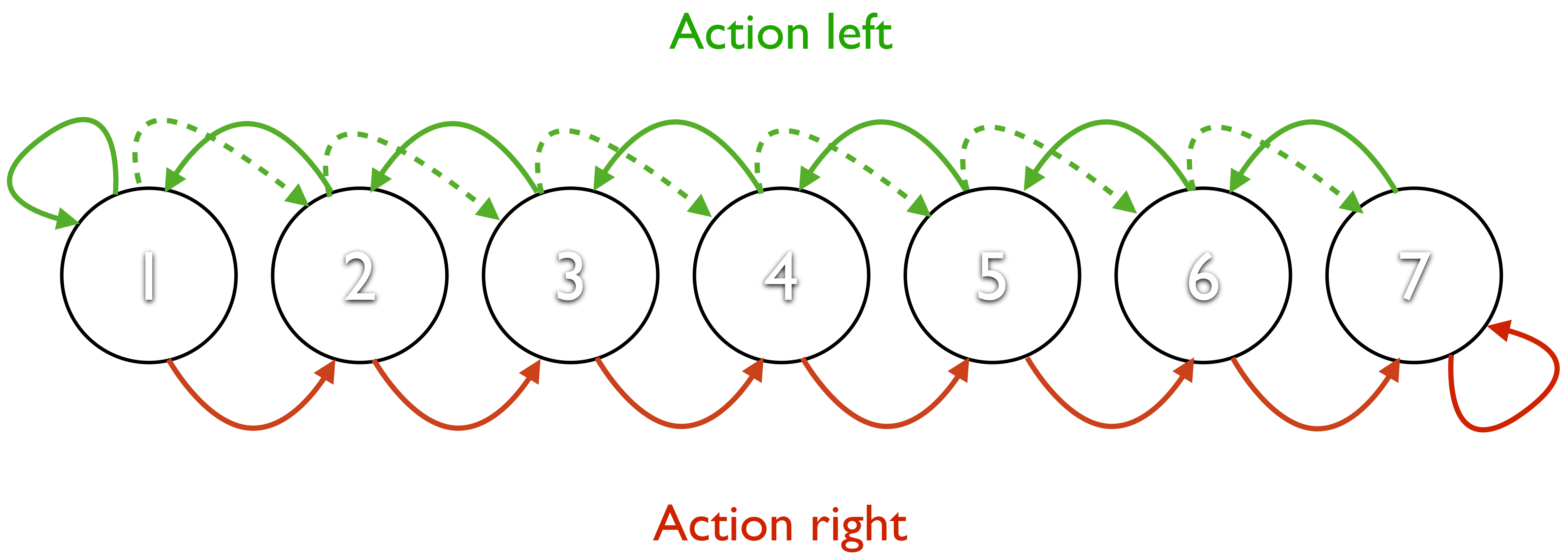


Actions



$$T^{\text{left}} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad T^{\text{right}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

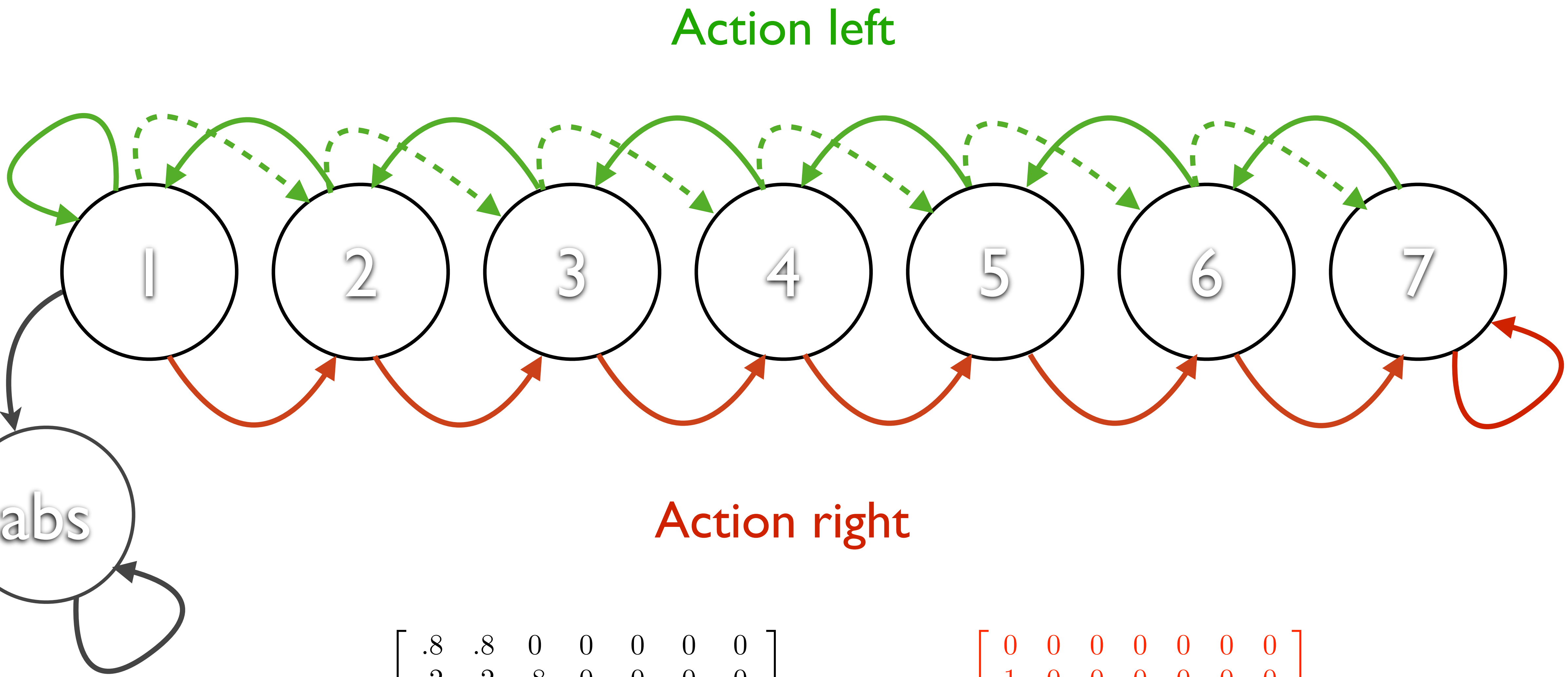
Actions



$$T^{\text{left}} = \begin{bmatrix} .8 & .8 & 0 & 0 & 0 & 0 & 0 \\ .2 & .2 & .8 & 0 & 0 & 0 & 0 \\ 0 & 0 & .2 & .8 & 0 & 0 & 0 \\ 0 & 0 & 0 & .2 & .8 & 0 & 0 \\ 0 & 0 & 0 & 0 & .2 & .8 & 0 \\ 0 & 0 & 0 & 0 & 0 & .2 & .8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad T^{\text{right}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Noisy: plants, environments, agent

Actions



$$T^{\text{left}} = \begin{bmatrix} .8 & .8 & 0 & 0 & 0 & 0 & 0 \\ .2 & .2 & .8 & 0 & 0 & 0 & 0 \\ 0 & 0 & .2 & .8 & 0 & 0 & 0 \\ 0 & 0 & 0 & .2 & .8 & 0 & 0 \\ 0 & 0 & 0 & 0 & .2 & .8 & 0 \\ 0 & 0 & 0 & 0 & 0 & .2 & .8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad T^{\text{right}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Noisy: plants, environments, agent

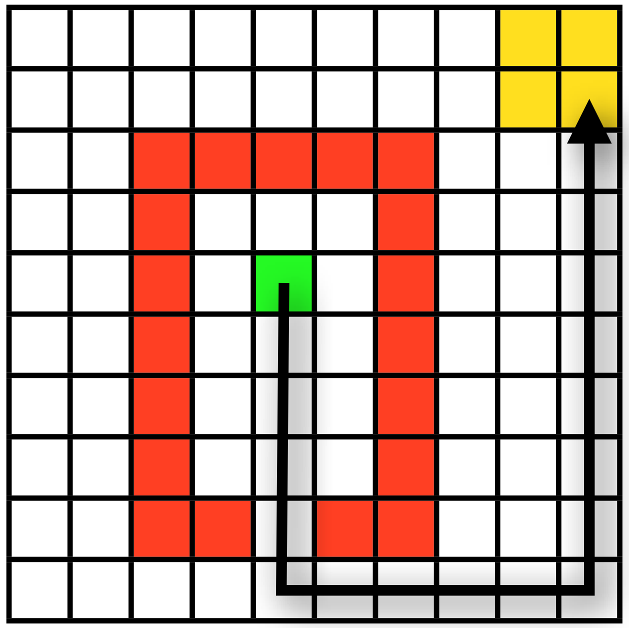
Absorbing state -> max eigenvalue < 1

Markovian dynamics

$$p(s_{t+1}|a_t, s_t, a_{t-1}, s_{t-1}, a_{t-2}, s_{t-2}, \cdots) = p(s_{t+1}|a_t, s_t)$$

$$\begin{tikzcd}
a_{t-2}, s_{t-2} \arrow[r] \arrow[r, bend left] \arrow[rr, bend left] & a_{t-1}, s_{t-1} \arrow[r] \arrow[r, bend right] & a_t, s_t
\end{tikzcd}$$

Velocity

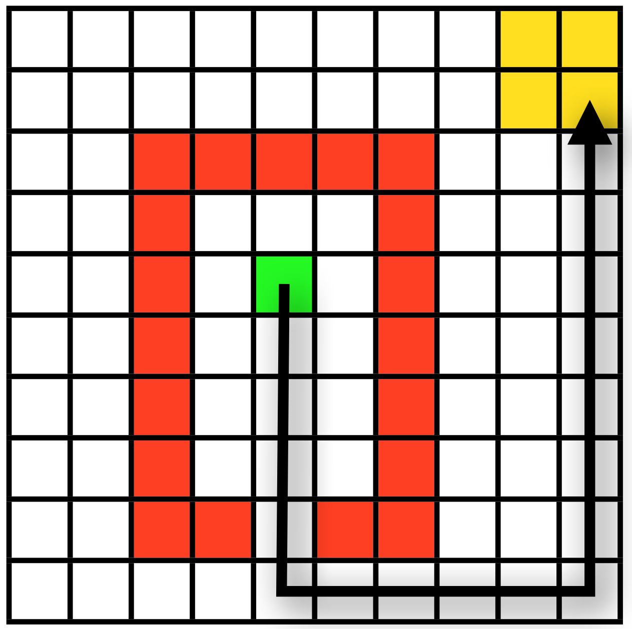


Markovian dynamics

$$p(s_{t+1}|a_t, s_t, a_{t-1}, s_{t-1}, a_{t-2}, s_{t-2}, \cdots) = p(s_{t+1}|a_t, s_t)$$

$$\begin{tikzcd} a_{t-2}, s_{t-2} \arrow[r] & a_{t-1}, s_{t-1} \arrow[r] & a_t, s_t \end{tikzcd}$$

Velocity



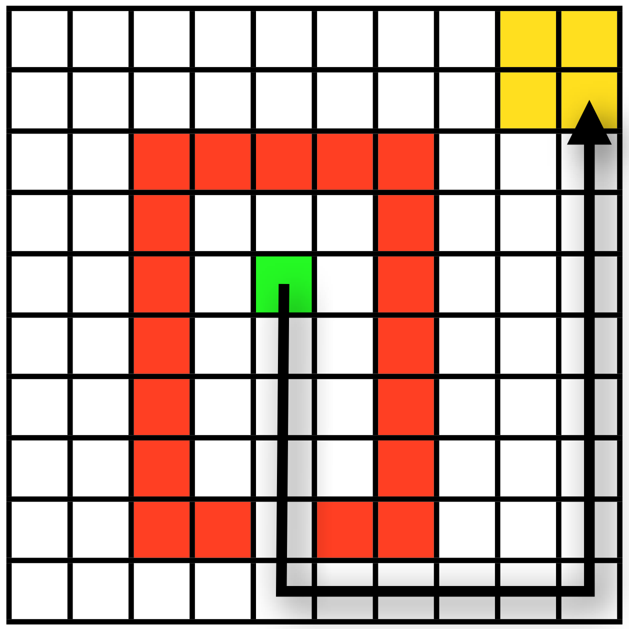
Markovian dynamics

$$p(s_{t+1}|a_t, s_t, a_{t-1}, s_{t-1}, a_{t-2}, s_{t-2}, \cdots) = p(s_{t+1}|a_t, s_t)$$

$$\begin{tikzcd} a_{t-2}, s_{t-2} \arrow[r] & a_{t-1}, s_{t-1} \arrow[r] & a_t, s_t \end{tikzcd}$$

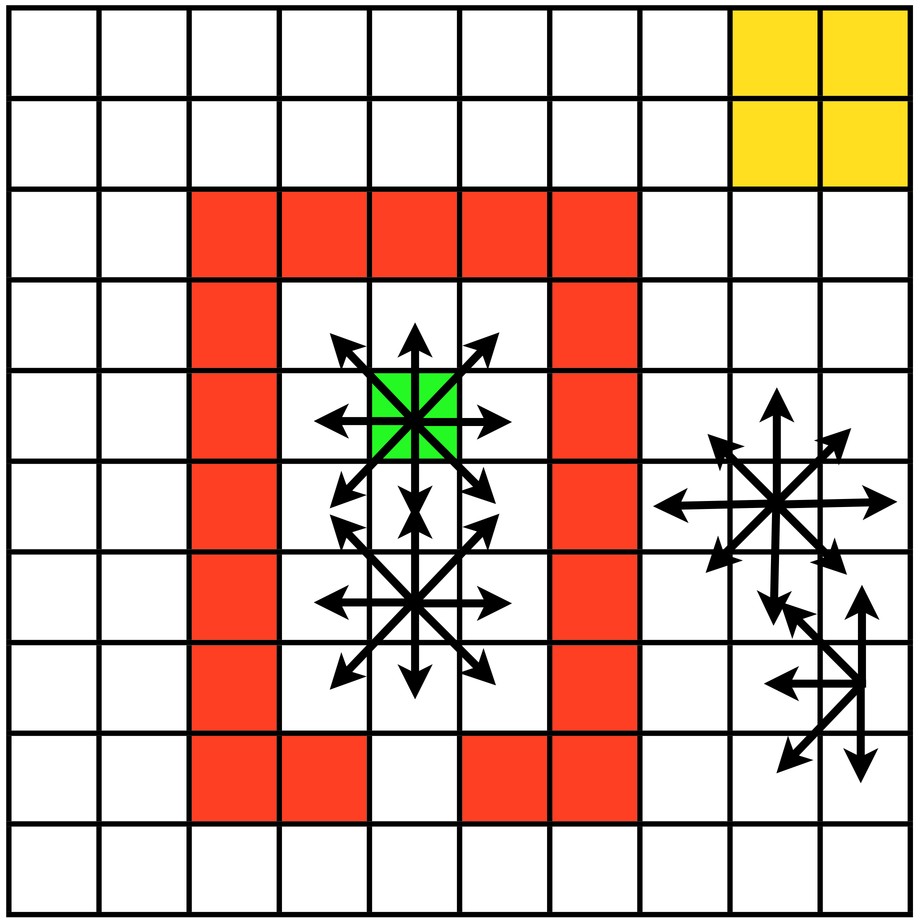
Velocity

$$s' = [\text{position}] \rightarrow s' = \begin{bmatrix} \text{position} \\ \text{velocity} \end{bmatrix}$$



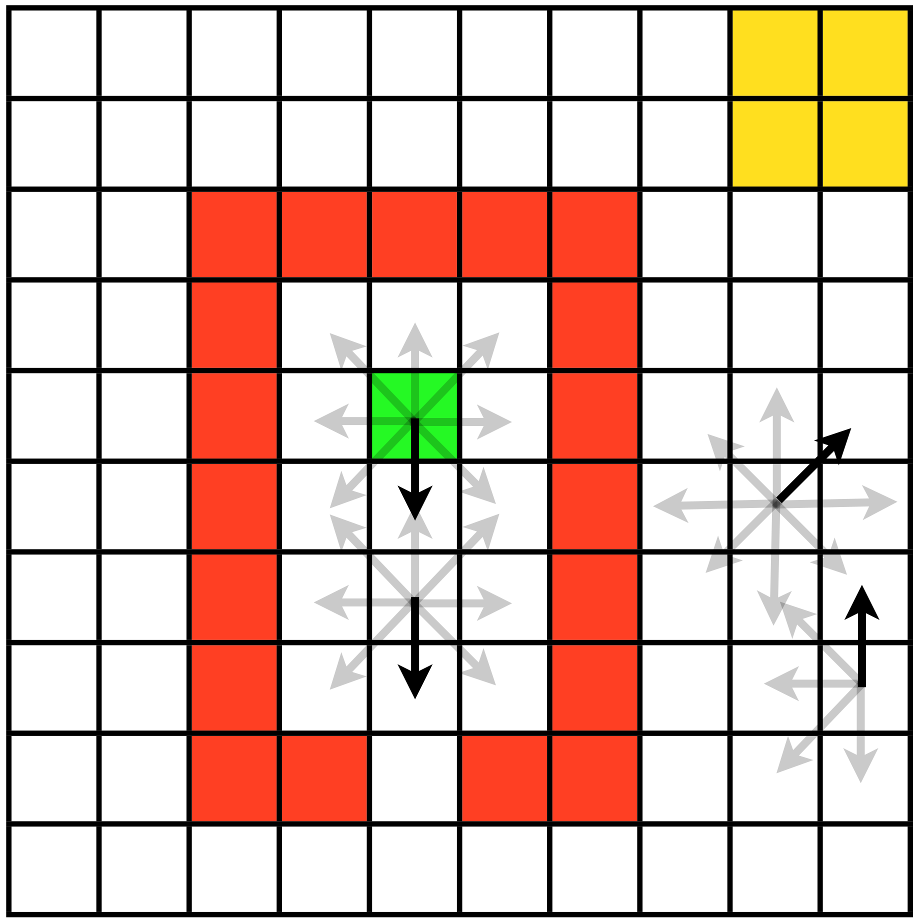
MDP

$$
\begin{aligned}
s_t &\in \mathcal{S} \\
a_t &\in \mathcal{A} \\
\mathcal{T}^a_{ss'} &= p(s_{t+1}|s_t, a_t) \\
r_t &\sim \mathcal{R}(s_{t+1}, a_t, s_t) \\
\boxed{\pi(a|s) = p(a|s)}
\end{aligned}
$$



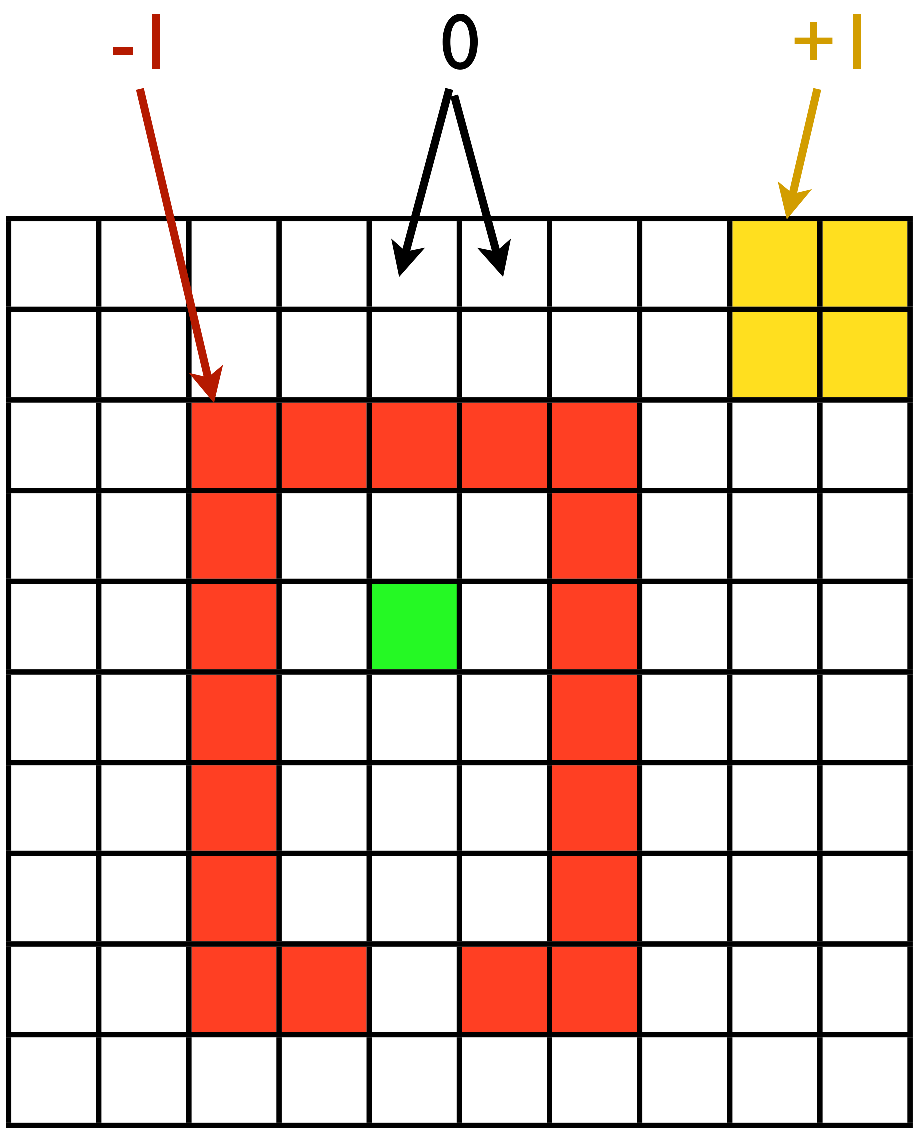
MDP

$$
\begin{aligned}
s_t &\in \mathcal{S} \\
a_t &\in \mathcal{A} \\
\mathcal{T}^a_{ss'} &= p(s_{t+1}|s_t, a_t) \\
r_t &\sim \mathcal{R}(s_{t+1}, a_t, s_t) \\
\boxed{\pi(a|s) = p(a|s)}
\end{aligned}
$$



MDP

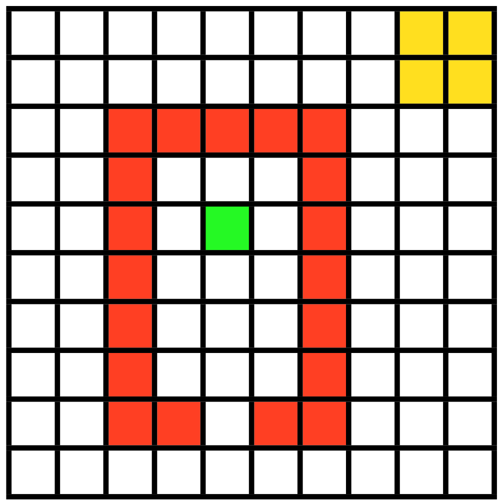
$$
\begin{aligned}
s_t &\in \mathcal{S} \\
a_t &\in \mathcal{A} \\
\mathcal{T}^a_{ss'} &= p(s_{t+1}|s_t, a_t) \\
\boxed{r_t \sim \mathcal{R}(s_{t+1}, a_t, s_t)} \\
\pi(a|s) &= p(a|s)
\end{aligned}
$$



Tall orders

- Aim: maximise total future reward

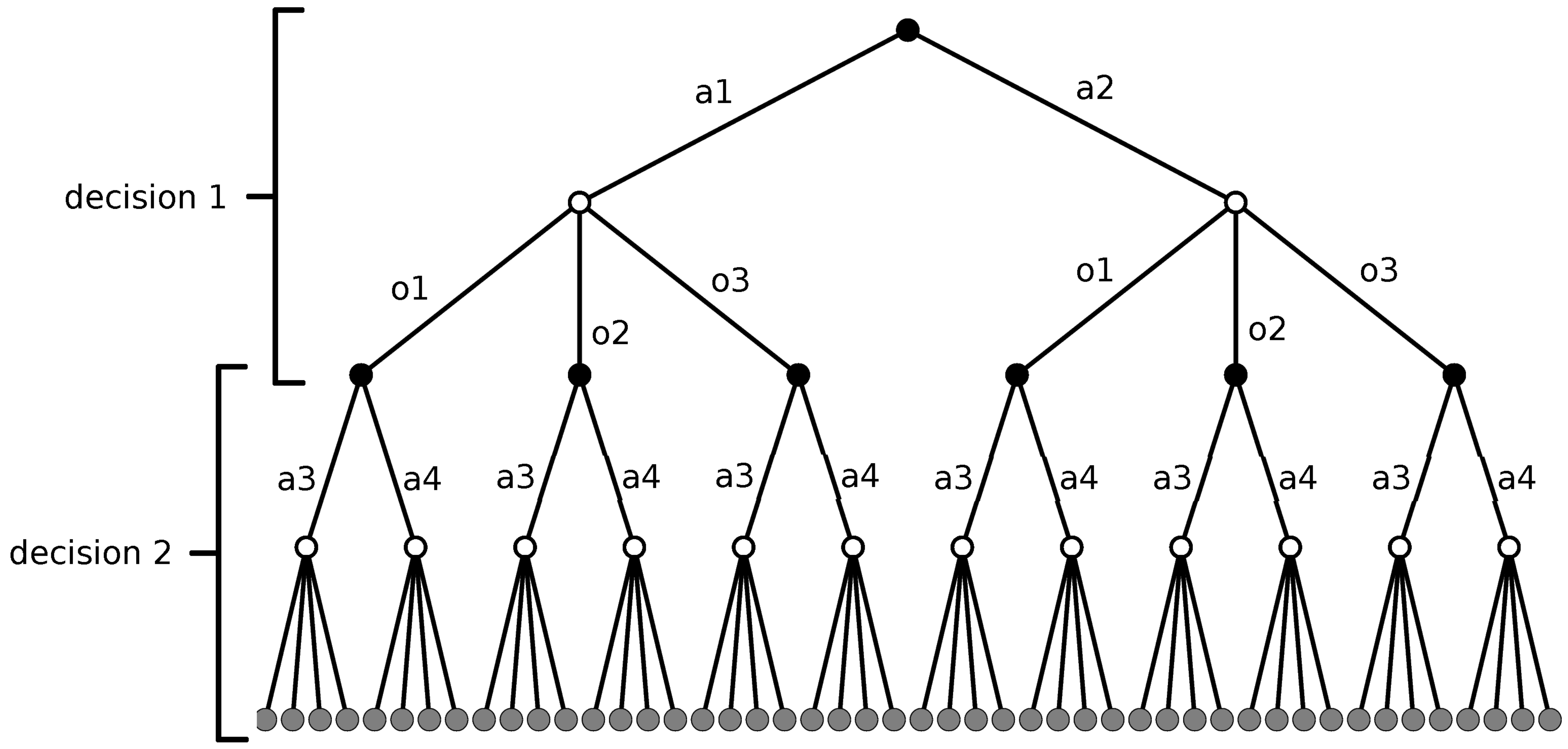
$$\sum_{t=1}^{\infty} r_t$$



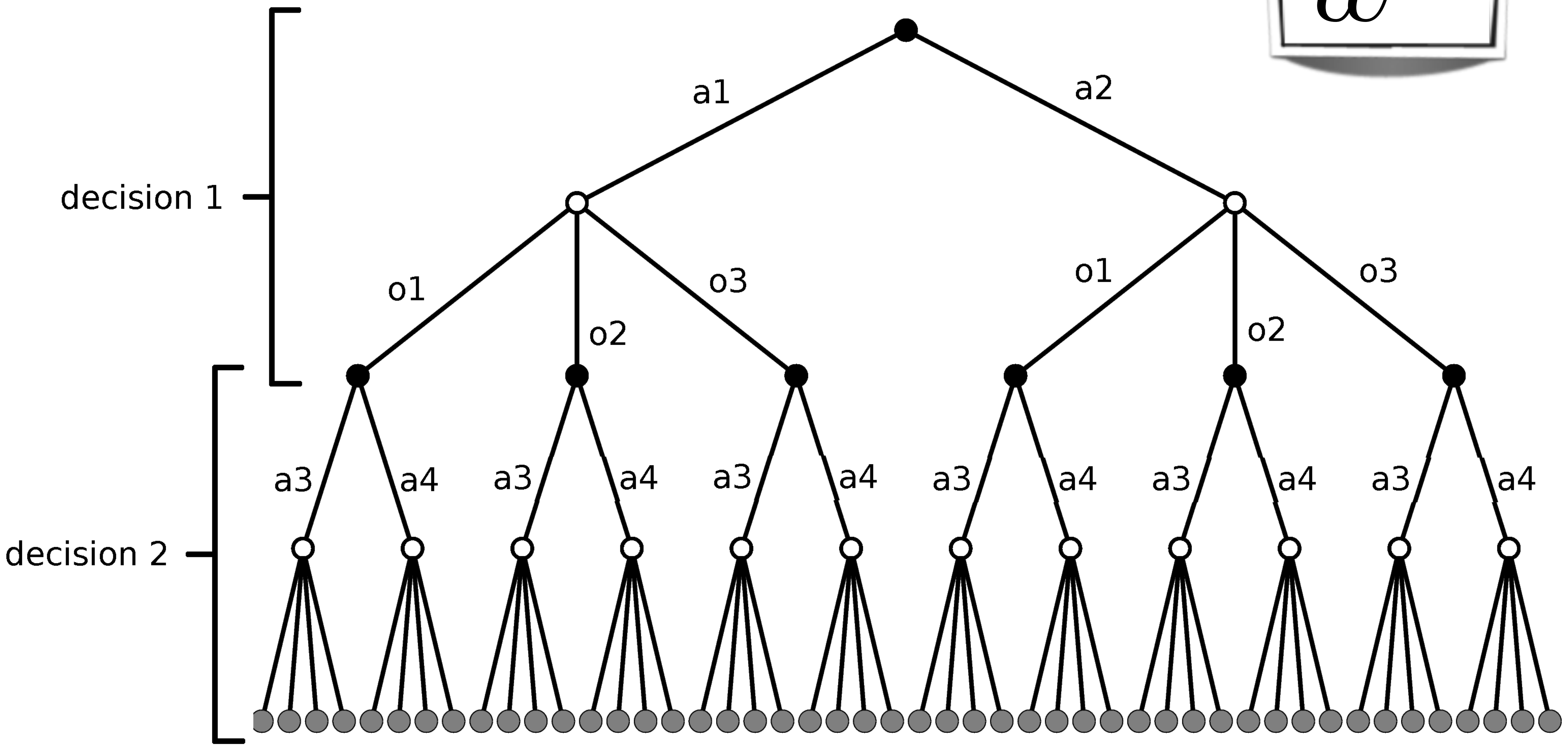
- i.e. we have to sum over paths through the future and weigh each by its probability

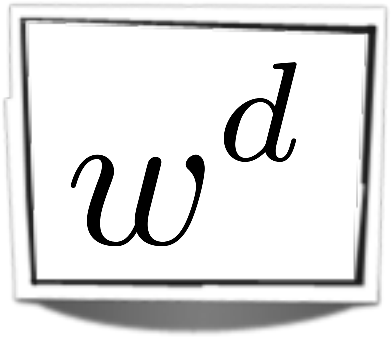
- Best policy achieves best long-term reward

Exhaustive tree search



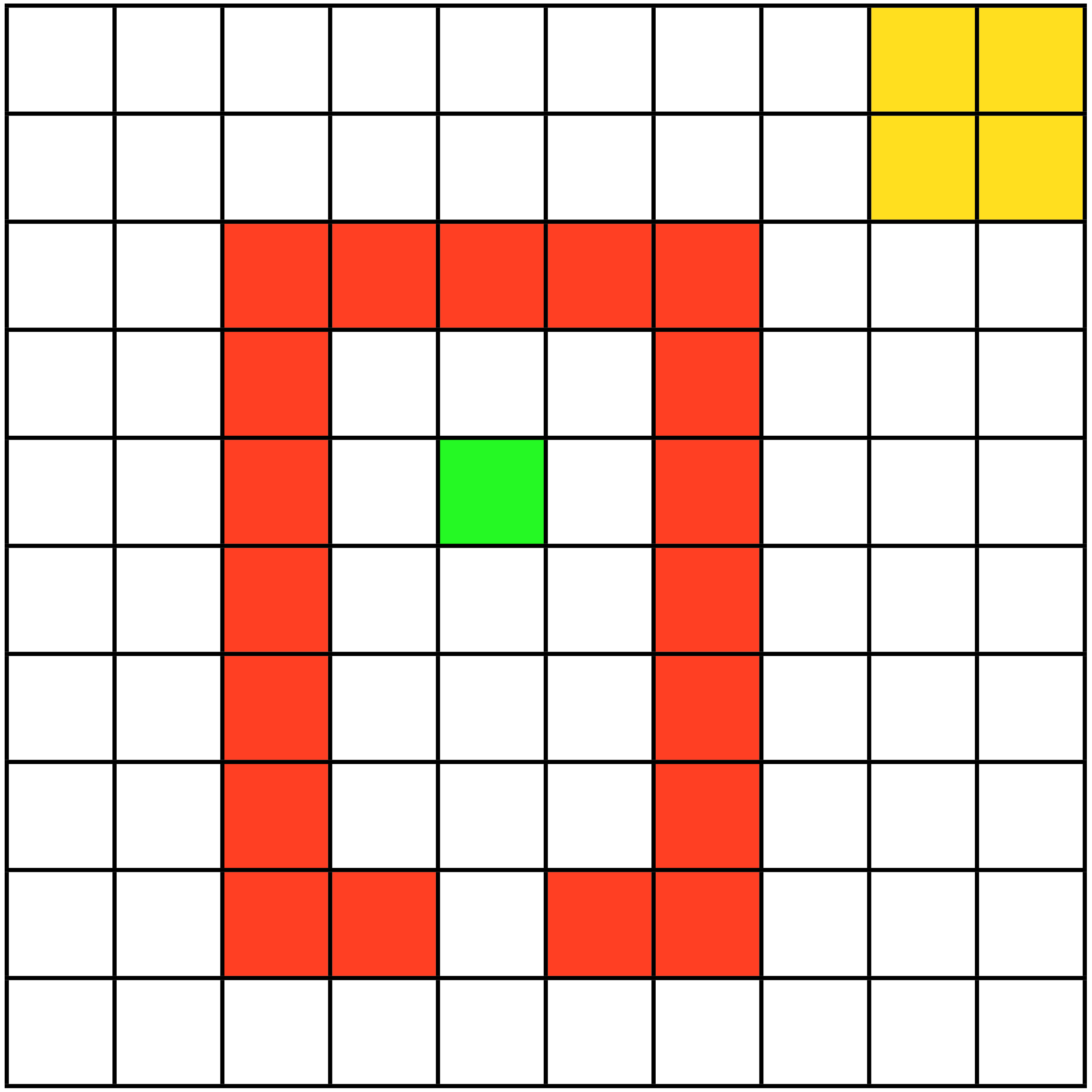
Exhaustive tree search




w^d

Decision tree

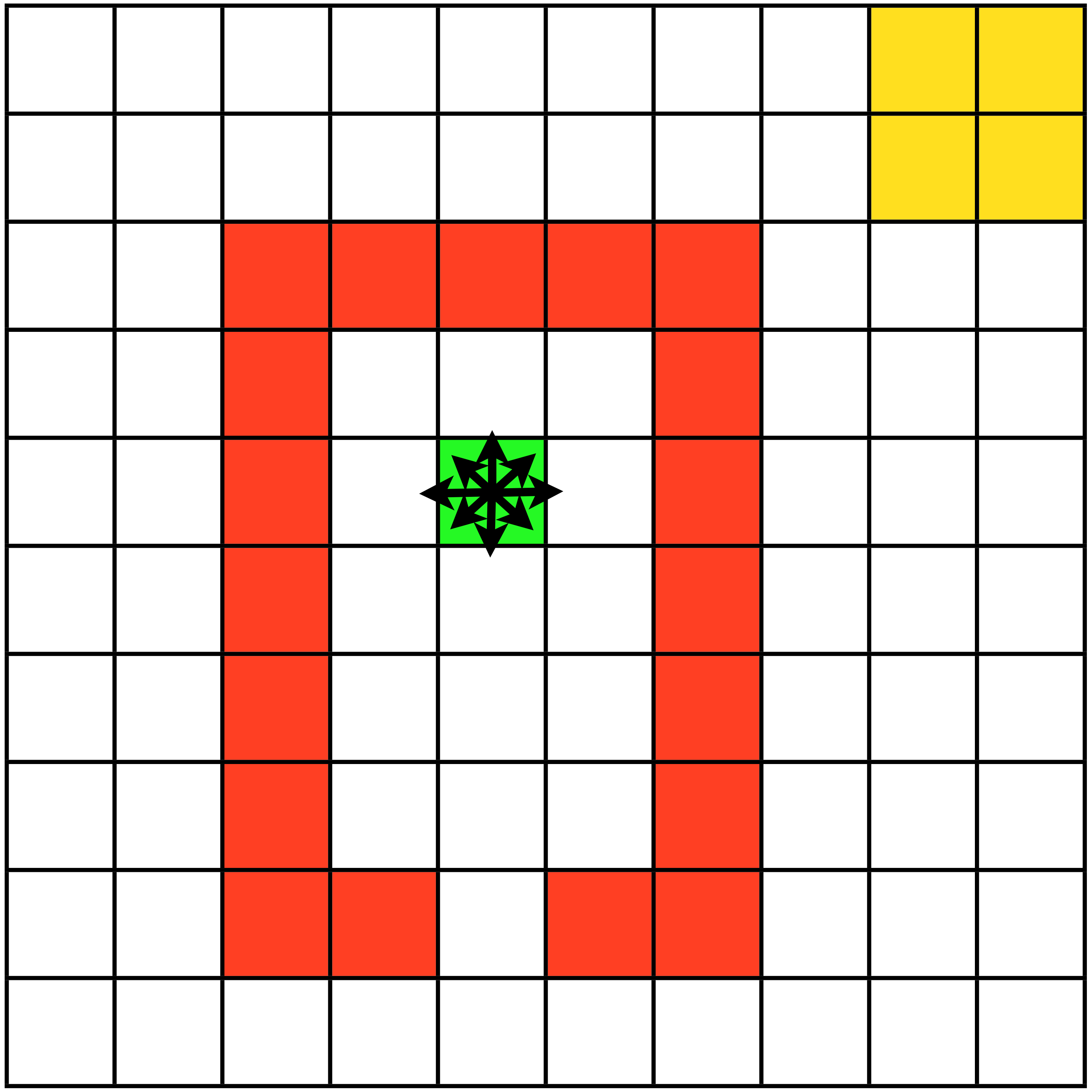
$$\sum_{t=1}^{\infty} r_t$$



Decision tree

$$\sum_{t=1}^{\infty} r_t$$

8

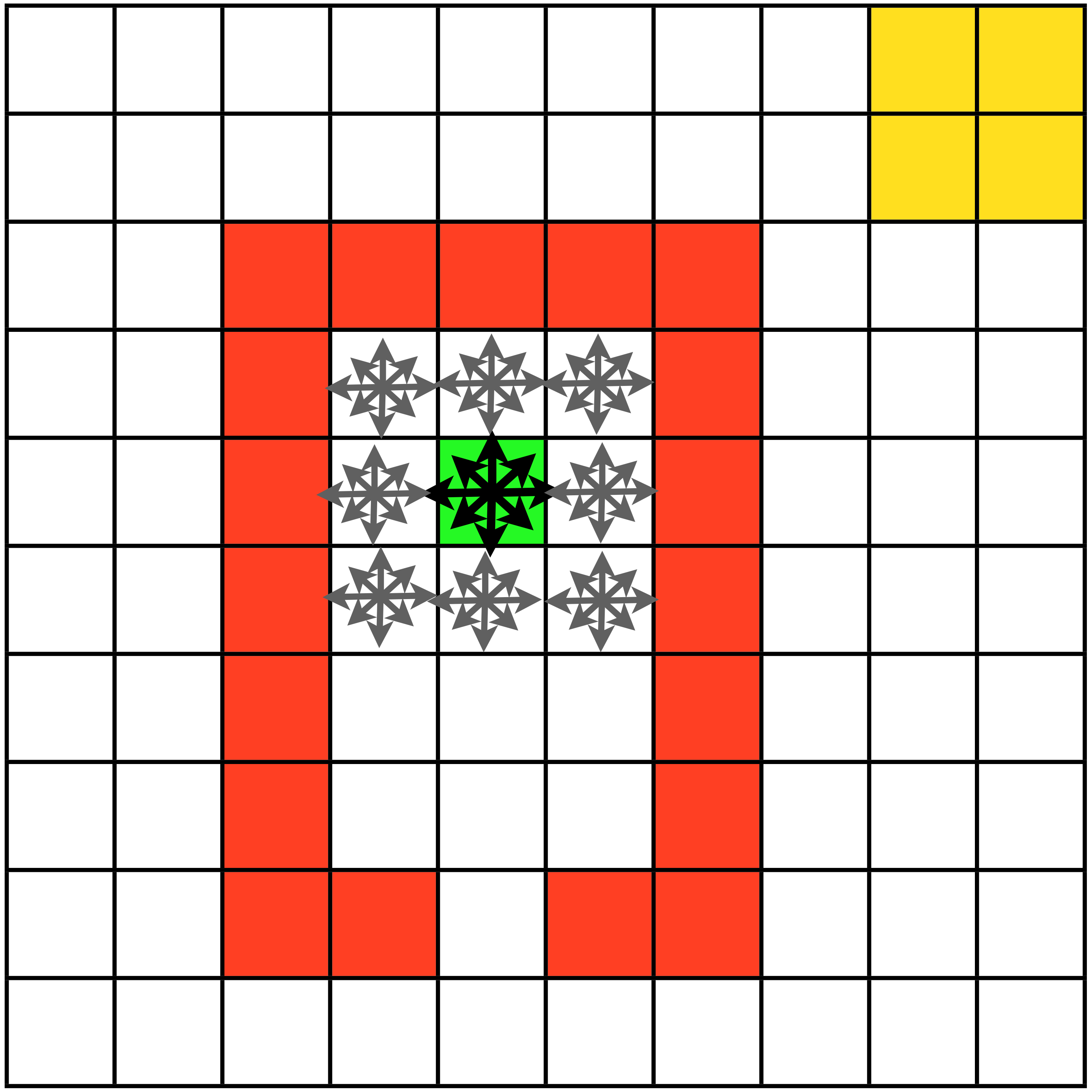


Decision tree

$$\sum_{t=1}^{\infty} r_t$$

8

64



Decision tree

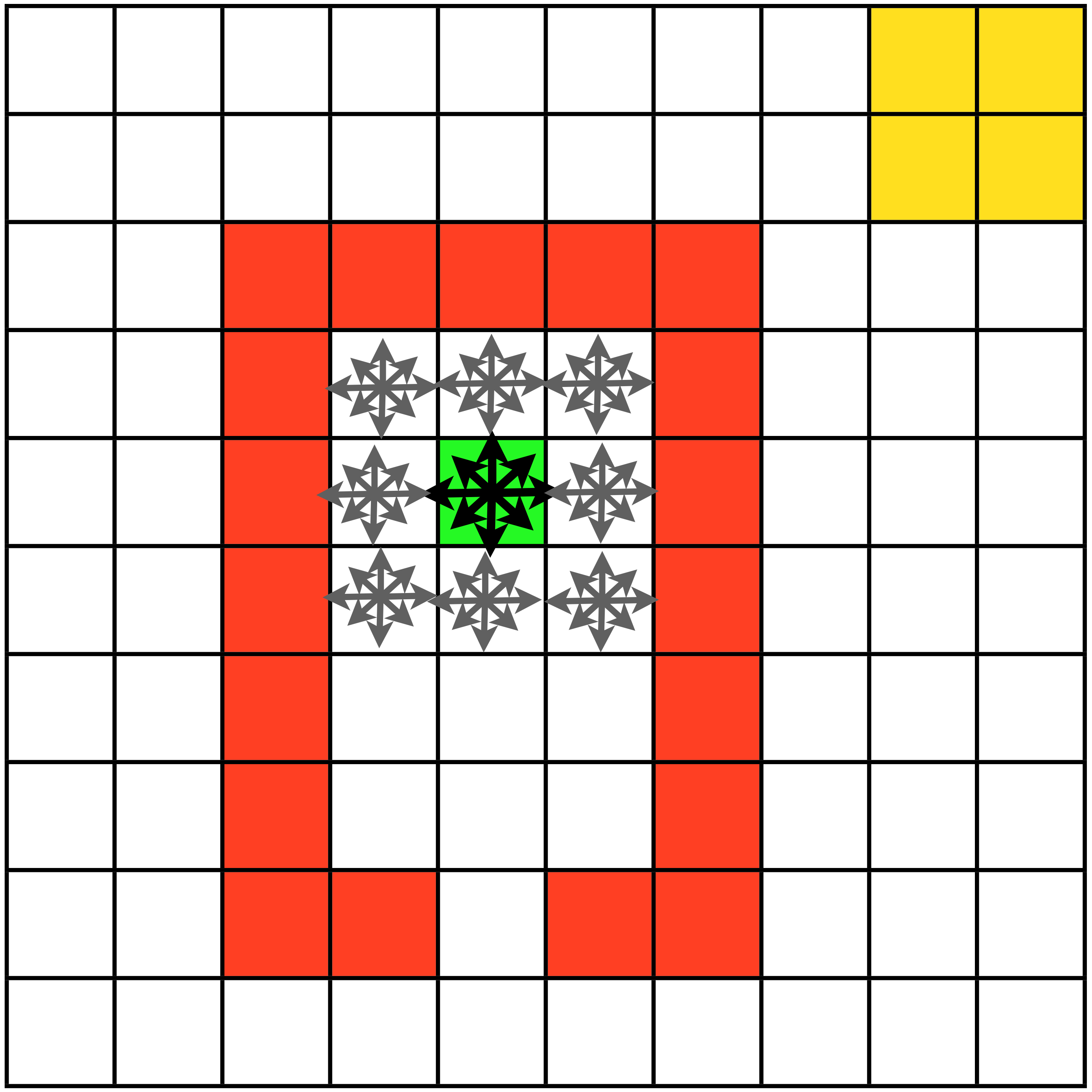
$$\sum_{t=1}^{\infty} r_t$$

8

64

512

...



Policy for this talk

- Pose the problem mathematically
- Policy evaluation
- Policy iteration
- Monte Carlo techniques: experience samples
- TD learning

$$\begin{tikzcd} & \text{Policy} & \\ \text{Evaluate} \arrow[rr, shift left] & & \text{Update} \arrow[ll, shift left] \end{tikzcd}$$

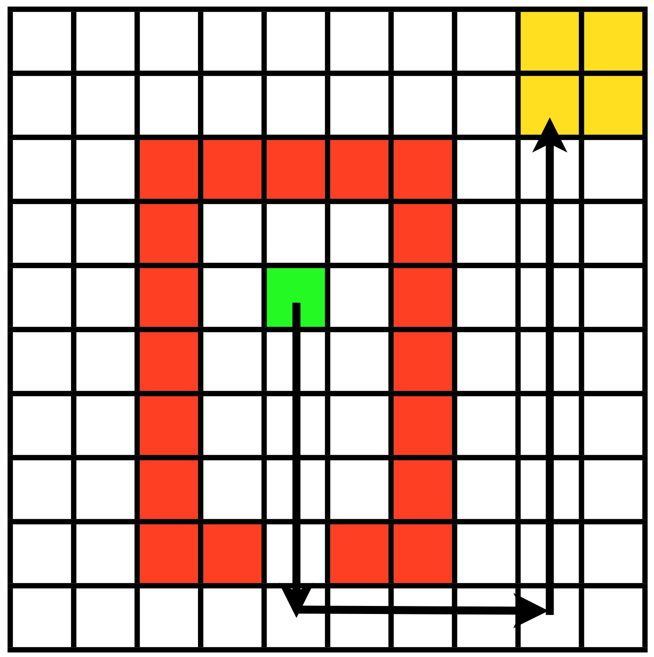
Evaluating a policy

- Aim: maximise total future reward

$$\sum_{t=1}^{\infty} r_t$$

- To know which is best, evaluate it first
- The policy determines the expected reward from each state

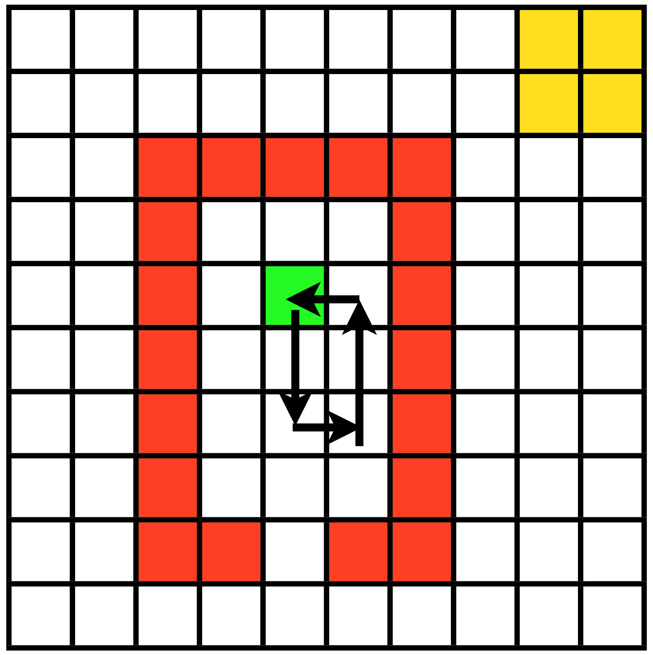
$$\mathcal{V}^{\pi}(s_1) \quad = \quad \mathbb{E}\left[\sum_{t=1}^{\infty} r_t | s_1 = 1, a_t \sim \pi\right]$$



Discounting

- Given a policy, each state has an expected value

$$\mathcal{V}^\pi(s_1) \quad = \quad \mathbb{E}\left[\sum_{t=1}^{\infty} r_t | s_1 = 1, a_t \sim \pi\right]$$



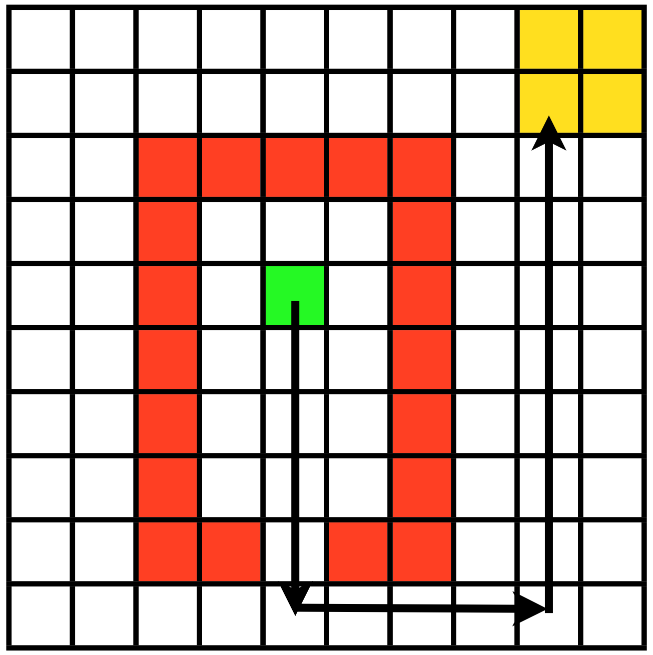
- But: $\sum_{t=0}^{\infty} r_t = \infty$
- Episodic $\sum_{t=0}^{T} r_t < \infty$
- Discounted
  - infinite horizons $\sum_{t=0}^{\infty} \gamma^t r_t < \infty$
  - finite, exponentially distributed horizons

$$\sum_{t=0}^{T} \gamma^t r_t \qquad\qquad T \sim \frac{1}{\tau} e^{t/\tau}$$

Discounting

- Given a policy, each state has an expected value

$$\mathcal{V}^\pi(s_1) \quad = \quad \mathbb{E}\left[\sum_{t=1}^{\infty} r_t | s_1 = 1, a_t \sim \pi\right]$$



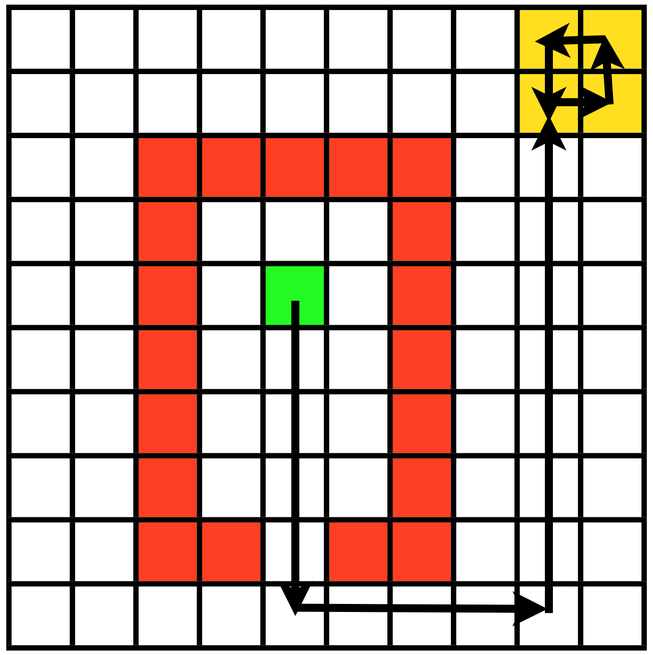
- But: $\sum_{t=0}^{\infty} r_t = \infty$
- Episodic $\sum_{t=0}^{T} r_t < \infty$
- Discounted
  - infinite horizons $\sum_{t=0}^{\infty} \gamma^t r_t < \infty$
  - finite, exponentially distributed horizons

$$\sum_{t=0}^{T} \gamma^t r_t \qquad\qquad T \sim \frac{1}{\tau} e^{t/\tau}$$

Discounting

- Given a policy, each state has an expected value

$$\mathcal{V}^\pi(s_1) \quad = \quad \mathbb{E}\left[\sum_{t=1}^{\infty} r_t | s_1 = 1, a_t \sim \pi\right]$$



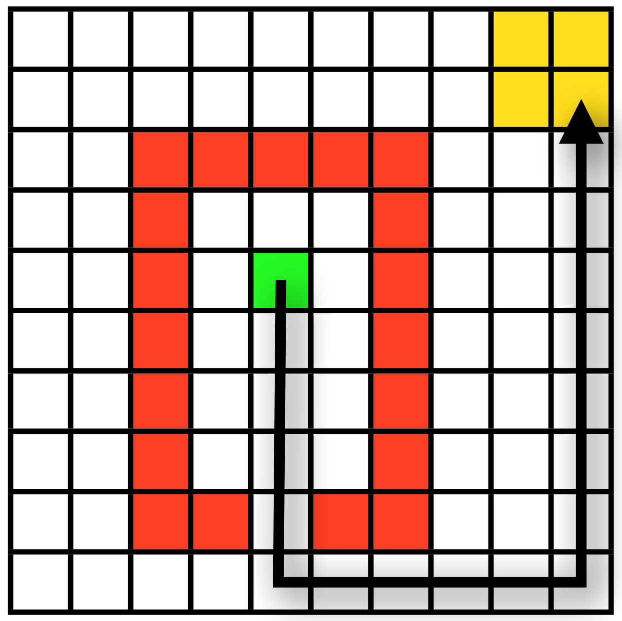
- But: $$\sum_{t=0}^{\infty} r_t = \infty$$

- Episodic $$\sum_{t=0}^{T} r_t < \infty$$

- Discounted
  - infinite horizons $$\sum_{t=0}^{\infty} \gamma^t r_t < \infty$$
  - finite, exponentially distributed horizons

$$\sum_{t=0}^{T} \gamma^t r_t \qquad\qquad T \sim \frac{1}{\tau} e^{t/\tau}$$

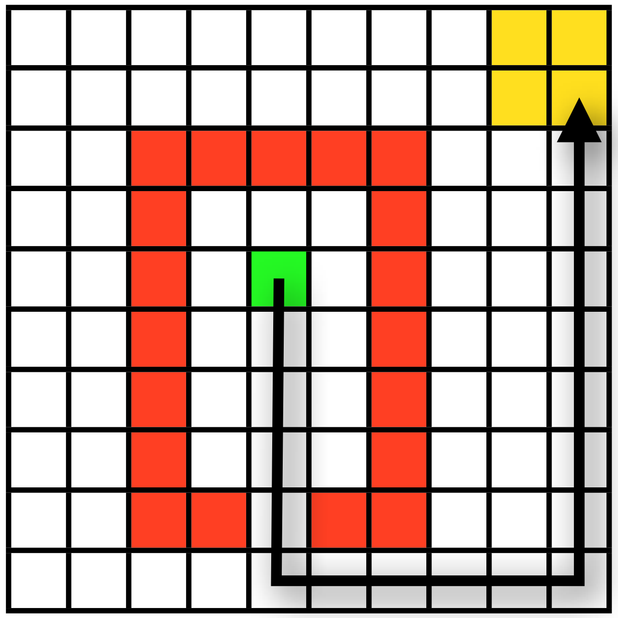
Markov Decision Problems



$$
\begin{aligned}
V^\pi(s_t) &= \mathbb{E}\left[\sum_{t'=1}^{\infty} r_{t'} | s_t = s, \pi\right] \\
&= \mathbb{E}\left[r_1 |\, s_t = s, \pi\right] + \mathbb{E}\left[\sum_{t=2}^{\infty} r_t | s_t = s, \pi\right] \\
&= \mathbb{E}\left[r_1 |\, s_t = s, \pi\right] + \mathbb{E}\left[V^\pi(s_{t+1}) | s_t = s, \pi\right]
\end{aligned}
$$

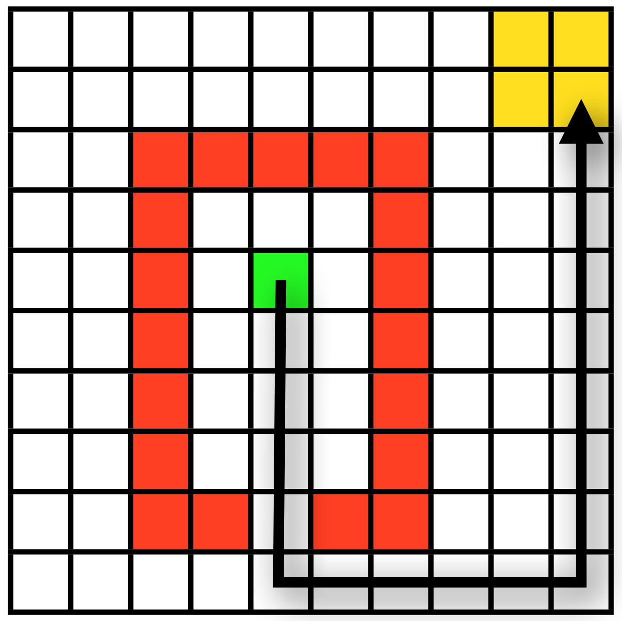
This dynamic consistency is key to many solution approaches.
It states that the value of a state s is related to
the values of its successor states s'.

Markov Decision Problems



$$
\begin{aligned}
V^\pi(s_t) &= \boxed{\mathbb{E}\left[r_1 \middle| s_t = s, \pi\right]} + \mathbb{E}\left[V(s_{t+1}), \pi\right] \\
r_1 &\sim \mathcal{R}(s_2, a_1, s_1) \\
\mathbb{E}\left[r_1 | s_t = s, \pi\right] &= \mathbb{E}\left[\sum_{s_{t+1}} p(s_{t+1}|s_t, a_t)\mathcal{R}(s_{t+1}, a_t, s_t)\right] \\
&= \sum_{a_t} p(a_t|s_t)\left[\sum_{s_{t+1}} p(s_{t+1}|s_t, a_t)\mathcal{R}(s_{t+1}, a_t, s_t)\right] \\
&= \sum_{a_t} \pi(a_t, s_t)\left[\sum_{s_{t+1}} \mathcal{T}^{a_t}_{s_t s_{t+1}} \mathcal{R}(s_{t+1}, a_t, s_t)\right]
\end{aligned}
$$

Bellman equation



$$
\begin{aligned}
V^\pi(s_t) &= \mathbb{E}\left[r_1 \middle| s_t = s, \pi\right] + \mathbb{E}\left[V(s_{t+1}), \pi\right] \\
\mathbb{E}\left[r_1 | s_t, \pi\right] &= \sum_a \pi(a, s_t) \left[\sum_{s_{t+1}} \mathcal{T}^a_{s_t s_{t+1}} \mathcal{R}(s_{t+1}, a, s_t)\right] \\
\mathbb{E}\left[V^\pi(s_{t+1}), \pi, s_t\right] &= \sum_a \pi(a, s_t) \left[\sum_{s_{t+1}} \mathcal{T}^a_{s_t s_{t+1}} V^\pi(s_{t+1})\right]
\end{aligned}
$$

$$
V^\pi(s) = \sum_a \pi(a|s) \left[\sum_{s'} \mathcal{T}^a_{ss'} \left[\mathcal{R}(s', a, s) + V^\pi(s')\right]\right]
$$

Bellman Equation

$$V^{\pi}(s) = \sum_a \pi(a|s) \left[\sum_{s'} \mathcal{T}^a_{ss'} \left[\mathcal{R}(s', a, s) + V^{\pi}(s') \right] \right]$$

Bellman Equation

$$\text{All future reward from state s} \quad = \quad \mathrm{E}\left[\ \text{Immediate reward} \quad + \quad \text{All future reward from next state s'}\ \right]$$

$$V^\pi(s) \quad = \quad \sum_a \pi(a|s)\left[\sum_{s'} \mathcal{T}^a_{ss'}\left[\mathcal{R}(s',a,s) + V^\pi(s')\right]\right]$$

Bellman Equation

$$V^\pi(s) \quad = \quad \sum_a \pi(a|s) \left[\sum_{s'} \mathcal{T}^a_{ss'} \left[\mathcal{R}(s', a, s) + V^\pi(s') \right] \right]$$

$$\begin{array}{c}\text{All future}\\ \text{reward}\\ \text{from state s}\end{array} \quad = \quad \text{E} \left[\begin{array}{c}\text{Immediate}\\ \text{reward}\end{array} \quad + \quad \begin{array}{c}\text{All future}\\ \text{reward}\\ \text{from}\\ \text{next state s'}\end{array} \right]$$

Q values = state-action values

$$V^\pi(s) = \sum_a \pi(a|s) \underbrace{\left[\sum_{s'} \mathcal{T}^a_{ss'} \left[\mathcal{R}(s',a,s) + V^\pi(s')\right]\right]}_{\mathcal{Q}^\pi(s,a)}$$

- so we can define state-action values as:

$$\begin{aligned}\mathcal{Q}(s,a) &= \sum_{s'} \mathcal{T}^a_{ss'} \left[\mathcal{R}(s',a,s) + V(s')\right] \\ &= \mathbb{E}\left[\sum_{t=1}^{\infty} r_t | s, a\right]\end{aligned}$$

- and state values are average state-action values:

$$V(s) = \sum_a \pi(a|s) \mathcal{Q}(s,a)$$

Bellman Equation

$$V^\pi(s) = \sum_a \pi(a|s) \left[\sum_{s'} \mathcal{T}^a_{ss'} \left[\mathcal{R}(s', a, s) + V^\pi(s') \right] \right]$$

- to evaluate a policy, we need to solve the above equation, i.e. find the self-consistent state values

- options for policy evaluation
  - exhaustive tree search - outwards, inwards, depth-first
  - linear solution in 1 step
  - value iteration: iterative updates
  - experience sampling

Solving the Bellman Equation

Option 1: turn it into update equation

Option 2: linear solution (w/ absorbing states)

$$
\begin{aligned}
V(s) &= \sum_a \pi(a, s_t) \left[\sum_{s'} \mathcal{T}^a_{ss'} \left[\mathcal{R}(s', a, s) + V(s') \right] \right] \\
\Rightarrow \mathbf{v} &= \mathbf{R}^\pi + \mathbf{T}^\pi \mathbf{v} \\
\Rightarrow \mathbf{v}^\pi &= (\mathbf{I} - \mathbf{T}^\pi)^{-1} \mathbf{R}^\pi \qquad \mathcal{O}(|\mathcal{S}|^3)
\end{aligned}
$$

Solving the Bellman Equation

Option 1: turn it into update equation

$$V^{k+1}(s) = \sum_a \pi(a, s_t) \left[\sum_{s'} \mathcal{T}^a_{ss'} \left[\mathcal{R}(s', a, s) + V^k(s') \right] \right]$$

Option 2: linear solution (w/ absorbing states)

$$\begin{aligned} V(s) &= \sum_a \pi(a, s_t) \left[\sum_{s'} \mathcal{T}^a_{ss'} \left[\mathcal{R}(s', a, s) + V(s') \right] \right] \\ \Rightarrow \mathbf{v} &= \mathbf{R}^\pi + \mathbf{T}^\pi \mathbf{v} \\ \Rightarrow \mathbf{v}^\pi &= (\mathbf{I} - \mathbf{T}^\pi)^{-1} \mathbf{R}^\pi \qquad \mathcal{O}(|\mathcal{S}|^3) \end{aligned}$$

Policy update

Given the value function for a policy, say via linear solution

$$V^\pi(s) \quad = \quad \sum_a \pi(a|s) \underbrace{\left[\sum_{s'} \mathcal{T}^a_{ss'} \left[\mathcal{R}(s', a, s) + V^\pi(s')\right]\right]}_{\mathcal{Q}^\pi(s,a)}$$

Given the values V for the policy, we can improve the policy by always choosing the best action:

$$\pi'(a|s) = \begin{cases} 1 \text{ if } a = \operatorname{argmax}_a \mathcal{Q}^\pi(s,a) \\ 0 \text{ else} \end{cases}$$

It is guaranteed to improve:

for deterministic policy

$$\mathcal{Q}^\pi(s, \pi'(s)) = \max_a \mathcal{Q}^\pi(s,a) \geq \mathcal{Q}^\pi(s, \pi(s)) = \mathcal{V}^\pi(s)$$

Policy iteration

Policy evaluation

$$\mathbf{v}^{\pi} \quad = \quad (\mathbf{I} - \mathbf{T}^{\pi})^{-1}\mathbf{R}^{\pi}$$

$$\pi(a|s) = \begin{cases} 1 \text{ if } a = \text{argmax}_a \sum_{s'} \mathcal{T}^a_{ss'} \left[\mathcal{R}^a_{ss} + V^{pi}(s')\right] \\ 0 \text{ else} \end{cases}$$

Policy iteration

Policy evaluation

$$\mathbf{v}^{\pi} \quad = \quad (\mathbf{I} - \mathbf{T}^{\pi})^{-1}\mathbf{R}^{\pi}$$

greedy policy improvement

$$\pi(a|s) = \begin{cases} 1 \,\text{if}\, a = \text{argmax}_a \sum_{s'} \mathcal{T}^a_{ss'} \left[\mathcal{R}^a_{ss} + V^{pi}(s')\right] \\ 0 \,\text{else} \end{cases}$$

Policy iteration

Policy evaluation

$$\mathbf{v}^{\pi} \quad = \quad (\mathbf{I} - \mathbf{T}^{\pi})^{-1}\mathbf{R}^{\pi}$$

Value iteration

$$V^*(s) = \max_a \sum_{s'} \mathcal{T}^a_{ss'}\left[\mathcal{R}^a_{ss} + V^*(s')\right]$$

greedy policy improvement

$$\pi(a|s) = \begin{cases} 1 \,\text{if}\, a = \text{argmax}_a \sum_{s'} \mathcal{T}^a_{ss'}\left[\mathcal{R}^a_{ss} + V^{pi}(s')\right] \\ 0 \,\text{else} \end{cases}$$

Model-free solutions

- So far we have assumed knowledge of R and T
  - R and T are the 'model' of the world, so we assume full knowledge of the dynamics and rewards in the environment
- What if we don't know them?
- We can still learn from state-action-reward samples
  - we can learn R and T from them, and use our estimates to solve as above
  - alternatively, we can directly estimate V or Q

Solving the Bellman Equation

Option 3: sampling

$$V(s) \quad = \quad \sum_a \pi(a, s_t)\left[\sum_{s'} \mathcal{T}^a_{ss'}\left[\mathcal{R}(s', a, s) + V(s')\right]\right]$$

So we can just draw some samples from the policy and the transitions and average over them:

$$a \quad = \quad \sum_k f(x_k)p(x_k)$$

$$x^{(i)} \sim p(x) \rightarrow \hat{a} = \frac{1}{N}\sum_i f(x^{(i)})$$

Solving the Bellman Equation

Option 3: sampling

So we can just draw some samples from the policy and the transitions and average over them:

$$
\begin{aligned}
a &= \sum_k f(x_k) p(x_k) \\
x^{(i)} \sim p(x) &\rightarrow \hat{a} = \frac{1}{N} \sum_i f(x^{(i)})
\end{aligned}
$$

Solving the Bellman Equation

Option 3: sampling

this is an expectation over policy and transition samples.

So we can just draw some samples from the policy and the transitions and average over them:

$$a = \sum_k f(x_k)p(x_k)$$

$$x^{(i)} \sim p(x) \rightarrow \hat{a} = \frac{1}{N}\sum_i f(x^{(i)})$$

Solving the Bellman Equation

Option 3: sampling

this is an expectation over policy and transition samples.

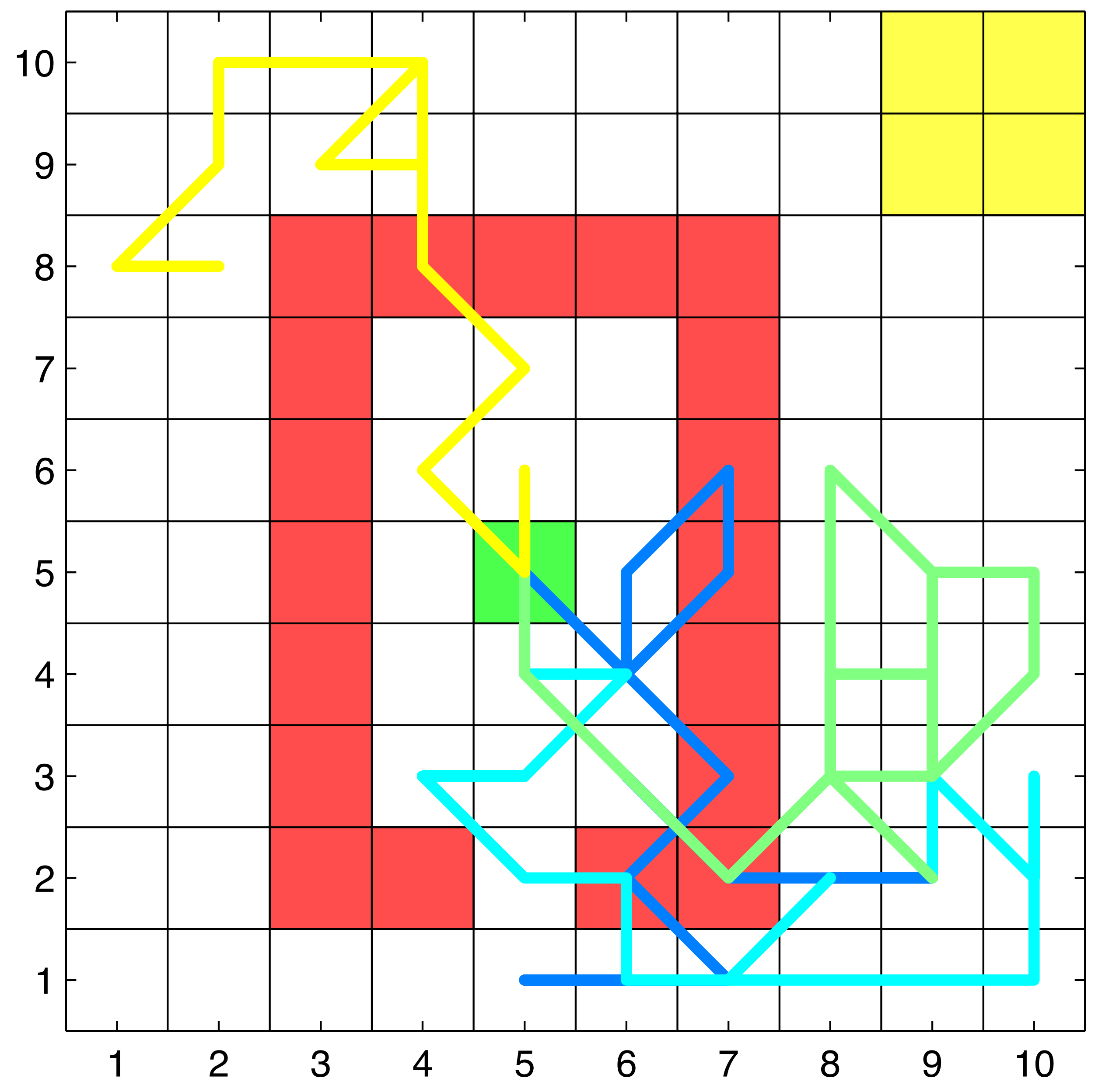
So we can just draw some samples from the policy and the transitions and average over them:

$$a = \sum_k f(x_k)p(x_k)$$

$$x^{(i)} \sim p(x) \rightarrow \hat{a} = \frac{1}{N}\sum_i f(x^{(i)})$$

more about this later...

Learning from samples



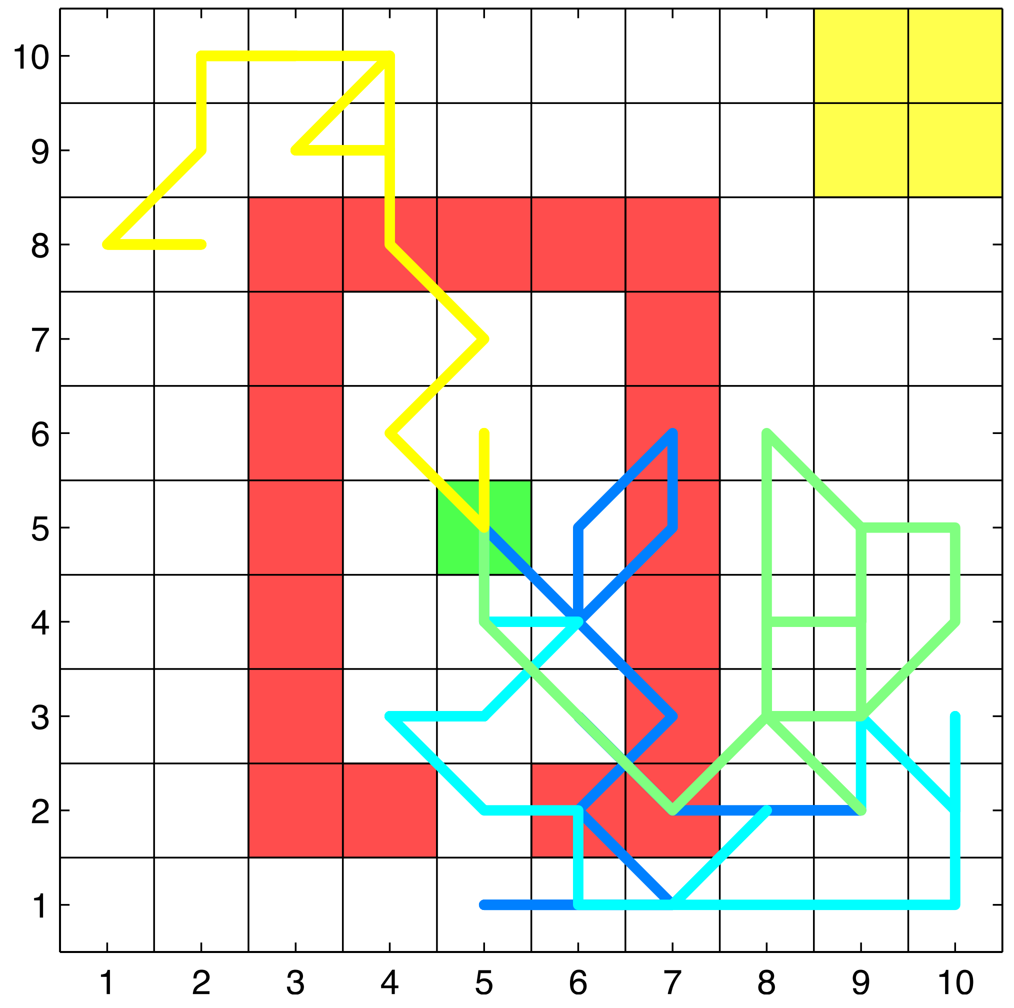
A new problem: exploration versus exploitation

Monte Carlo

- First visit MC
  - randomly start in all states, generate paths, average for starting state only

$$\mathcal{V}(s) = \frac{1}{N}\sum_i \left\{ \sum_{t'=1}^{T} r^i_{t'} | s_0 = s \right\}$$

- More efficient use of samples
  - Every visit MC
  - Bootstrap: TD
  - Dyna
- Better samples
  - on policy versus off policy
  - Stochastic search, UCT...



Update equation: towards TD

Bellman equation

$$V(s) \quad = \quad \sum_a \pi(a,s)\left[\sum_{s'} \mathcal{T}^a_{ss'}\left[\mathcal{R}(s',a,s)+V(s')\right]\right]$$

Not yet converged, so it doesn't hold:

$$dV(s) = -V(s) + \sum_a \pi(a,s)\left[\sum_{s'} \mathcal{T}^a_{ss'}\left[\mathcal{R}(s',a,s)+V(s')\right]\right]$$

And then use this to update

$$V^{i+1}(s) = V^i(s) + dV(s)$$

TD learning

$$dV(s) = -V(s) + \sum_a \pi(a,s) \left[\sum_{s'} \mathcal{T}^a_{ss'} \left[\mathcal{R}(s',a,s) + V(s') \right] \right]$$

TD learning

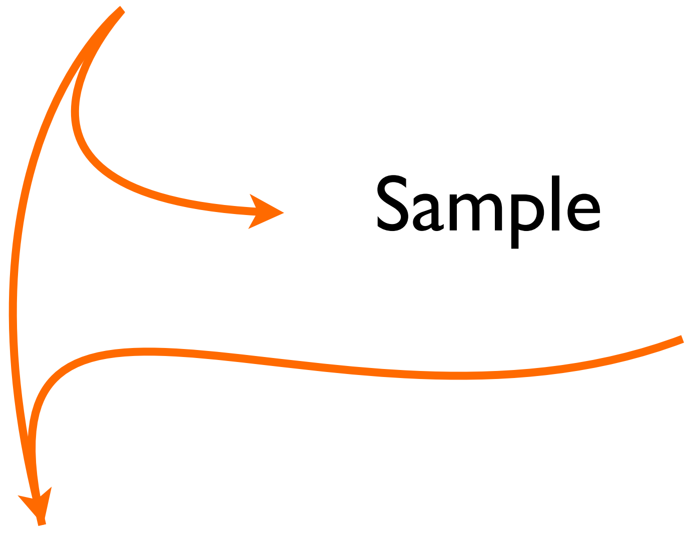
$$dV(s) = -V(s) + \sum_a \pi(a,s) \left[\sum_{s'} \mathcal{T}^a_{ss'} \left[\mathcal{R}(s',a,s) + V(s') \right] \right]$$

$\xrightarrow{}$ Sample

$$\begin{aligned} a_t &\sim \pi(a|s_t) \\ s_{t+1} &\sim \mathcal{T}^{a_t}_{s_t,s_{t+1}} \\ r_t &= \mathcal{R}(s_{t+1}, a_t, s_t) \end{aligned}$$

TD learning

$$dV(s) = -V(s) + \sum_a \pi(a,s) \left[\sum_{s'} \mathcal{T}^a_{ss'} \left[\mathcal{R}(s',a,s) + V(s') \right] \right]$$



$$\begin{aligned} a_t &\sim \pi(a|s_t) \\ s_{t+1} &\sim \mathcal{T}^{a_t}_{s_t,s_{t+1}} \\ r_t &= \mathcal{R}(s_{t+1},a_t,s_t) \end{aligned}$$

$$\delta_t = -V_{t-1}(s_t) + r_t + V_{t-1}(s_{t+1})$$

TD learning

$$dV(s) = -V(s) + \sum_a \pi(a,s) \left[\sum_{s'} \mathcal{T}^a_{ss'} \left[\mathcal{R}(s', a, s) + V(s') \right] \right]$$

Sample

$$\begin{aligned} a_t &\sim \pi(a|s_t) \\ s_{t+1} &\sim \mathcal{T}^{a_t}_{s_t, s_{t+1}} \\ r_t &= \mathcal{R}(s_{t+1}, a_t, s_t) \end{aligned}$$

$$\delta_t = -V_{t-1}(s_t) + r_t + V_{t-1}(s_{t+1})$$

$$V^{i+1}(s) = V^i(s) + dV(s) \longrightarrow V_t(s_t) = V_{t-1}(s_t) + \alpha \delta_t$$

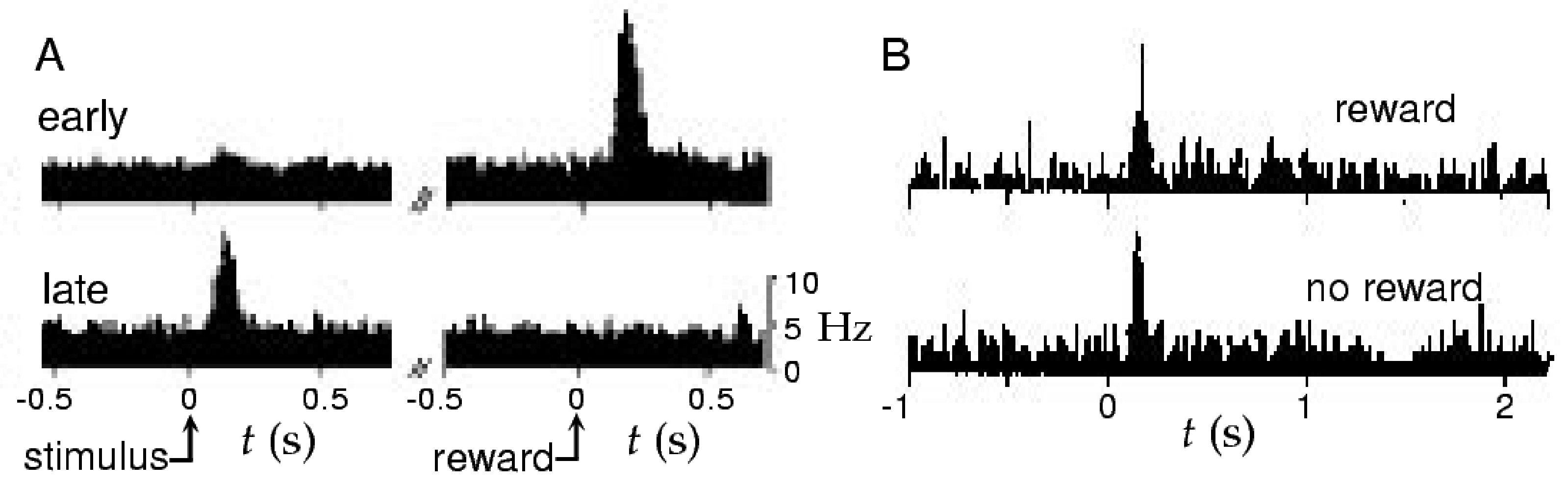
TD learning

$$\begin{aligned} a_t &\sim \pi(a|s_t) \\ s_{t+1} &\sim \mathcal{T}^{a_t}_{s_t,s_{t+1}} \\ r_t &= \mathcal{R}(s_{t+1}, a_t, s_t) \end{aligned}$$

$$\delta_t = -V_t(s_t) + r_t + V_t(s_{t+1})$$

$$V_{t+1}(s_t) = V_t(s_t) + \alpha\delta_t$$

Aside: what makes a TD error?



- unpredicted reward expectation change
- disappears with learning
- stays with probabilistic reinforcement
- sequentiality
  - TD error vs prediction error
- see Niv and Schoenbaum 2008

Schultz et al.

The effect of bootstrapping

B1
B1
B1
B1
B1
B1
B0
A0   B0

Markov (every visit)
V(B)=3/4
V(A)=0

TD
V(B)=3/4
V(A)=~3/4

- Average over various bootstrappings: TD(λ)

after Sutton and Barto 1998

Actor-critic

- policy and value separately parameterised

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

$$w(s,a) \leftarrow w(s,a) + \beta\delta_t$$
$$w(s,a) \leftarrow w(s,a) + \beta\delta_t(1-\pi(s,a))$$

$$\pi(a|s) = \frac{e^{w(s,a)}}{\sum_{a'} e^{w(s,a')}}$$

SARSA

- Do TD for state-action values instead:

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

$$s_t, a_t, r_t, s_{t+1}, a_{t+1}$$

- convergence guarantees - will estimate $\mathcal{Q}^\pi(s, a)$

Q learning: off-policy

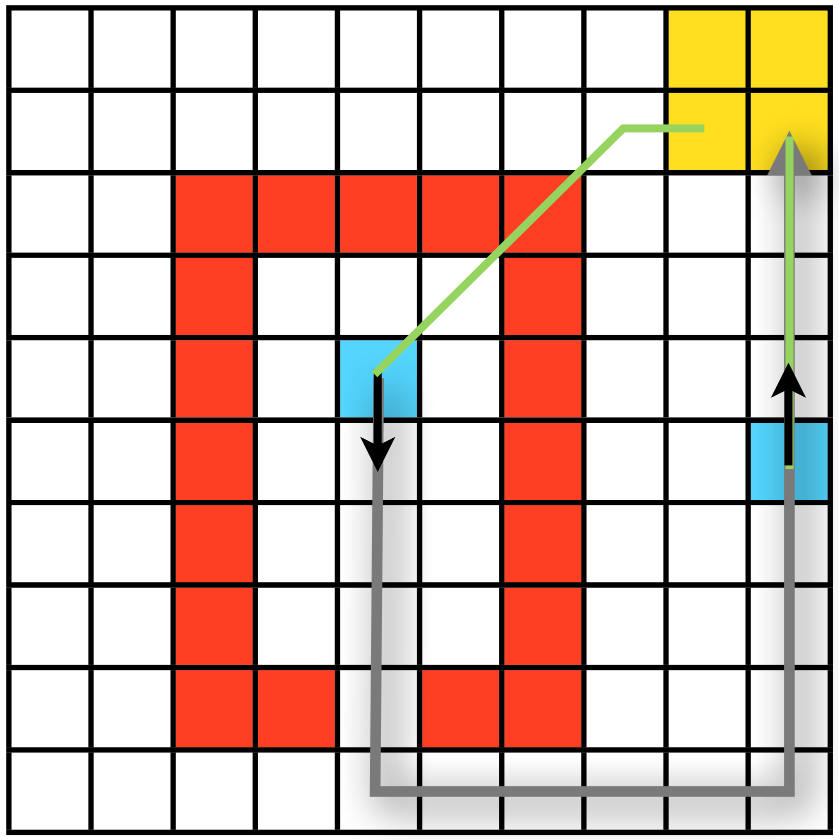
- Learn off-policy
  - draw from some policy
  - “only” require extensive sampling

$$\mathcal{Q}(s_t, a_t) \leftarrow \mathcal{Q}(s_t, a_t) + \alpha \left[\underbrace{r_t + \gamma \max_a \mathcal{Q}(s_{t+1}, a)}_{\text{update towards optimum}} - \mathcal{Q}(s_t, a_t) \right]$$

- will estimate $\mathcal{Q}^*(s, a)$

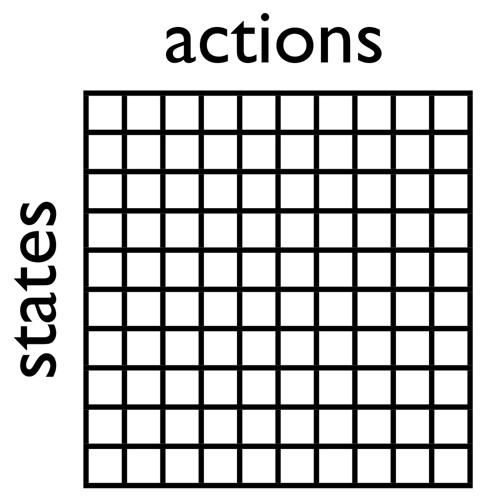
Learning in the wrong state space

- states=distance from goal
- state-space choice crucial
  - too big -> curse of dimensionality
  - too small -> can't express good policies
  - unsolved problem
- humans in tasks have to infer state-space

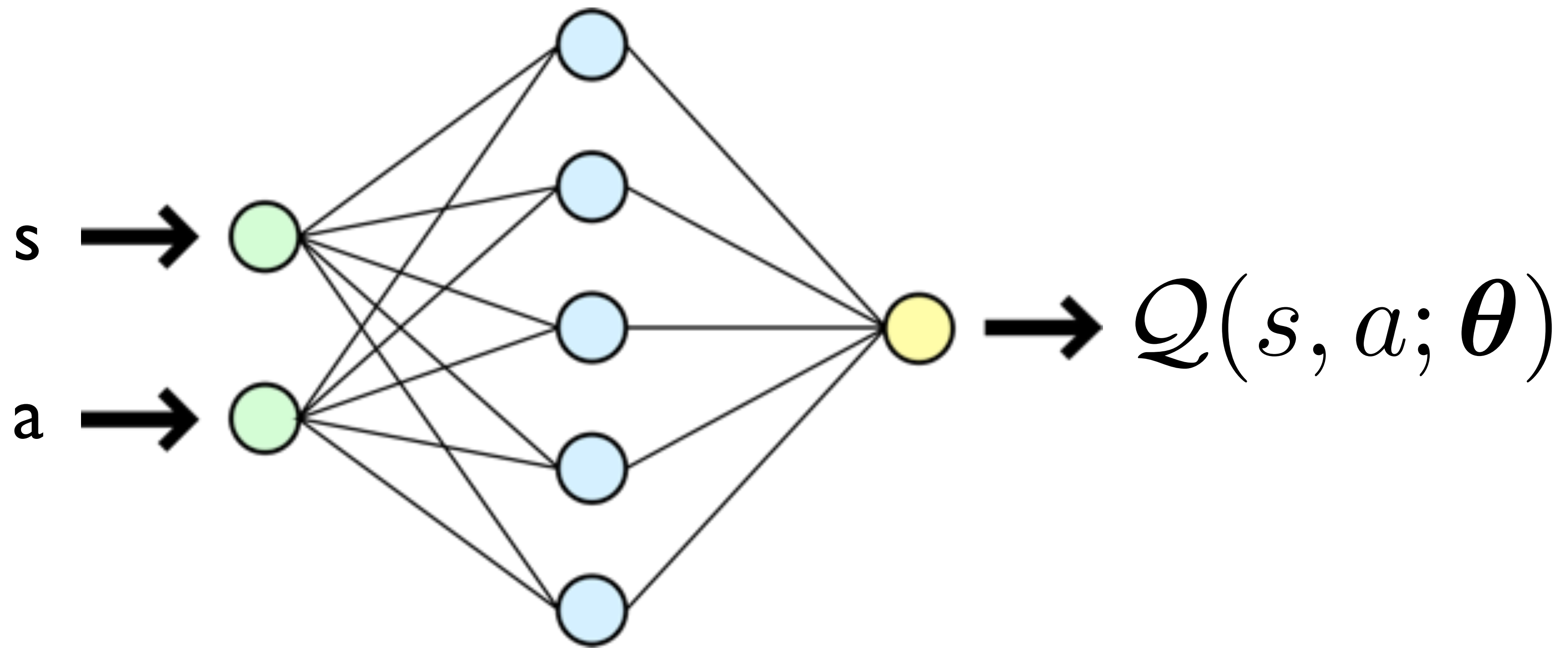


Neural network approximations

- So far: look-up tables



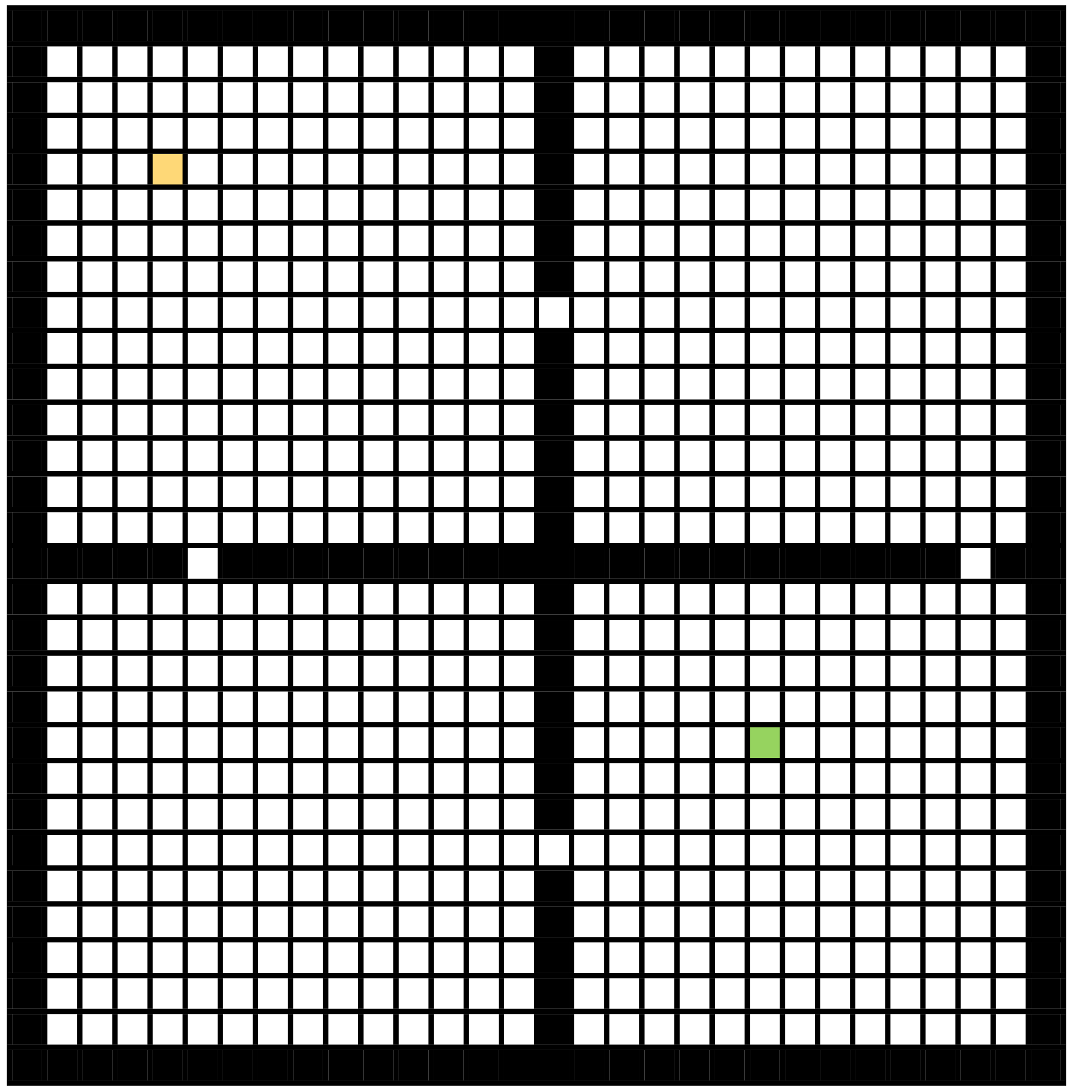
- Parametric value functions



- Humans and animals generalize

Hierarchical decompositions

- Subtasks stay the same
  - Learn subtasks
  - Learn how to use subtasks
- Macroactions
  - ‘go to door’
  - search goal
- Humans establish subgoals on-line
  - how is not yet known



Learning a model

- So far we've concentrated on model-free learning
- What if we want to build some model of the environment?

$$V(s) \quad = \quad \sum_a \pi(a,s) \left[\sum_{s'} \boxed{\mathcal{T}^a_{ss'}} \left[\boxed{\mathcal{R}(s',a,s)} + V(s') \right] \right]$$

- Count transitions

$$\hat{\mathcal{T}}^a_{ss'} = \frac{\sum_t \mathbf{1}(s_t = s, a_t = a, s_{t+1} = s')}{\sum_t \mathbf{1}(s_t = s, a_t = a)}$$

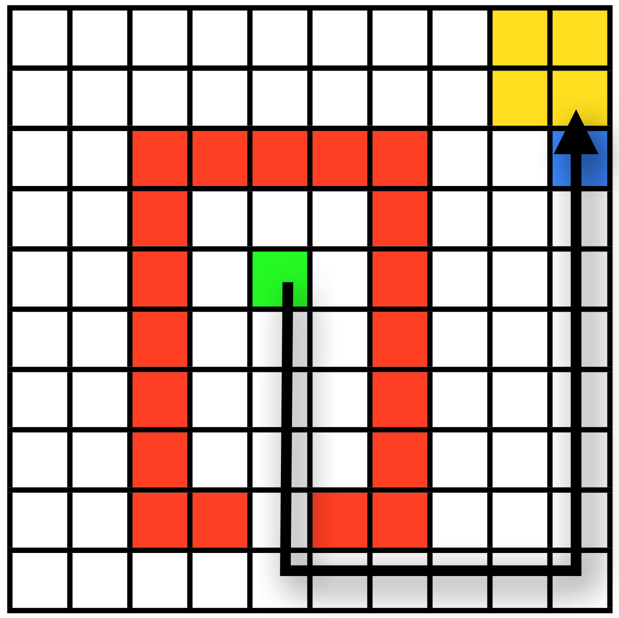
- Average rewards

$$\hat{\mathcal{R}}^a_{ss'} = \frac{\sum_t r_t \mathbf{1}(s_t = s, a_t = a, s_{t+1} = s')}{\sum_t \mathbf{1}(s_t = s, a_t = a, s_{t+1} = s')}$$

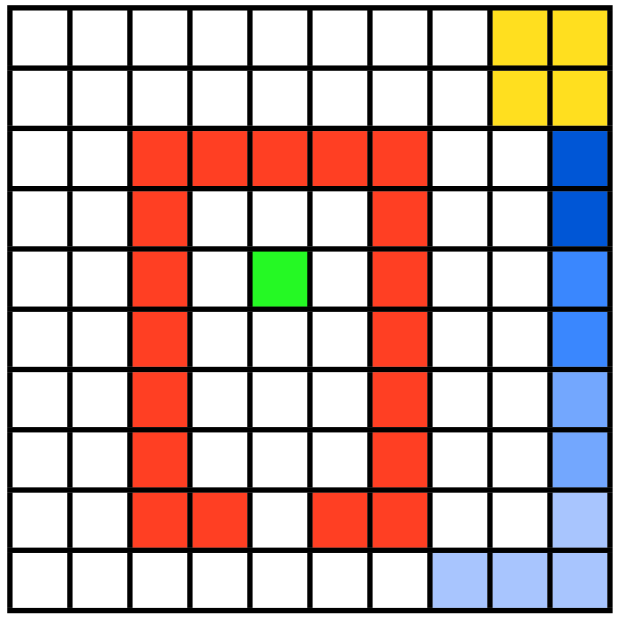
Dyna

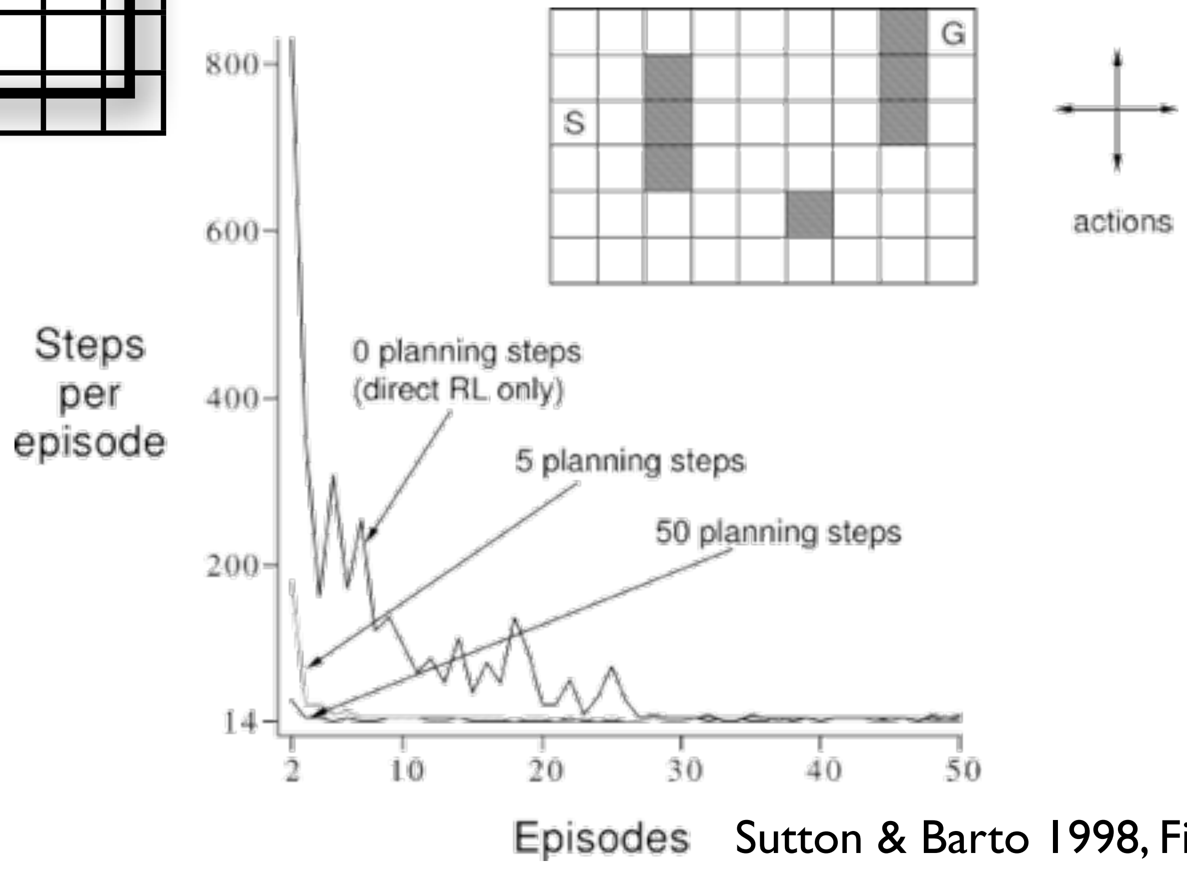
- Combine model estimation with TD learning

$$V_{t+1}(s_t) = V_t(s_t) + \alpha\delta_t$$



Generate extra experience samples from estimated model





Sutton & Barto 1998, Figure 9.5

Conclusion I

- Long-term rewards have internal consistency
- This can be exploited for solution
- Exploration and exploitation trade off when sampling
- Clever use of samples can produce fast learning
  - Brain most likely does something like this