Reinforcement learning

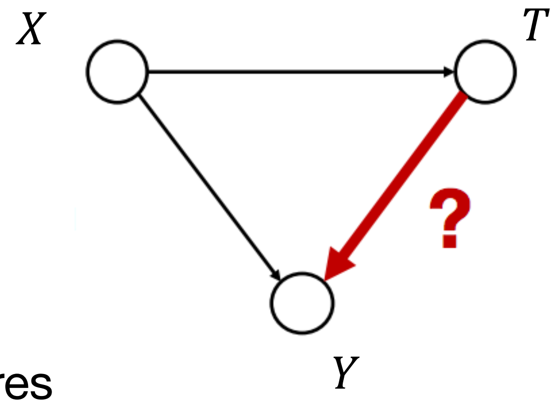
Fredrik D. Johansson

Clinical ML @ MIT

6.S897/HST.956: Machine Learning for Healthcare, 2019

Reminder: Causal effects

- Potential outcomes under treatment and control, $Y(1), Y(0)$
- Covariates and treatment, X, T



- Conditional average treatment effect (CATE)

$$CATE(X) = \mathbb{E}[Y(1) - Y(0) \mid X]$$

Potential outcomes   Features

Today: Treatment policies/regimes

- A **policy** π assigns treatments to patients (typically depending on their medical history/state)

- **Example:** For a patient with medical history x,

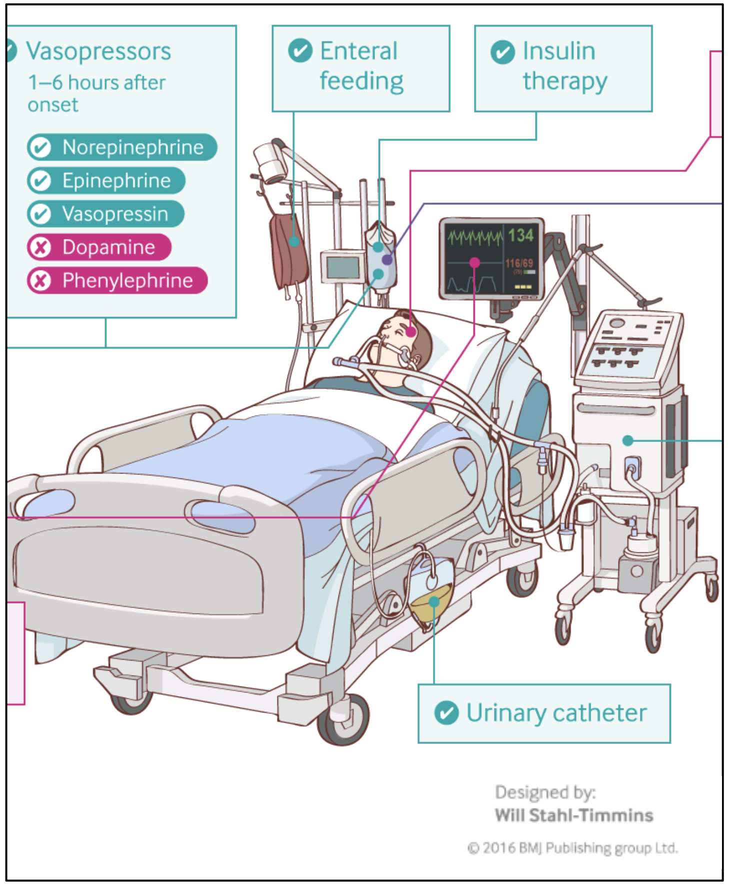
$$\pi(x) = \mathbb{I}[CATE(x) > 0]$$

"Treat if effect is positive"

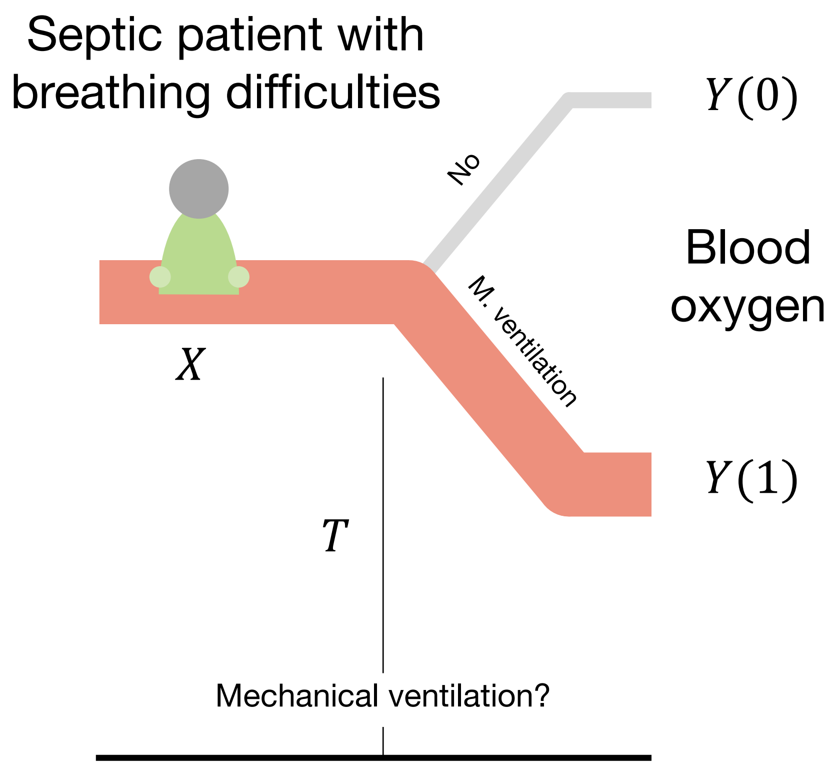
- Today we focus on policies guided by clinical outcomes (as opposed to legislation, monetary cost or side-effects)

Example: Sepsis management

- **Sepsis** is a complication of an infection which can lead to massive organ failure and death
- One of the leading causes of death in the **ICU**
- The primary treatment target is the **infection**
- Other symptoms need **management:** breathing difficulties, low blood pressure, …



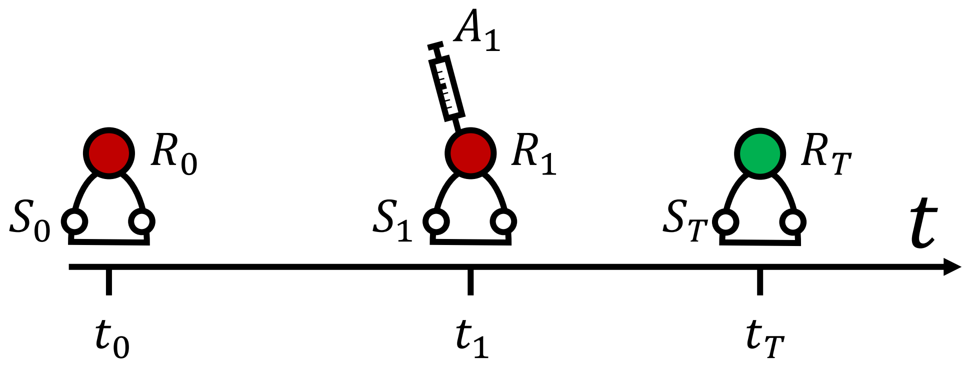
Recall: Potential outcomes



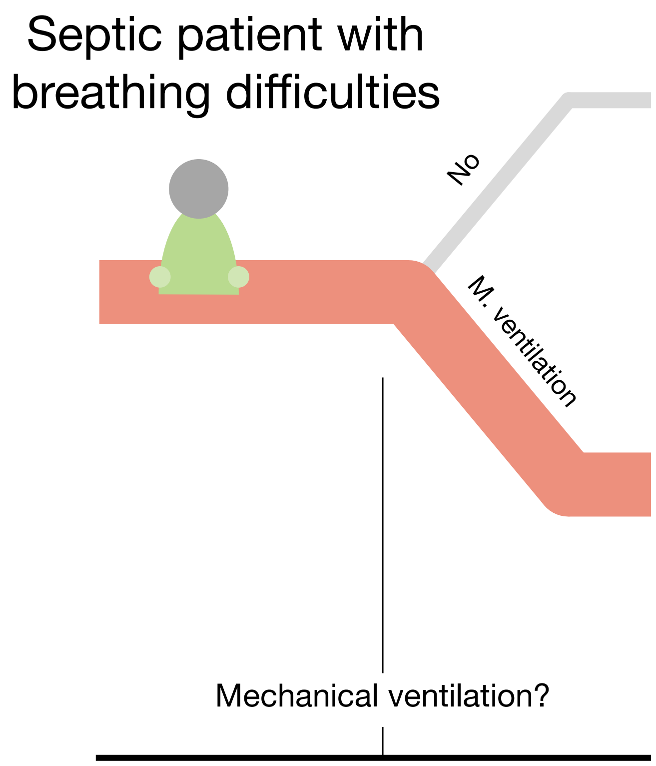
1. Should the patient be put on mechanical ventilation?

Today: Sequential decision making

- Many clinical decisions are made in **sequence**
- Choices early **may rule out** actions later
- Can we optimize the **policy** by which actions are made?

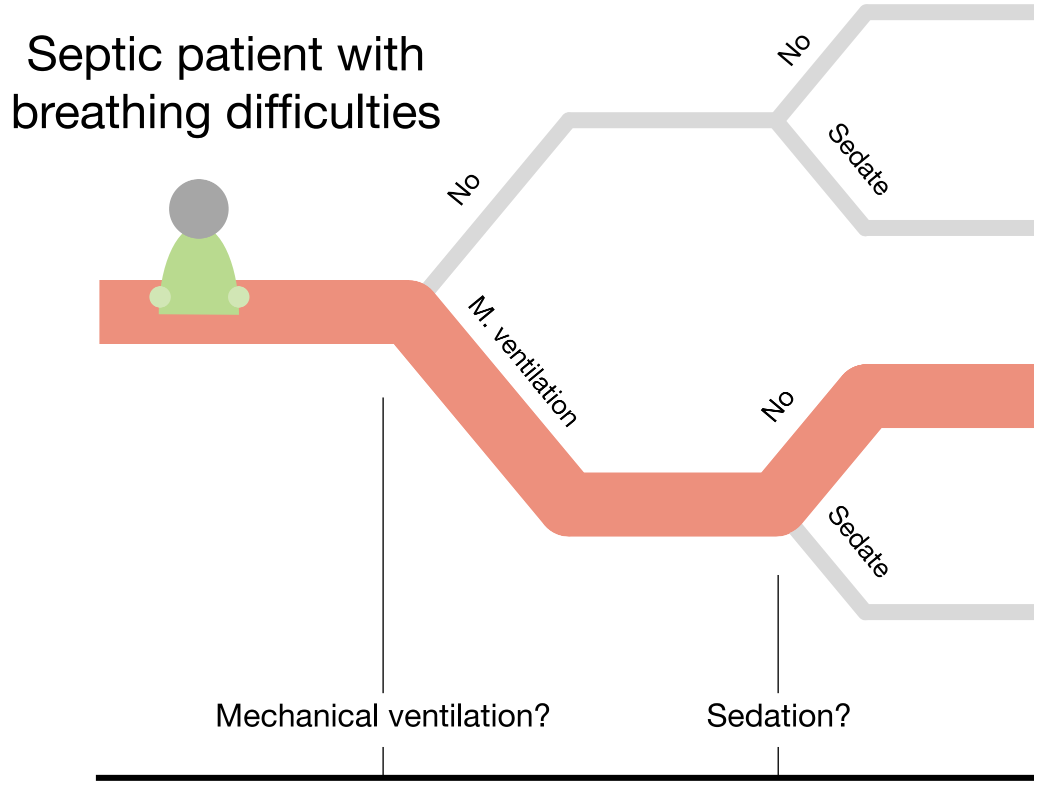


Recall: Potential outcomes



1. Should the patient be put on mechanical ventilation?

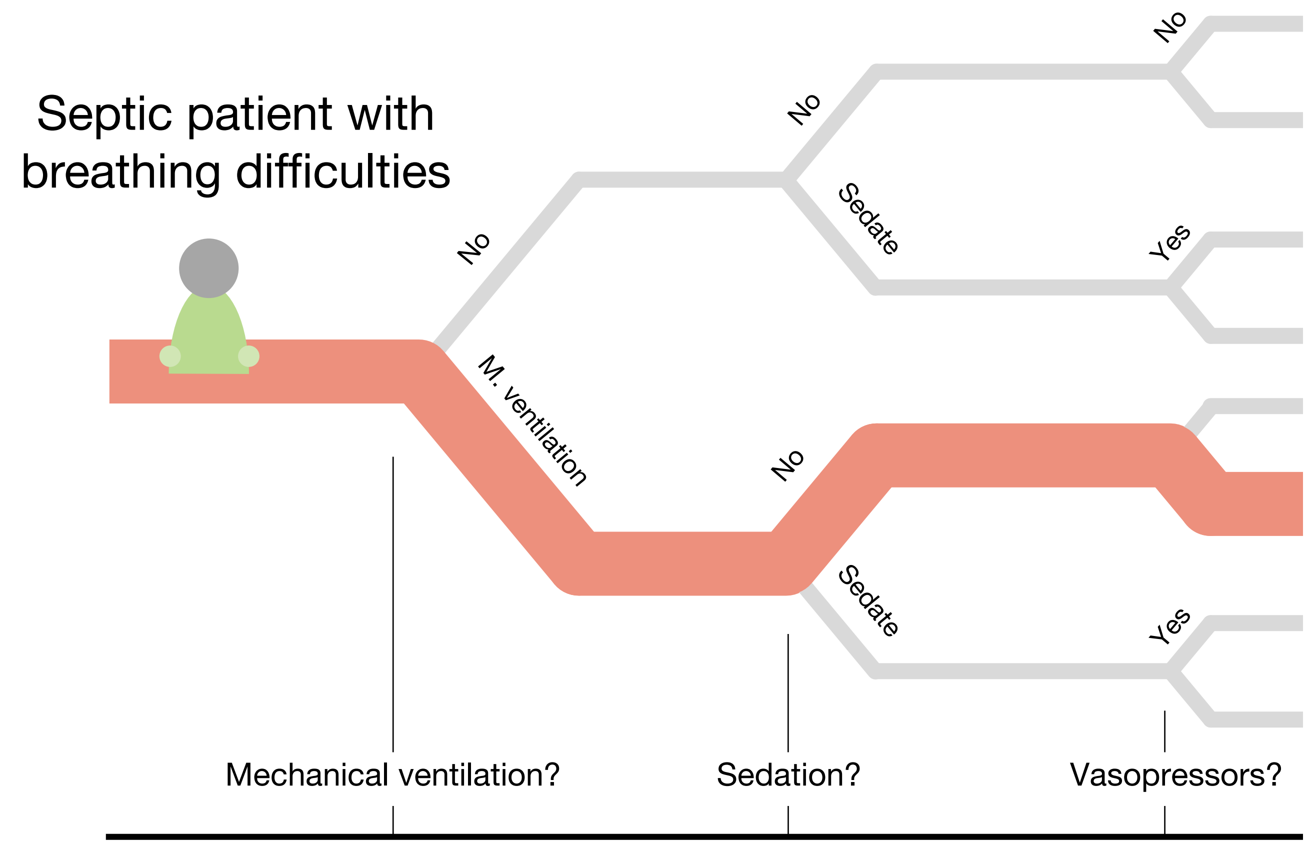
Example: Sepsis management



2. Should the patient be sedated?

(To alleviate discomfort due to mech. ventilation)

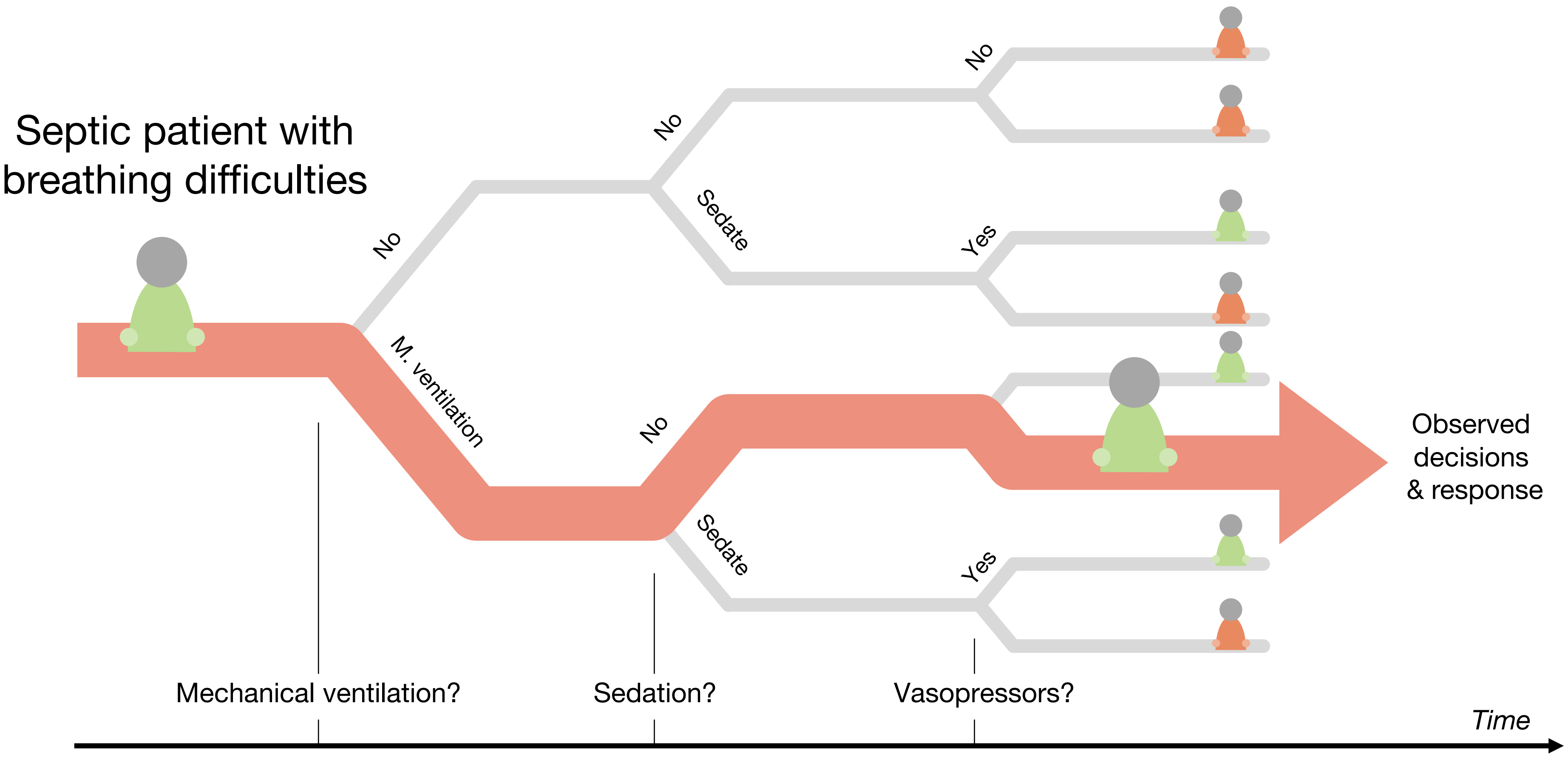
Example: Sepsis management



3. Should we artificially raise blood pressure?

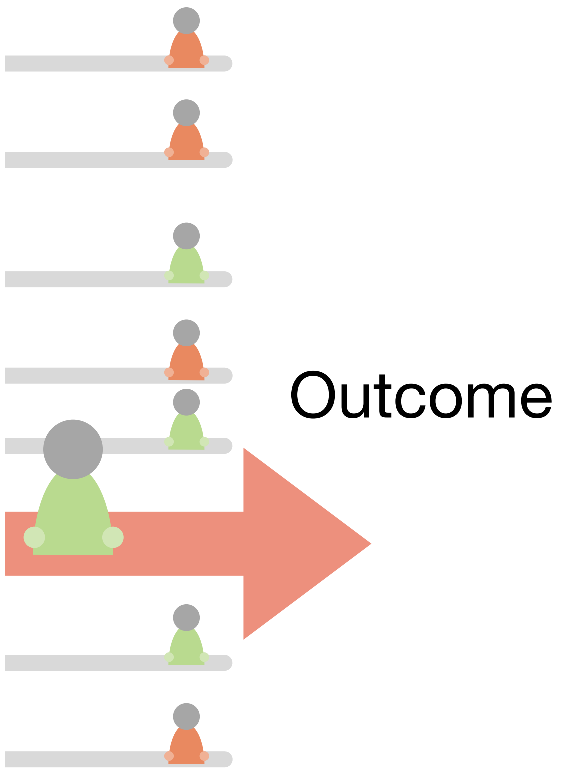
(Which may have dropped due to sedation)

Example: Sepsis management



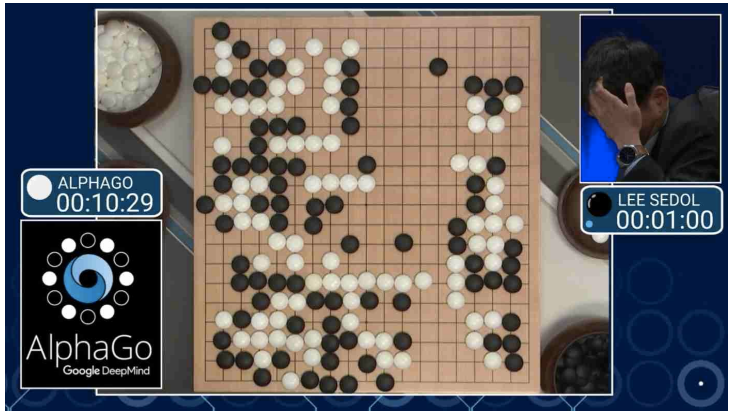
Finding optimal policies

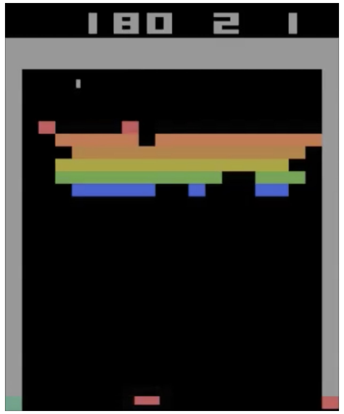
- How can we treat patients so that their outcomes are **as good as possible**?
- What are **good outcomes**?
- Which **policies** should we consider?



Success stories in popular press

- AlphaStar
- AlphaGo
- DQN Atari
- Open AI Five





Reinforcement learning

- Maximize reward!

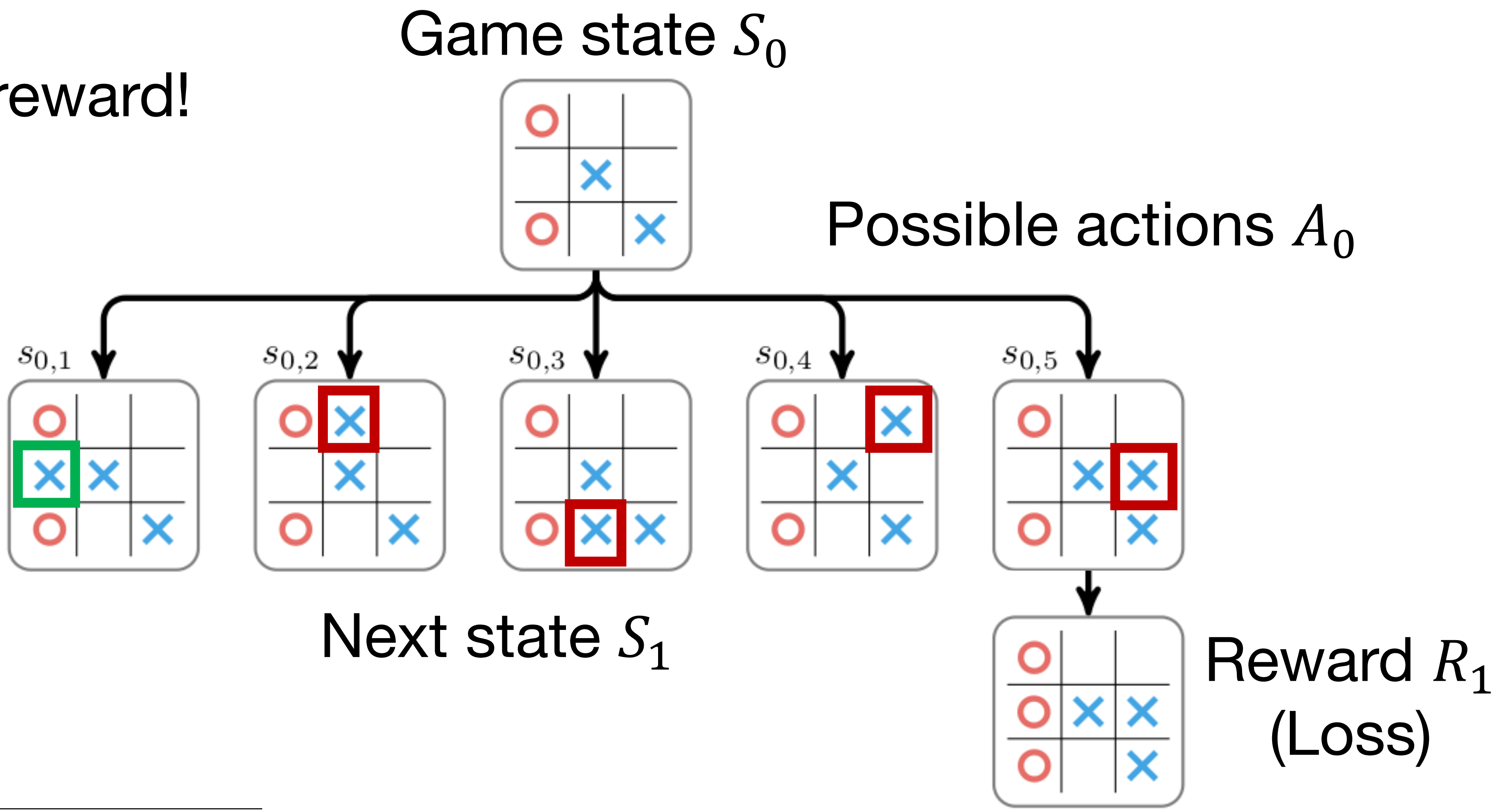
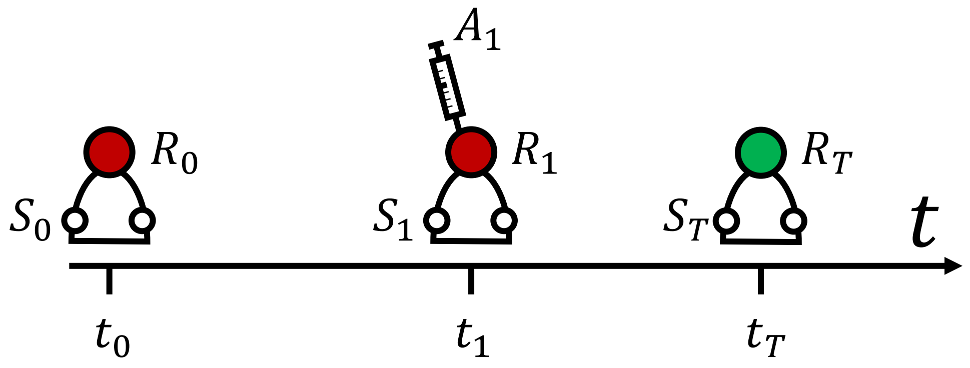


Figure by Tim Wheeler, tim.hibal.org

Great! Now let's treat patients

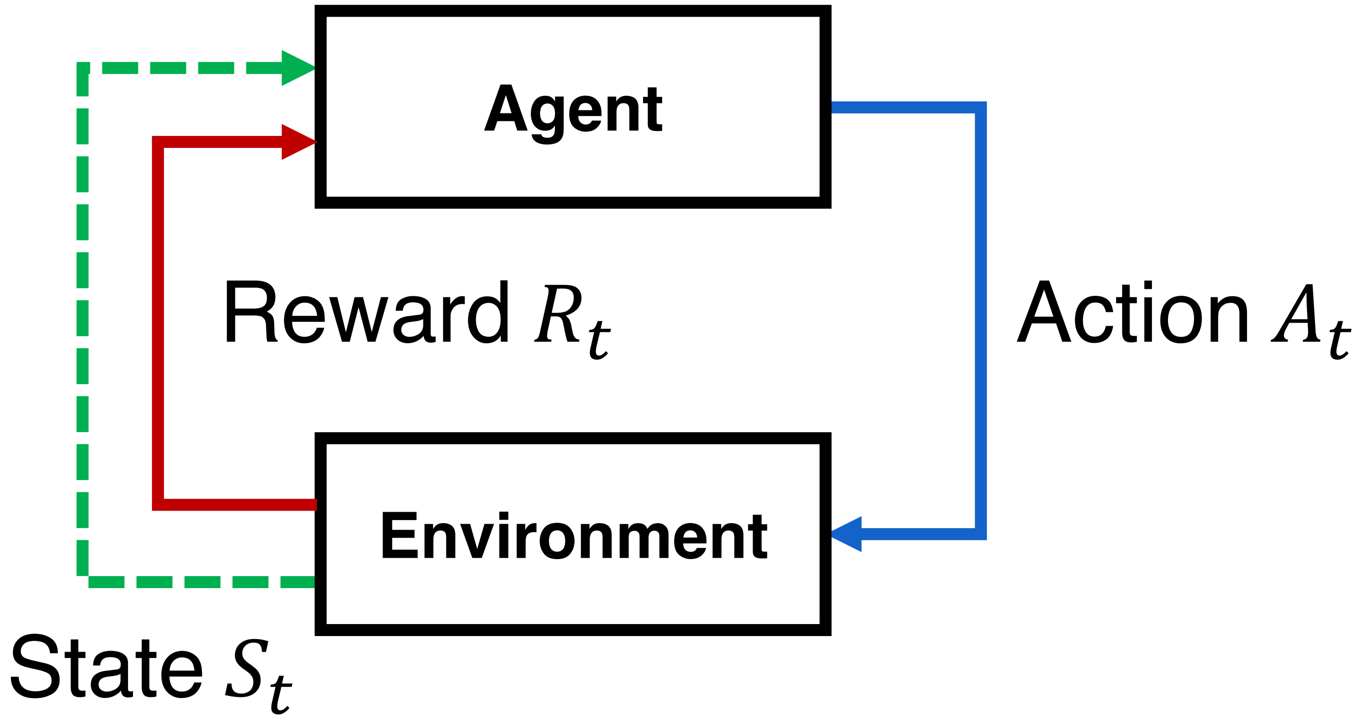
- Patient **state** at time S_t is like the game board
- Medical **treatments** A_t are like the actions
- **Outcomes** R_t are the rewards in the game
- What could **possibly** go wrong?



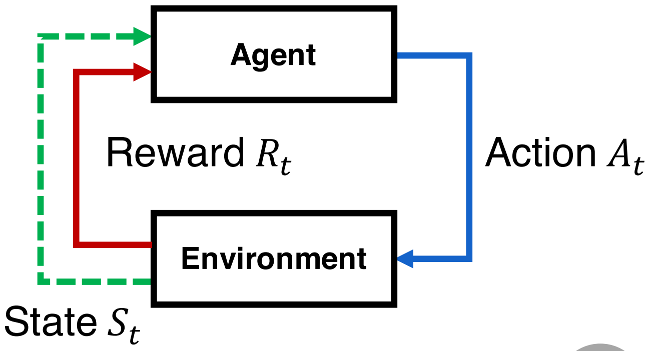
1. **Decision processes**
2. Reinforcement learning
3. Learning from batch (off-policy) data
4. Reinforcement learning in healthcare

Decision processes

- An **agent** repeatedly, at times t takes **actions** A_t to receive **rewards** R_t from an **environment**, the **state** S_t of which is (partially) observed



Decision process: Mechanical ventilation



$$R_t = R_t^{vitals} + R_t^{vent\ off} + R_t^{vent\ on}$$

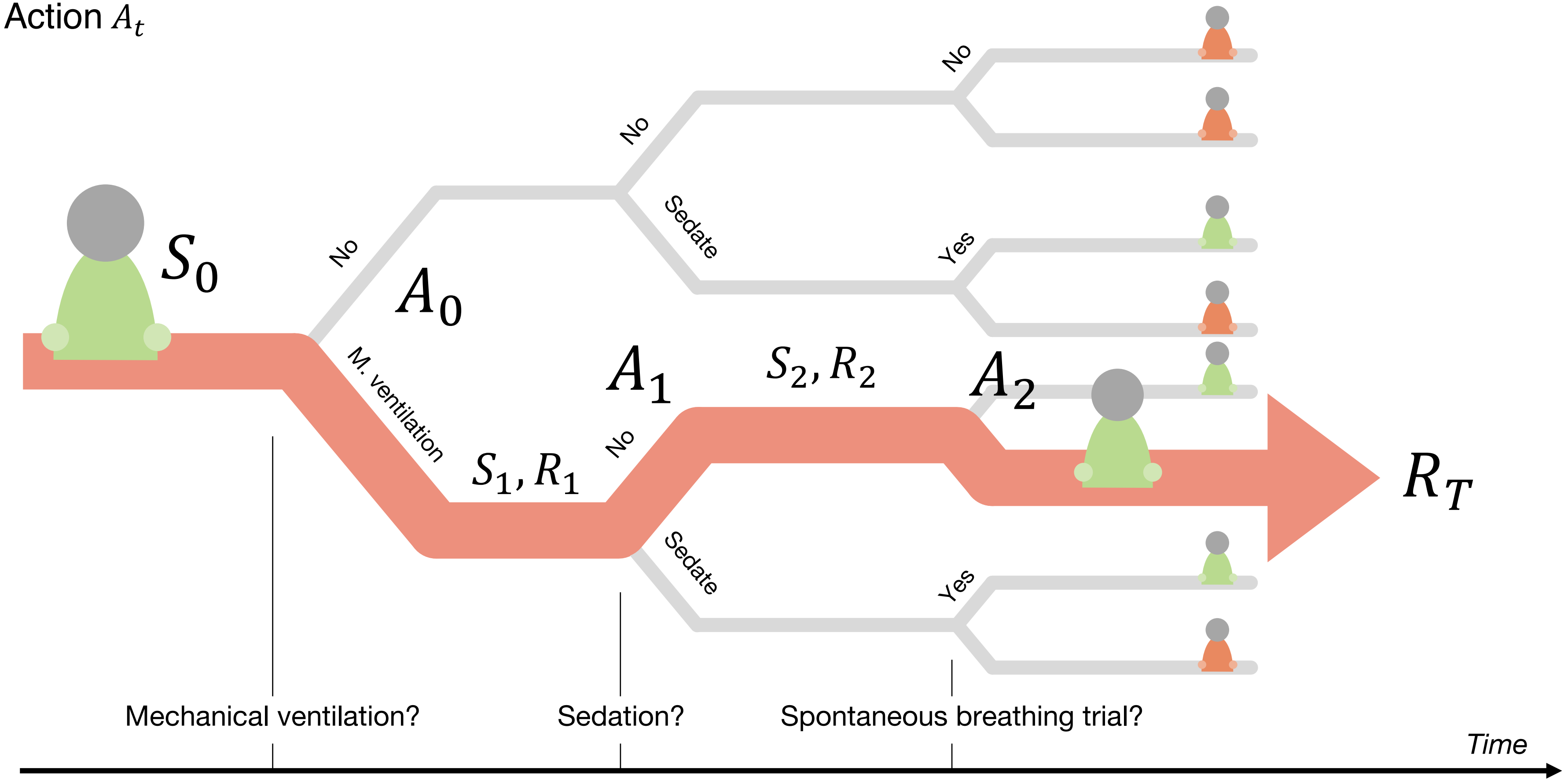

arXiv:1704.06300v1 [cs.AI] 20 Apr 2017

A Reinforcement Learning Approach to Weaning of Mechanical Ventilation in Intensive Care Units

Niranjani Prasad, Princeton University; Li-Fang Cheng, Princeton University; Corey Chivers, Penn Medicine; Michael Draugelis, Penn Medicine; Barbara E. Engelhardt, Princeton University

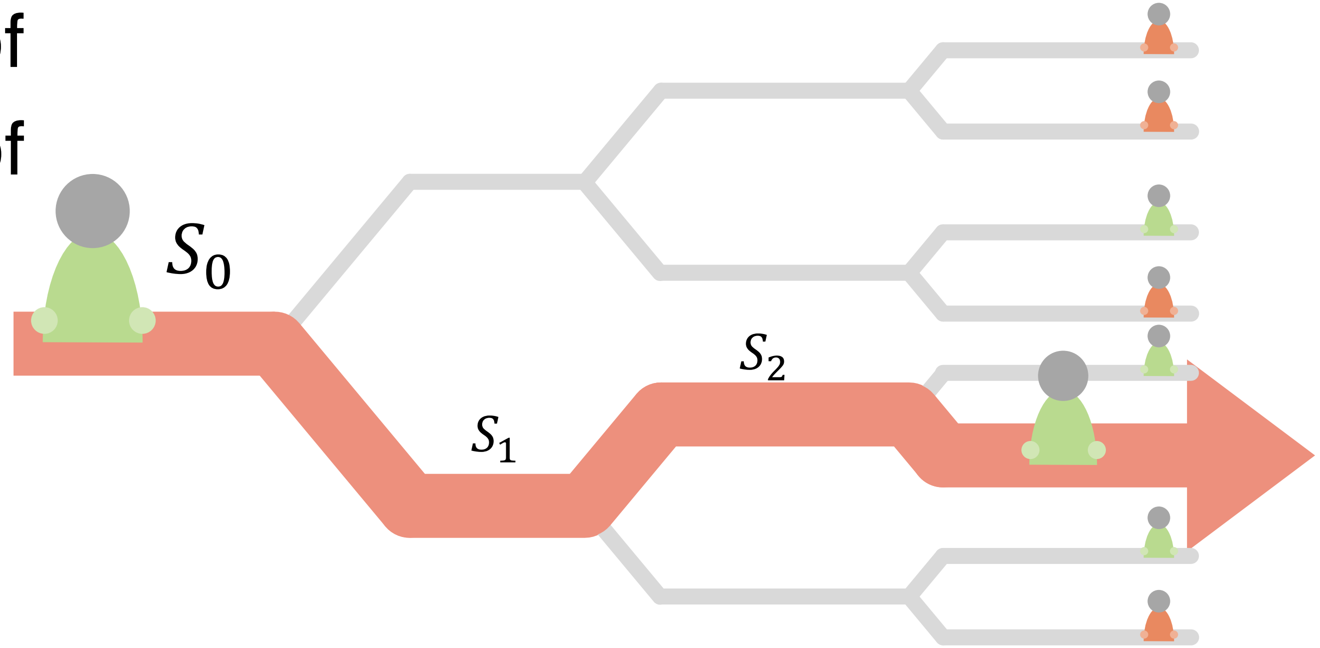
Abstract

1 Introduction



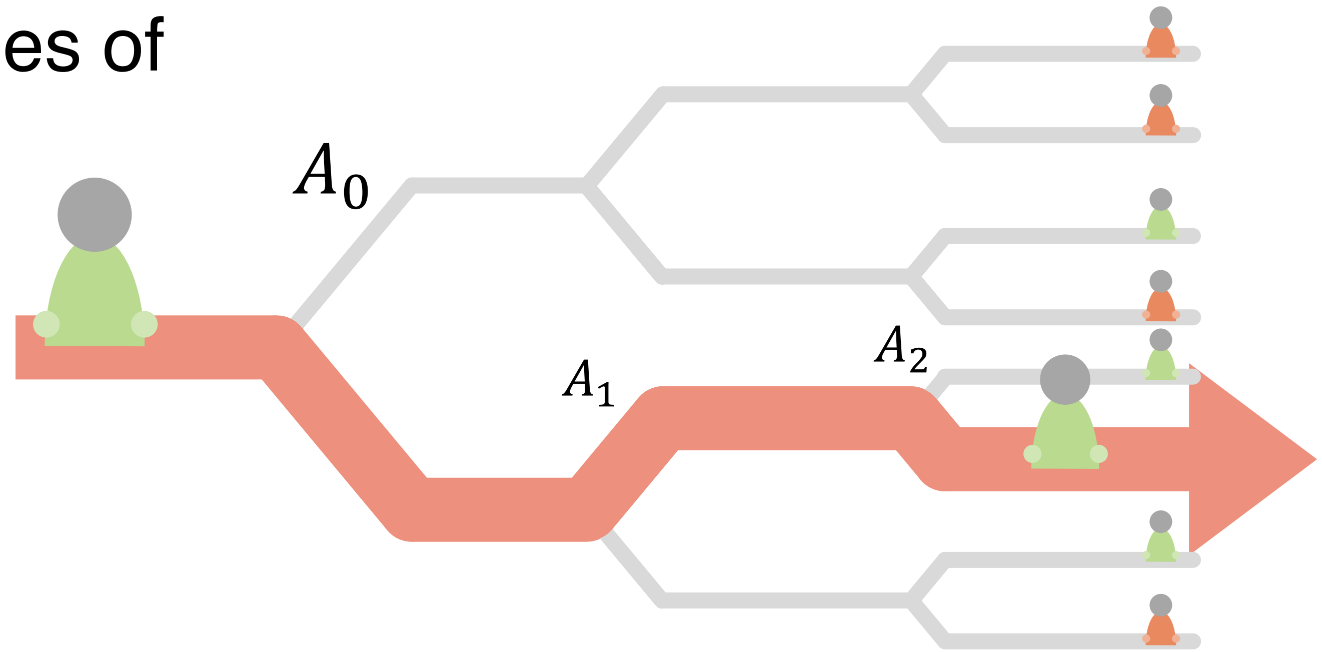
Decision process: Mechanical ventilation

- **State** S_t includes demographics, physiological measurements, ventilator settings, level of consciousness, dosage of sedatives, time to ventilation, number of intubations



Decision process: Mechanical ventilation

- **Actions** A_t include intubation and extubation, as well as administration and dosages of sedatives



Decision processes

- A decision process specifies how states S_t, actions A_t, and rewards R_t are **distributed**: $p(S_0, \ldots, S_T, A_0, \ldots, A_T, R_0, \ldots, R_T)$

- The agent interacts with the environment according to a **behavior policy** $\mu = p(A_t \mid \cdots)^*$

---

* The … depends on the type of agent

Markov Decision Processes

- Markov decision processes (MDPs) are a special case

- Markov **transitions**:

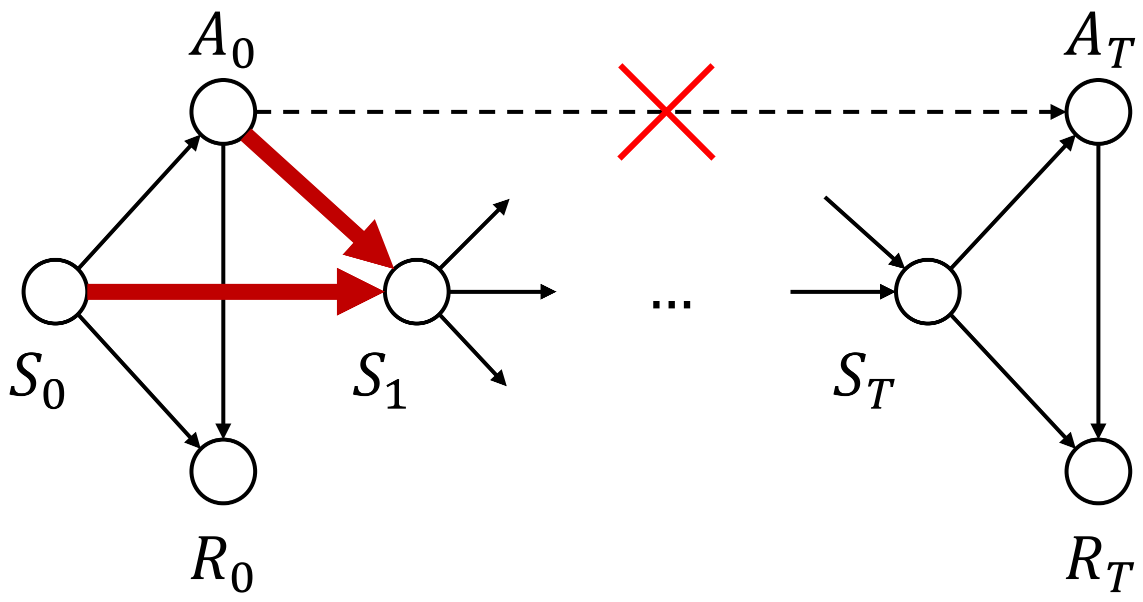
$$p(S_t \mid S_0, \dots, S_{t-1}, A_0, \dots, A_{t-1}) = p(S_t \mid S_{t-1}, A_{t-1})$$

- Markov **reward** function: $p(R_t \mid S_t, A_t) = p(R_t \mid S_0, \dots, S_{t-1}, A_0, \dots, A_{t-1})$

- Markov **action** policy $\mu = p(A_t \mid S_t) = p(A_t \mid S_0, \dots, S_{t-1}, A_0, \dots, A_{t-1})$

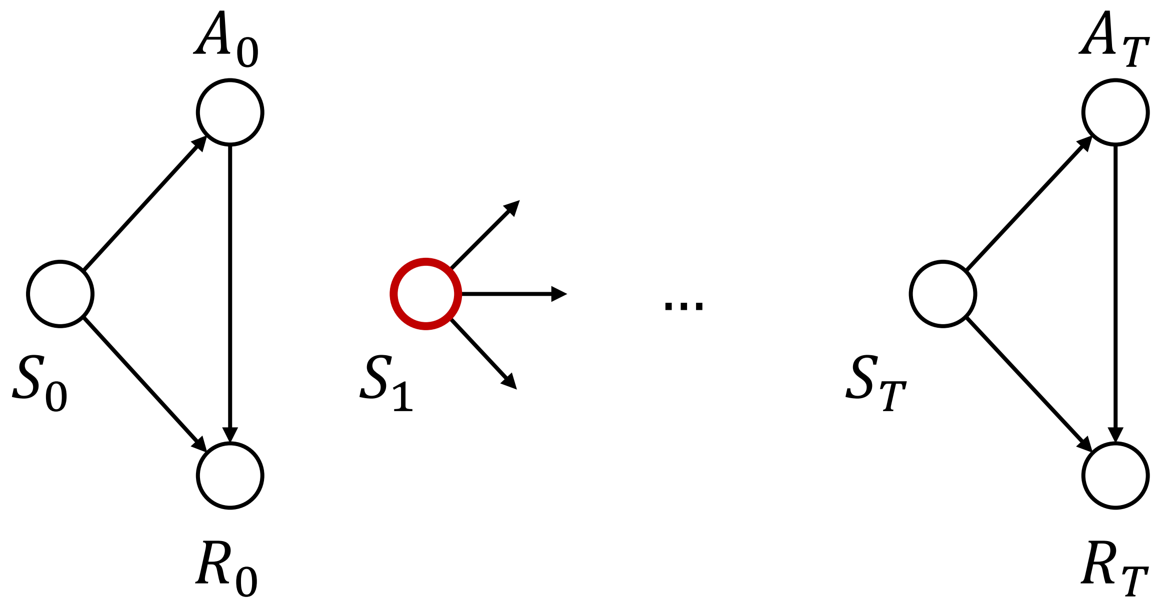
Markov assumption

- State transitions, actions and reward depend only on most recent state-action pair



Contextual bandits (special case)*

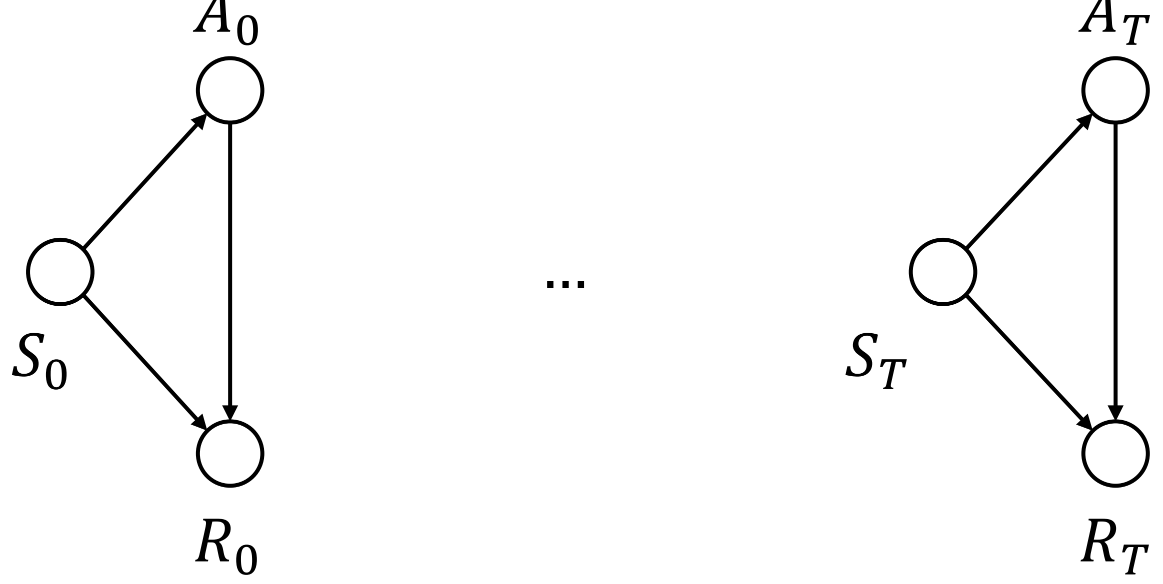
- States are independent: $p(S_t \mid S_{t-1}, A_{t-1}) = p(S_t)$
- Equivalent to **single-step case**: potential outcomes!



* The term "contextual bandits" has connotations of efficient exploration, which is not addressed here

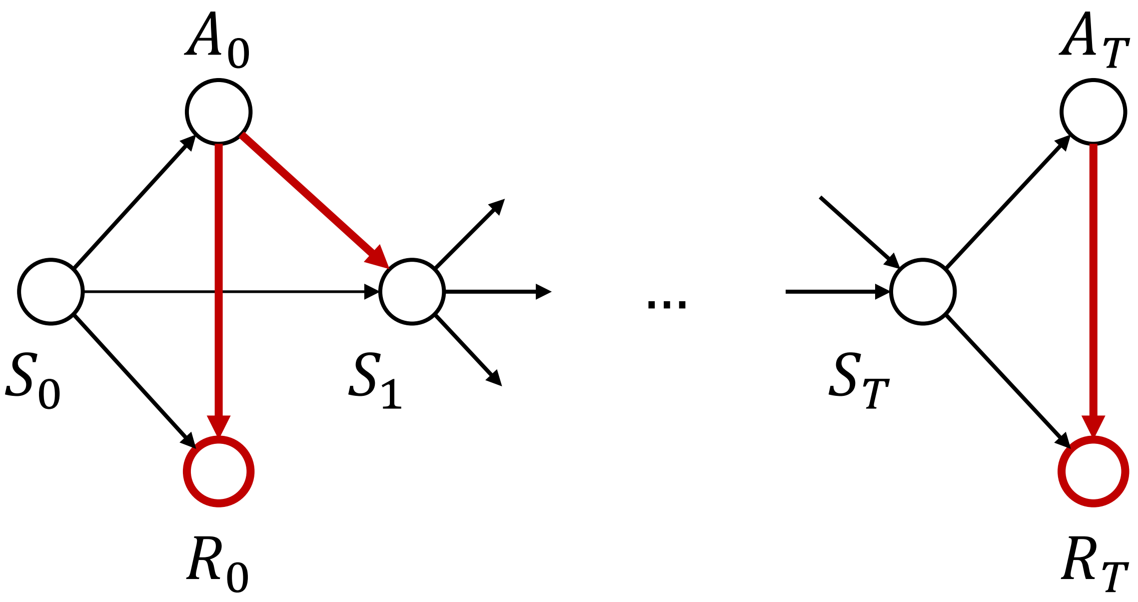
Contextual bandits & potential outcomes

- Think of each state S_i as an i.i.d. patient, the actions A_i as the treatment group indicators and R_i as the outcomes



Goal of RL

- Like previously with causal effect estimation, we are interested in the effects of actions A_t on future rewards



Value maximization

- The goal of most RL algorithms is to maximize the expected cumulative reward—the **value** V_π of its policy π

- **Return**: $G_t = \sum_{s=t}^{T} R_s$ —— Sum of future rewards
- **Value:** $V_\pi = \mathbb{E}_{A_t \sim \pi}[G_0]$ —— Expected sum of rewards under policy π

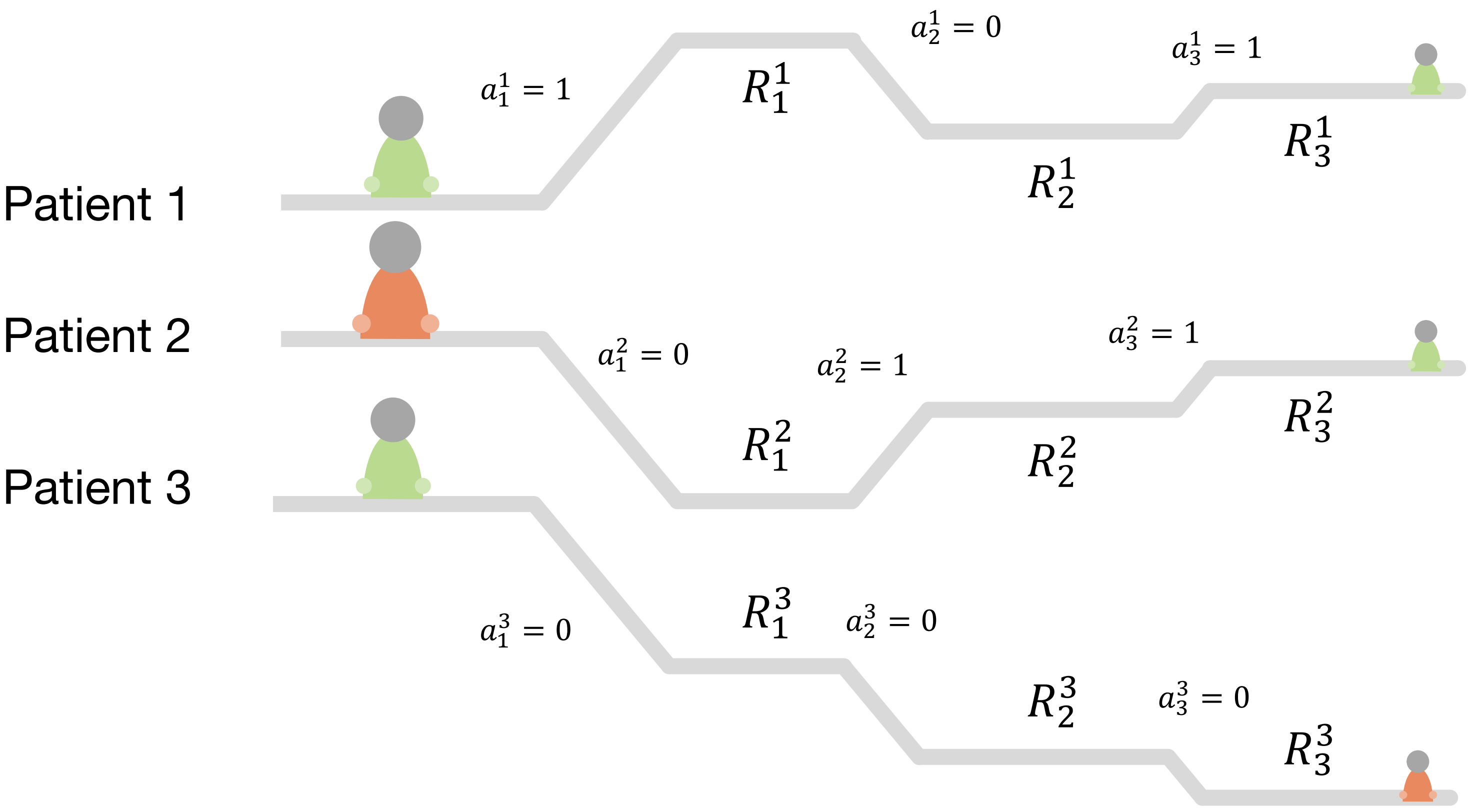
- The expectation is taken with respect to scenarios acted out according to the learned **policy** π

Example

Value

$$V_\pi \approx \frac{1}{n}\sum_{i=1}^{n} G^n$$

- Let's say that we have data from a policy π



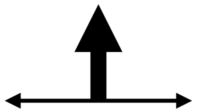
Return

$$G^1 = R_1^1 + R_2^1 + R_3^1$$

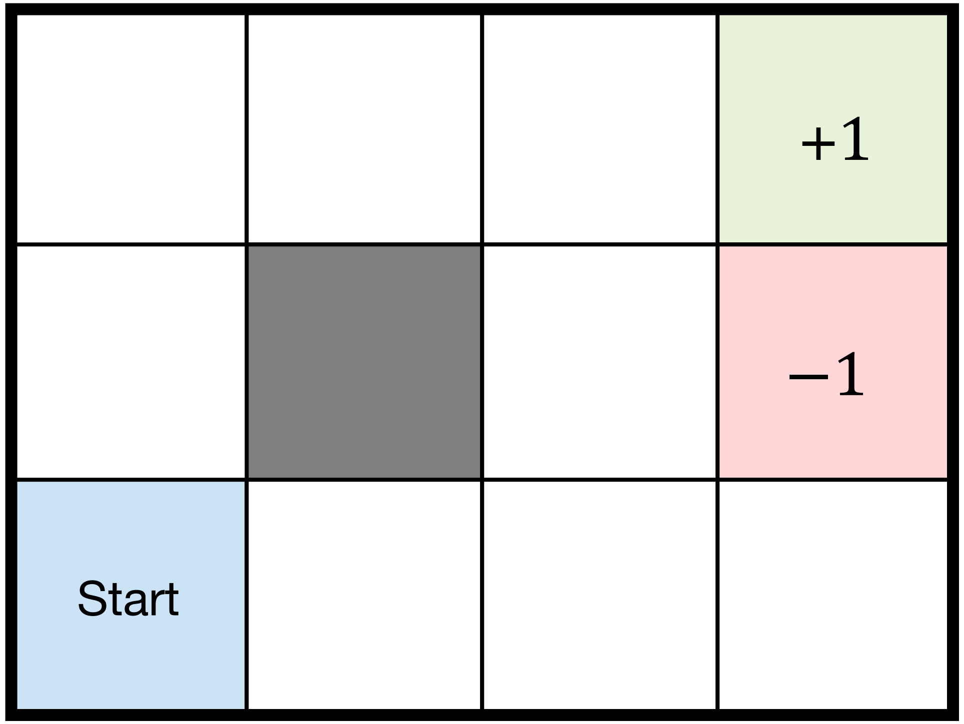
$$G^2 = R_1^2 + R_2^2 + R_3^2$$

$$G^3 = R_1^3 + R_2^3 + R_3^3$$

Robot in a room

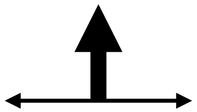
- Stochastic actions 
  $p(\text{ Move up } | A = \text{"}up\text{"}) = 0.8$
  Available non-opposite moves have uniform probability

- Rewards:
  +1 at [4,3] (terminal state)
  -1 at [4,2] (terminal)
  -0.04 per step



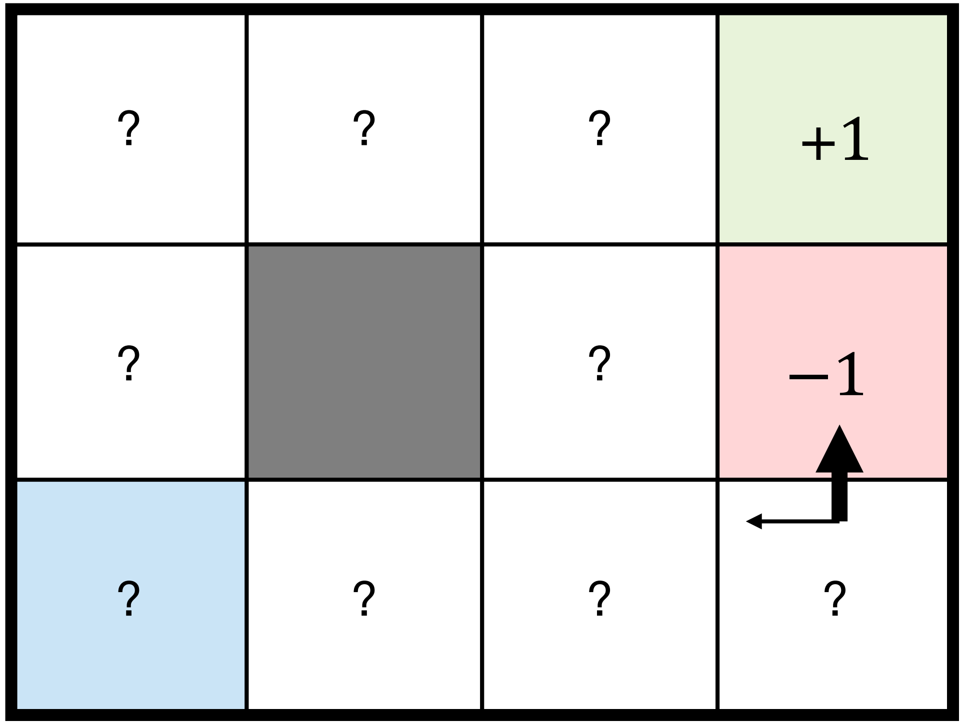
Slide from Peter Bodik

Robot in a room

- Stochastic actions 
  $p(\text{Move up} \mid A = \text{"}up\text{"}) = 0.8$
  Available non-opposite moves have uniform probability

- Rewards:
  +1 at [4,3] (terminal state)
  -1 at [4,2] (terminal)
  -0.04 per step

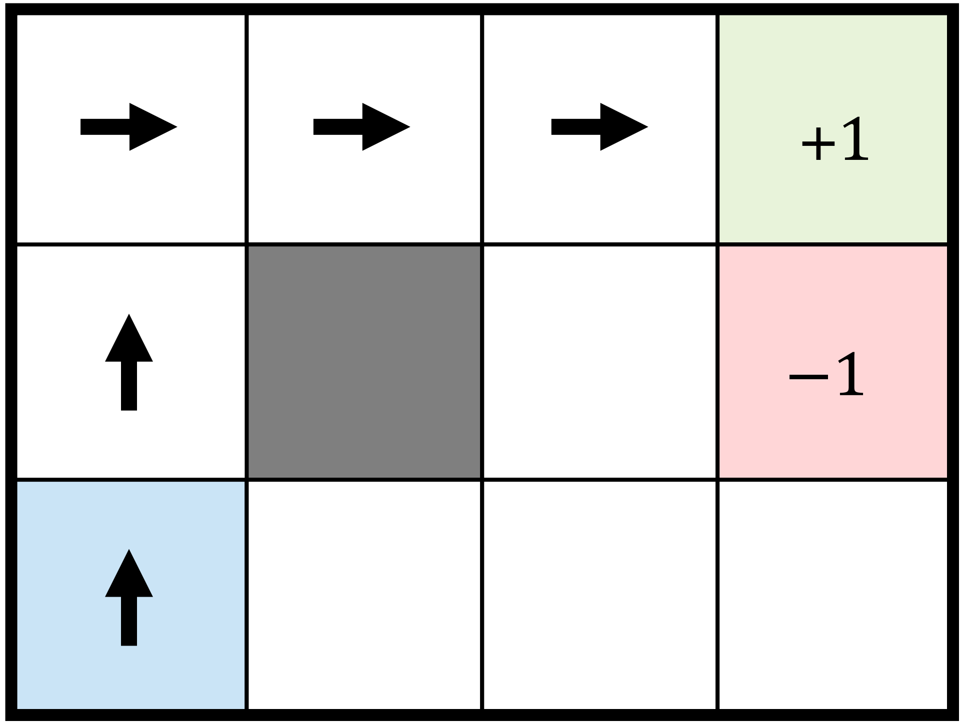
What is the optimal policy?



Slide from Peter Bodik

Robot in a room

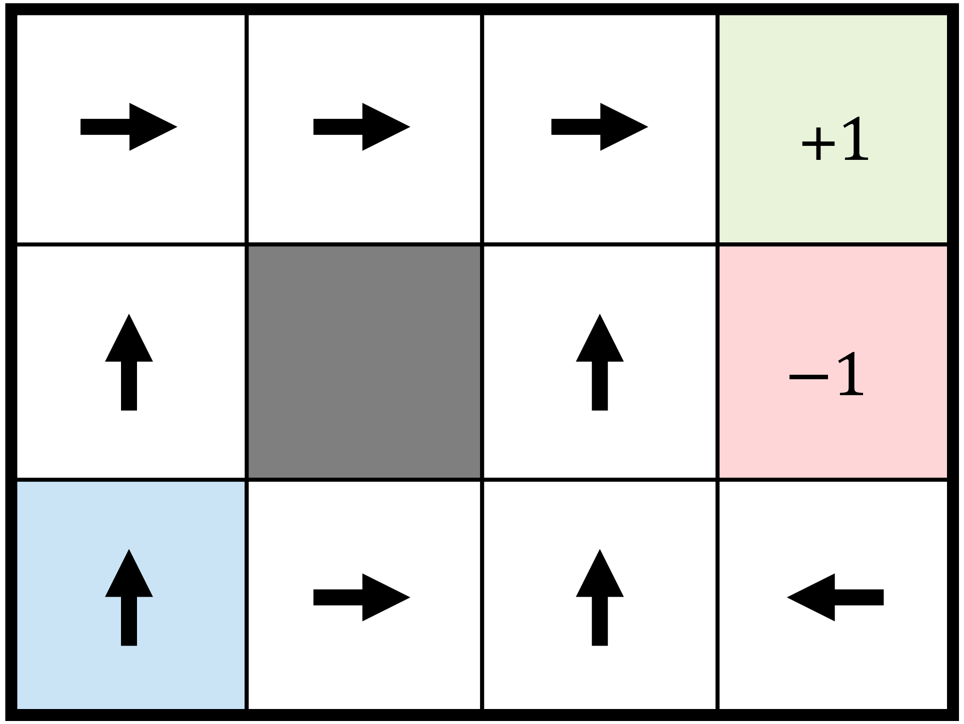
- The following is the optimal policy/trajectory under **deterministic transitions**
- Not achievable in our stochastic transition model



Slide from Peter Bodik

Robot in a room

- Optimal policy
- **How can we learn this?**



Slide from Peter Bodik

1. Decision processes
2. **Reinforcement learning**
3. Learning from batch (off-policy) data
4. Reinforcement learning in healthcare

Paradigms*

Model-based RL	**Value-based RL**	**Policy-based RL**
Transitions	Value/return	Policy
$p(S_t \mid S_{t-1}, A_{t-1})$	$p(G_t \mid S_t, A_t)$	$p(A_t \mid S_t)$
G-computation	**Q-learning**	REINFORCE
MDP estimation	G-estimation	Marginal structural models

*We focus on off-policy RL here

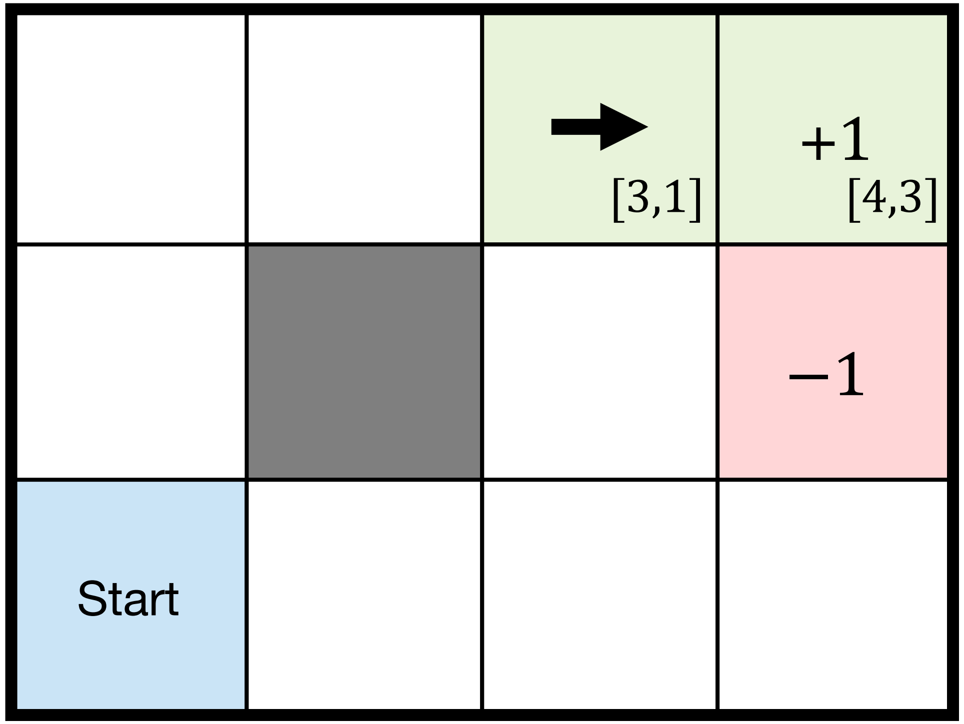
Paradigms*

Model-based RL	Value-based RL	Policy-based RL
Transitions	Value/return	Policy
$p(S_t \mid S_{t-1}, A_{t-1})$	$p(G_t \mid S_t, A_t)$	$p(A_t \mid S_t)$
G-computation	**Q-learning**	REINFORCE
MDP estimation	G-estimation	Marginal structural models

*We focus on off-policy RL here

Dynamic programming

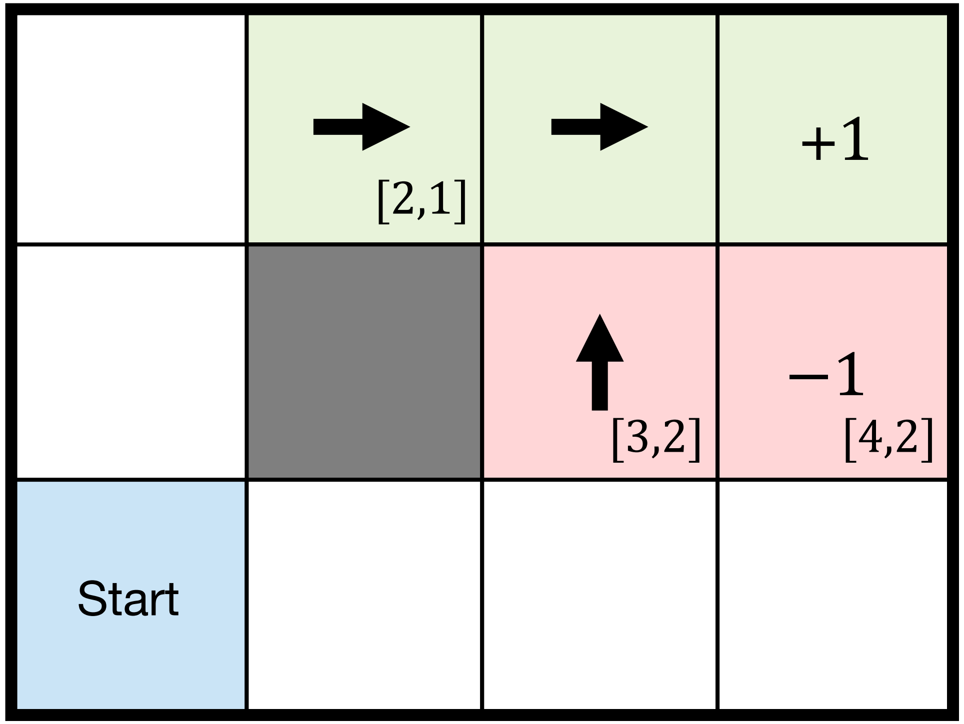
- Assume that we know how good a state-action pair is
- **Q:** Which end state is the best? **A:** [4,3]
- **Q:** What is the best way to get there? **A:** Only [3,1]



Slide from Peter Bodik

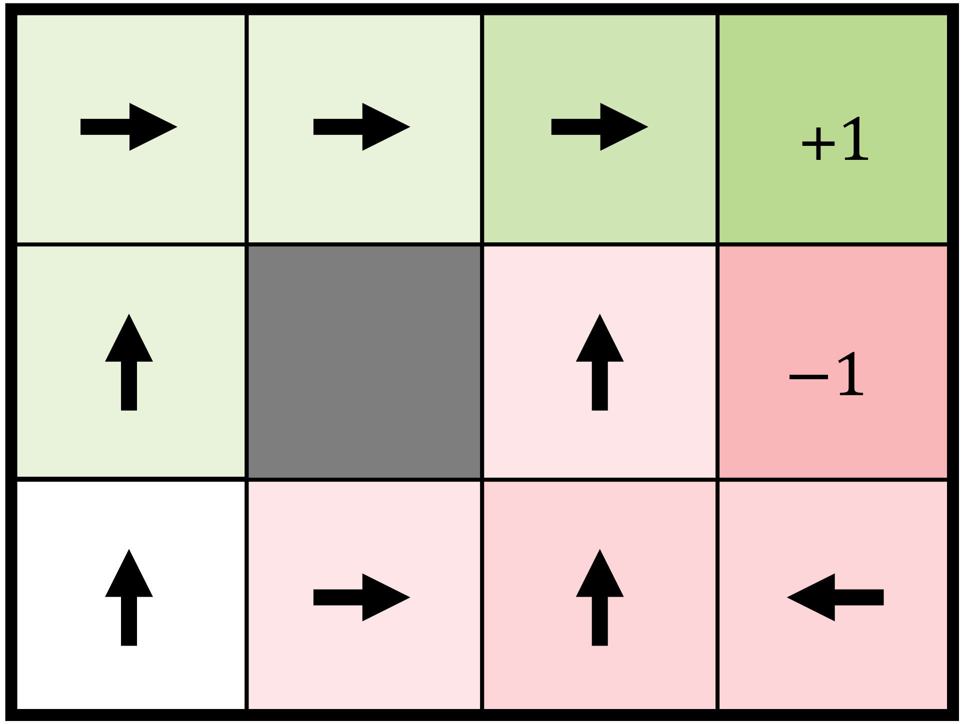
Dynamic programming

- [2,1] is slightly better than [3,2] because of the risk of transitioning to [4,2] from [3,2]
- Which is the best way to [2,1]?



Dynamic programming

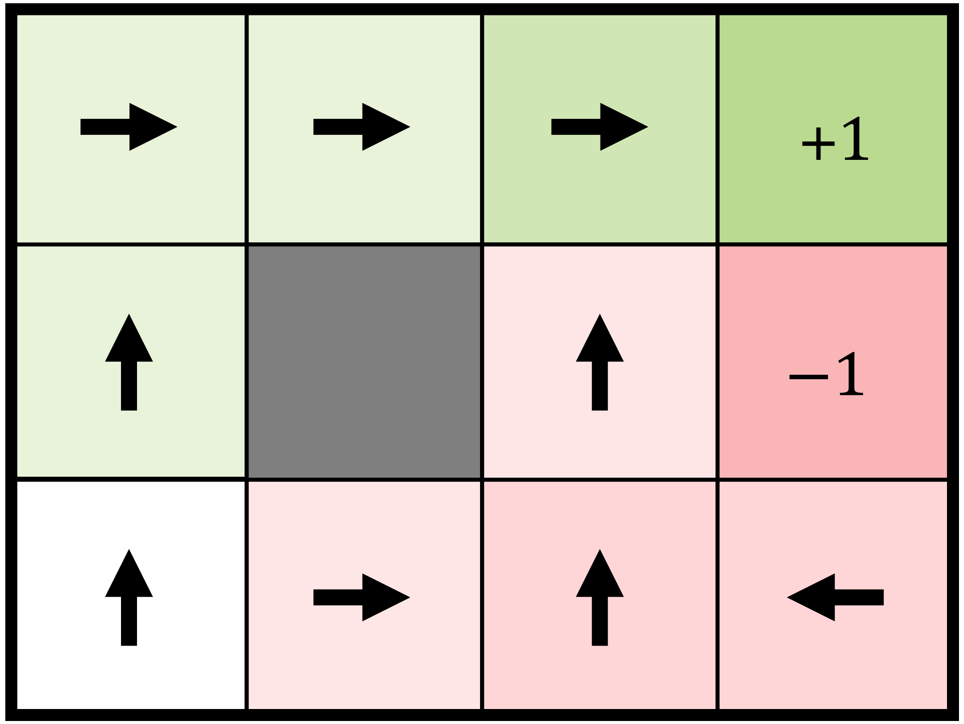
- The idea of dynamic programming for reinforcement learning is to **recursively** learn the best action/value in a previous state given the best action/value in future states



Slide from Peter Bodik

Dynamic programming

- **Next:** How do we get the value of each state?



Slide from Peter Bodik

Q-learning

- Q-learning is a value-based reinforcement learning method

- **Recall:** The value of a state s under a policy π is

$$v_\pi(s) \coloneqq \mathbb{E}_\pi[G_t \mid S_t = s] \quad \coloneqq \mathbb{E}_\pi[\sum_{j=0}^{\infty} \gamma^j R_{t+j} \mid S_t = s]$$

Reward discount factor*

*Mathematical tool more than anything

Q-learning

- Q-learning is a value-based reinforcement learning method

- The value of a **state** s under a policy π is

$$v_\pi(s) \coloneqq \mathbb{E}_\pi[G_t \mid S_t = s] \quad \coloneqq \mathbb{E}_\pi[\textstyle\sum_{j=0}^{\infty} \gamma^j R_{t+j} \mid S_t = s]$$

Reward discount factor*

- The value of a **state-action pair** (s, a) is

$$q_\pi(s, a) \coloneqq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

*Mathematical tool more than anything

Q-learning

- Q-learning attempts to **estimate q_π** with a function $Q(s,a)$ such that π is the deterministic policy

$$\pi(s) = \arg\max_{\mathrm{a}} Q(s,a)$$

- The best Q is the best **state-action value** function

$$Q^*(s,a) = \max_\pi q_\pi(s,a) =: q^*(s,a)$$

Bellman equation

- For the optimal Q-function q^*, "**Bellman optimality**" holds*

$$q^*(s,a) = \mathbb{E}_\pi\left[R_t + \gamma \max_{a'} q^*(S_{t+1}, a') \mid S_t = s, A_t = a\right]$$

State-action value | Immediate reward | Future (discounted) rewards*

- Look for functions with this property!

*A necessary property for optimality of dynamic programming

Q-learning with discrete states

- If states are **discrete**, $s \in \{0, \ldots, K\}$, Q-learning can be solved exactly using dynamic programming (for small enough K)*

- Initialize a **table** of $Q(s, a)$
- Repeat

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_t + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

Learning rate

*Converges to the optimal q^* if all state-action pairs visited over and over again

Q-learning with discrete states

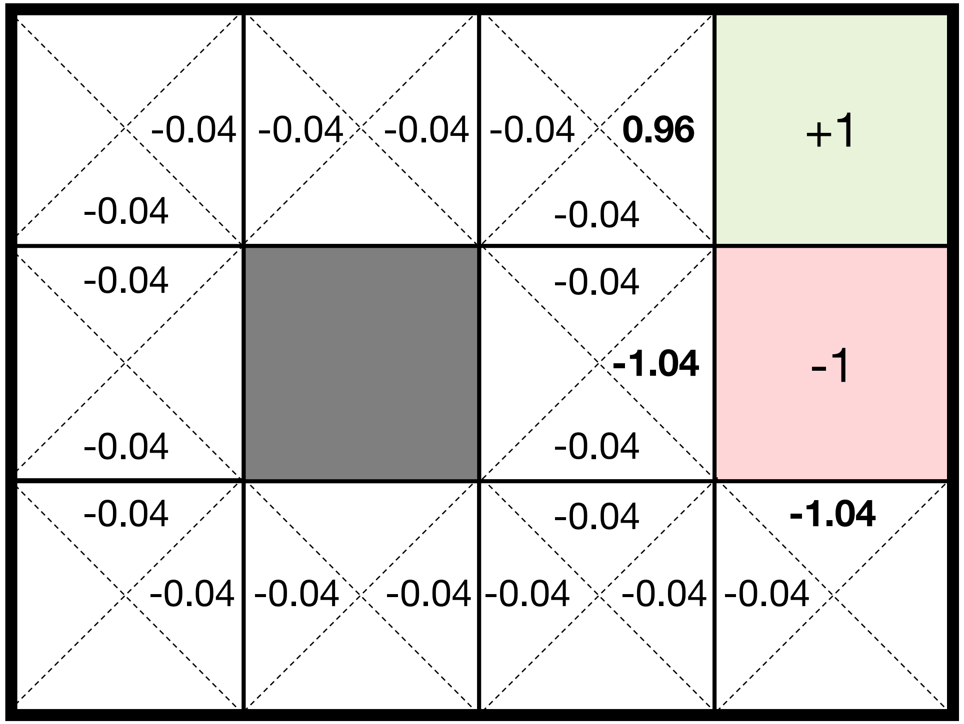
1. Initialize $Q(s,a) = 0$, let $\alpha, \gamma = 1$
2. Repeat

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_t + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

Assume that transitions are deterministic for now

Let each state-pair be visited in order, over and over*

Q-table



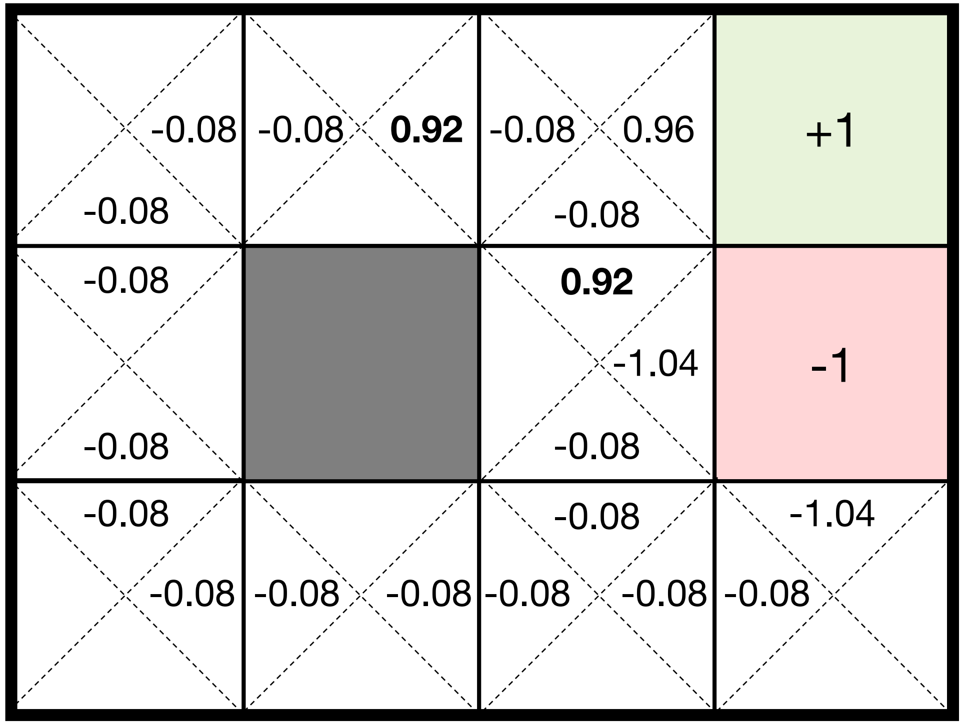
* We will come back to this

Q-learning with discrete states

1. Initialize $Q(s, a) = 0$, let $\alpha, \gamma = 1$
2. Repeat

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_t + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

Q-table

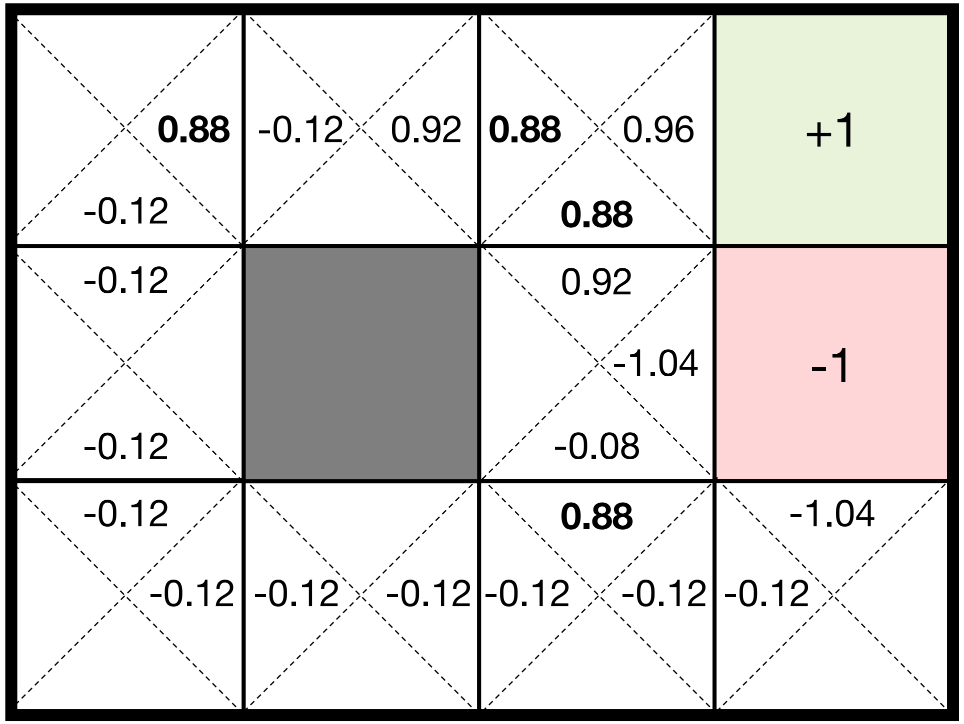


Q-learning with discrete states

1. Initialize $Q(s, a) = 0$, let $\alpha, \gamma = 1$
2. Repeat

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_t + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right]$$

Q-table

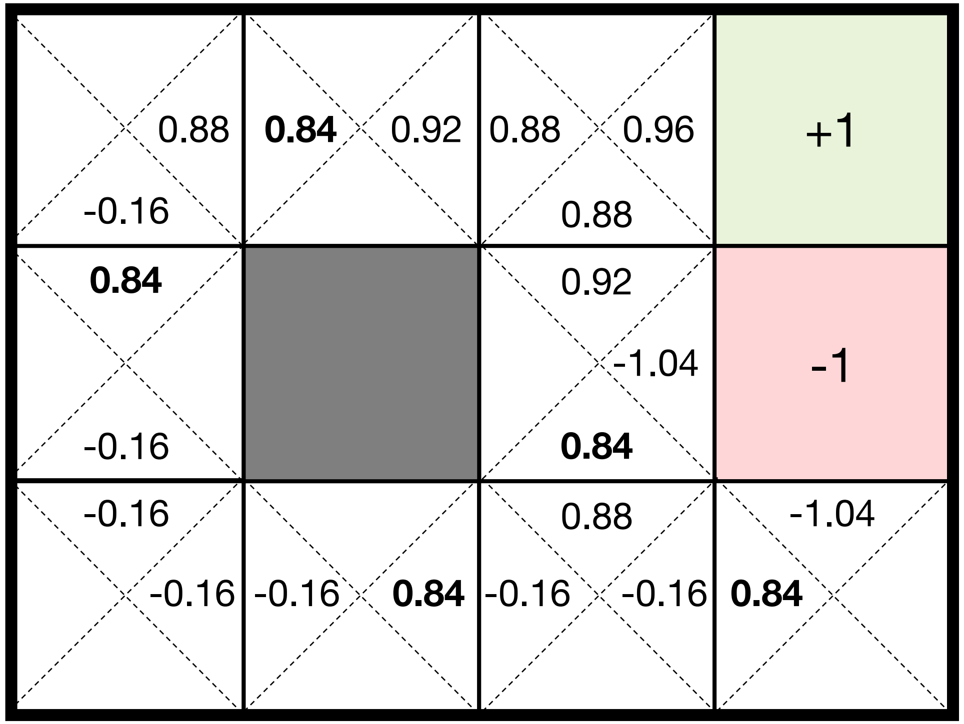


Q-learning with discrete states

1. Initialize $Q(s, a) = 0$, let $\alpha, \gamma = 1$
2. Repeat

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_t + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

Q-table

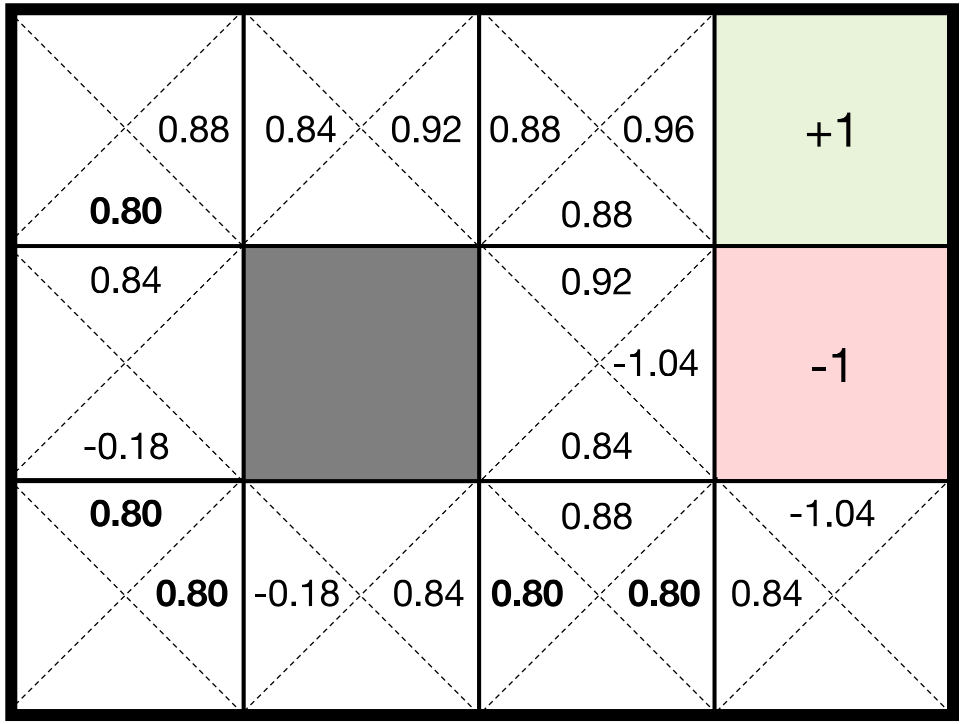


Q-learning with discrete states

1. Initialize $Q(s,a) = 0$, let $\alpha, \gamma = 1$
2. Repeat

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_t + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right]$$

Q-table

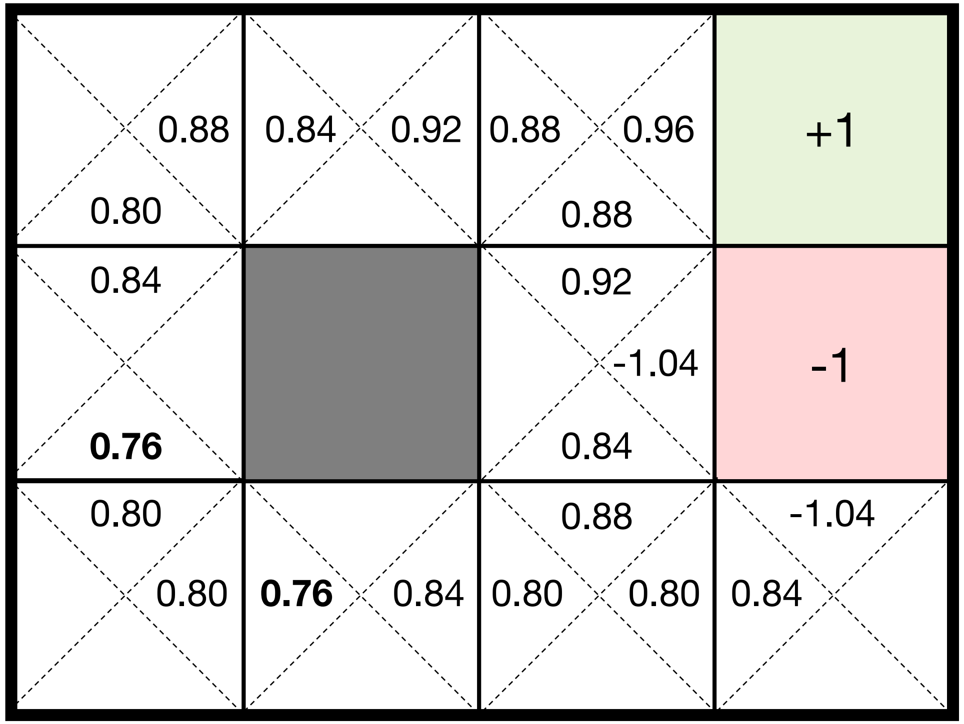


Q-learning with discrete states

1. Initialize $Q(s, a) = 0$, let $\alpha, \gamma = 1$
2. Repeat

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_t + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right]$$

Q-table



Fitted Q-learning (with function approximation)

- If the number of states K is large or S_t is not discrete, we cannot maintain a table for $Q(s,a)$

- Instead, we may represent $Q(s,a)$ by a **function** Q_θ and minimize the risk

$$R(Q_\theta) = \mathbb{E}_\pi\left[\left(R + \gamma \max_{a'} \hat{Q}(S', A') - Q_\theta(S, A)\right)^2\right]$$

Old estimate of Q — Current estimate

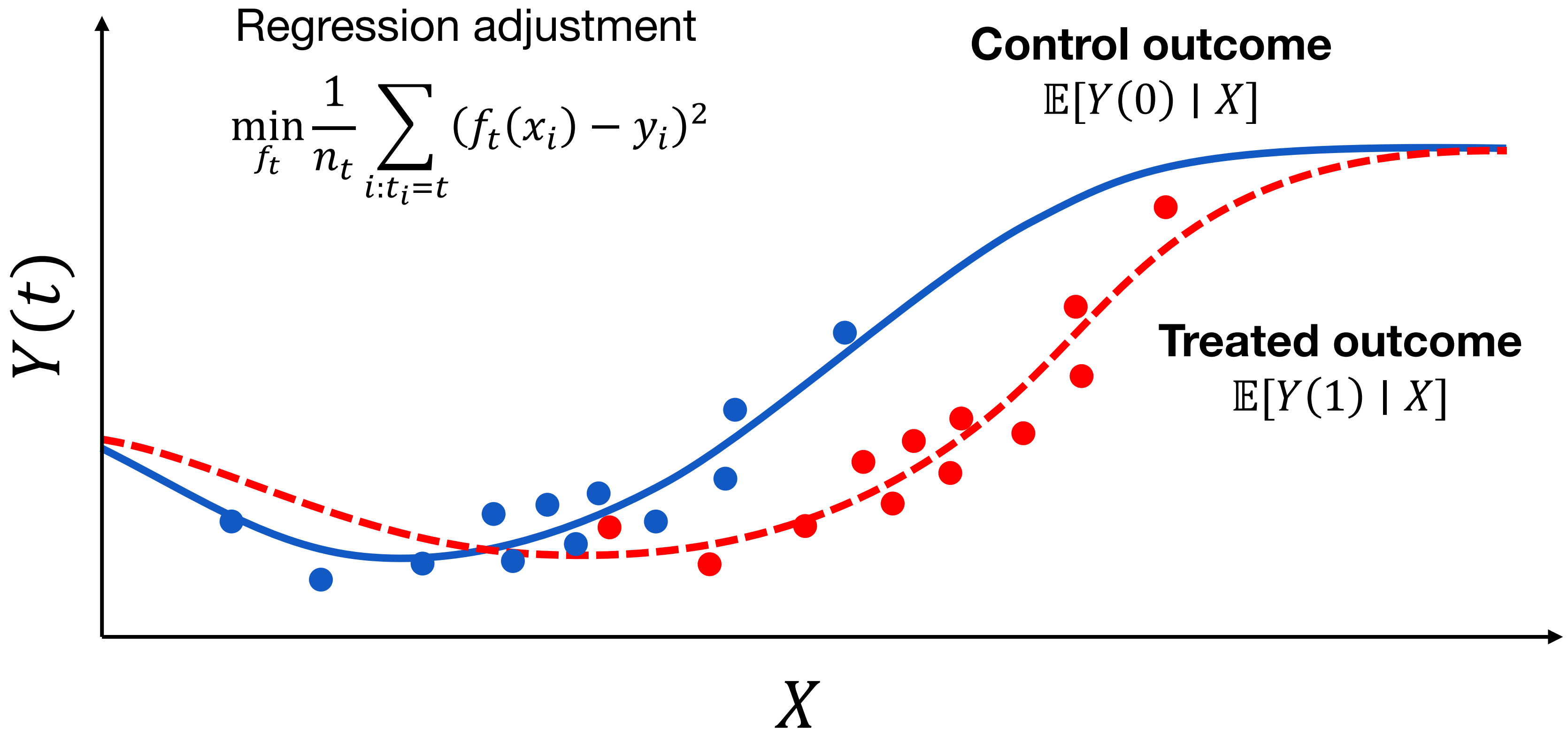
Bellman equation (one step)

- In the one-step case (no future states)

$$R(Q_\theta) = \mathbb{E}_\pi\left[\left(R_t + \cancel{\gamma \max_{a'} \hat{Q}(S', a')} - Q_\theta(S, A)\right)^2\right]$$
$$= \mathbb{E}_\pi\left[\left(R_t - Q_\theta(S, A)\right)^2\right]$$

- Finding $q(s, a)$ is analogous to finding expected potential outcomes $\mathbb{E}[R(a) \mid S = s]$ in the one-step case!

Recall: Potential outcomes



Fitted Q-learning as covariate adjustment

- Fitted Q-learning is like covariate adjustment (regression) with a moving target (which is updated during learning)

Choice of loss, (here squared)

$$R(Q_\theta) = \mathbb{E}_\pi\left[\left(\hat{G}(S,A,S',R) - Q_\theta(S,A)\right)^2\right]$$

$$\hat{G}(S,A,S',R) := R + \gamma \max_{a'} \hat{Q}(S',a')$$

Expectation over transitions (s,a,s',r) — Target — Prediction

Off-policy learning

- Where does our data come from?

$$R(Q_\theta) = \mathbb{E}_\pi \left[\left(R + \gamma \max_{a'} \hat{Q}(S', a') - Q_\theta(S, A) \right)^2 \right]$$

How do we evaluate this expectation?

- "What are the inputs and outputs of our regression?"

- Alternate between updates of $\hat{Q}$ and Q_θ

Exploration in RL

- Tuples (s, a, s', r) may be obtained by:
  - **On-policy exploration**—"Playing the game" with the current policy
  - **Randomized trials**—Executing a sequentially random policy
  - **Off-policy (observational)**—E.g., healthcare records

- The latter is most relevant to us!

1. Decision processes
2. Reinforcement learning paradigms
3. **Learning from batch (off-policy) data**
4. Reinforcement learning in healthcare

Off-policy learning

- Trajectories $(s_1, a_1, r_1), \ldots, (s_T, a_T, r_T)$,of states s_t, actions a_t, and rewards r_t observed in e.g. medical record
- Actions are drawn according to a behavior policy μ, but we want to know the value of a new policy π
- Learning policies from this data is **at least as hard** as estimating treatment effects from observational data

Assumptions for (off-policy) RL

- Sufficient conditions for identifying value function

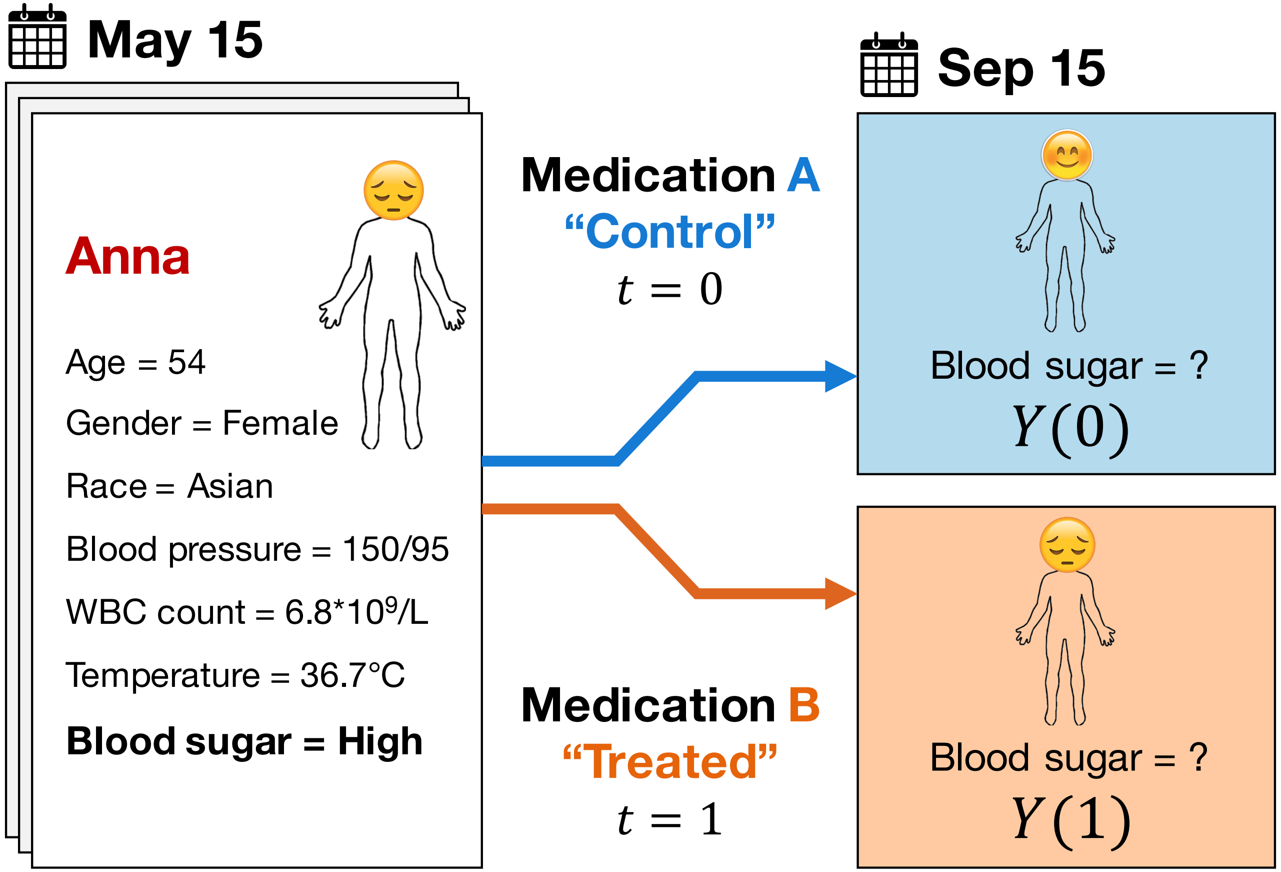
Single-step case	Sequential case
Strong ignorability:	**Sequential randomization**:
$Y(0), Y(1) \perp\!\!\!\perp T \mid X$	$G(\ldots) \perp\!\!\!\perp A_t \mid \bar{S}_t, \bar{A}_{t-1}$
“No *hidden* confounders”	“Reward indep. of policy given history”
Overlap:	**Positivity**:
$\forall x, t\colon\ p(T = t \mid X = x) > 0$	$\forall a, t\colon\ p(A_t = a \mid \bar{S}_t, \bar{A}_{t-1}) > 0$
“All actions possible”	“All actions possible at all times”

Assumptions for (off-policy) RL

- Sufficient conditions for identifying value function

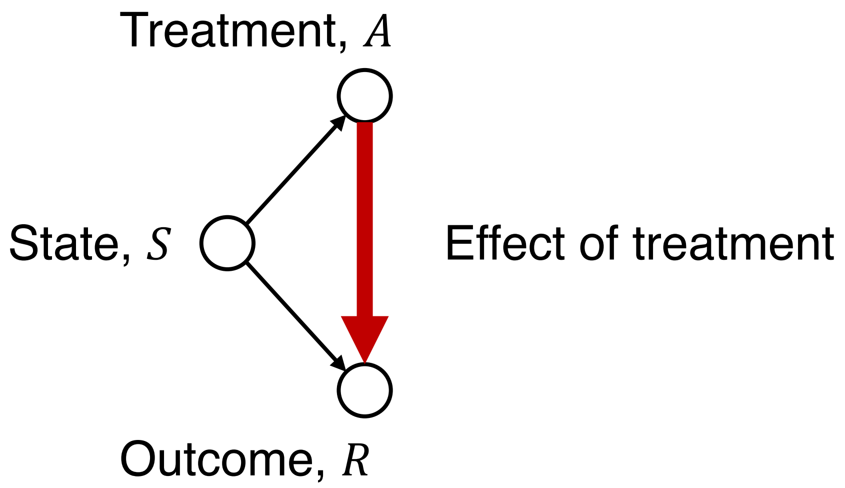
Single-step case	**Sequential case**
Strong ignorability: $Y(0), Y(1) \perp\!\!\!\perp T \mid X$ "No *hidden* confounders"	**Sequential randomization**: $G(\ldots) \perp\!\!\!\perp A_t \mid \bar{S}_t, \bar{A}_{t-1}$ "Reward indep. of policy given history"
Overlap: $\forall x, t\colon p(T = t \mid X = x) > 0$ "All actions possible"	**Positivity**: $\forall a, t\colon p(A_t = a \mid \bar{S}_t, \bar{A}_{t-1}) > 0$ "All actions possible at all times"

Recap: Learning potential outcomes



Treating Anna once

- We assumed a simple causal graph. This let us identify the causal effect of treatment on outcome from observational data



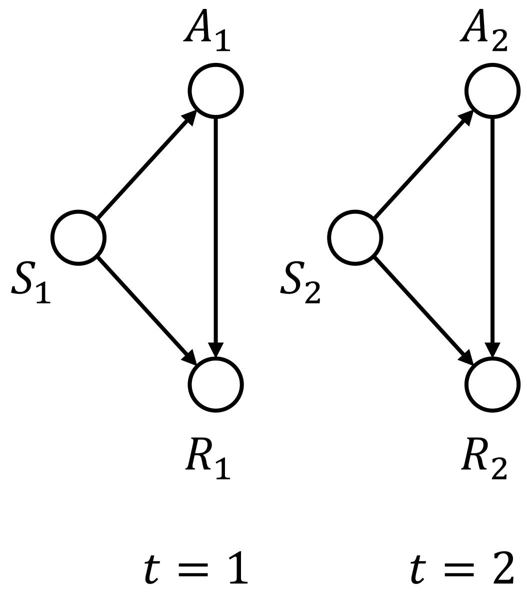
Ignorability

$R(a) \perp\!\!\!\perp A \mid S$

Potential outcome under action a

Treating Anna over time

- Let's add a time point…



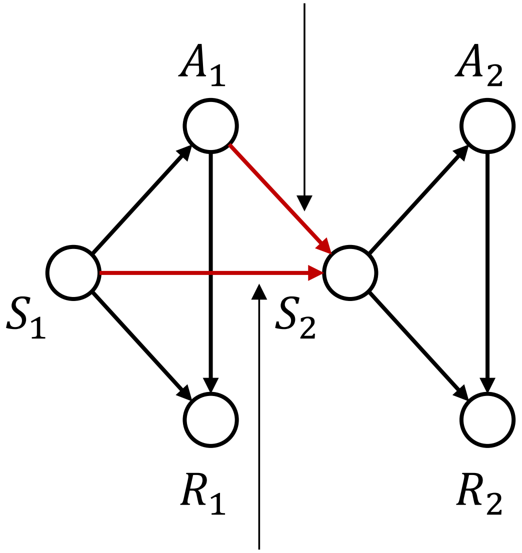
Ignorability

$R_t(a) \perp\!\!\!\perp A_t \mid S_t$

Treating Anna over time

- What influences her state?

Anna's health status depends on how we treated her



It is likely that if Anna is diabetic, she will remain so

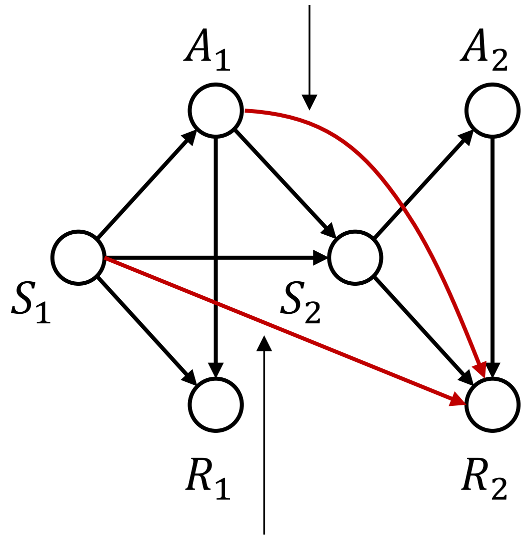
Ignorability

$R_t(a) \perp\!\!\!\perp A_t \mid S_t$

Treating Anna over time

- What influences her state?

The outcome at a later time point may depend on earlier choices



The outcome at a later time may depend on an earlier state

Ignorability

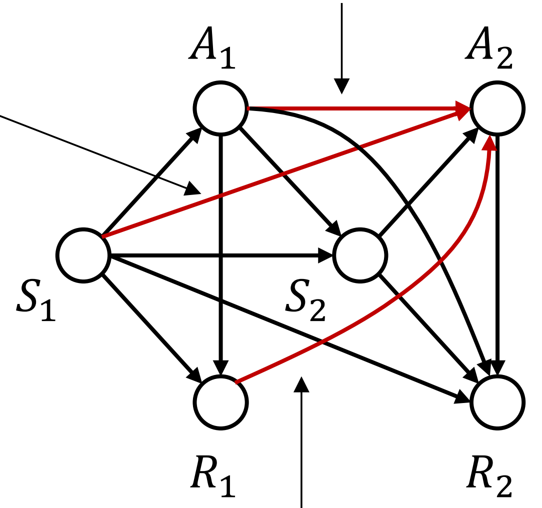
$R_t(a) \perp\!\!\!\perp A_t \mid S_t$

Treating Anna over time

- What influences her state?

If we know that a patient had a symptom previously, it may affect future decisions

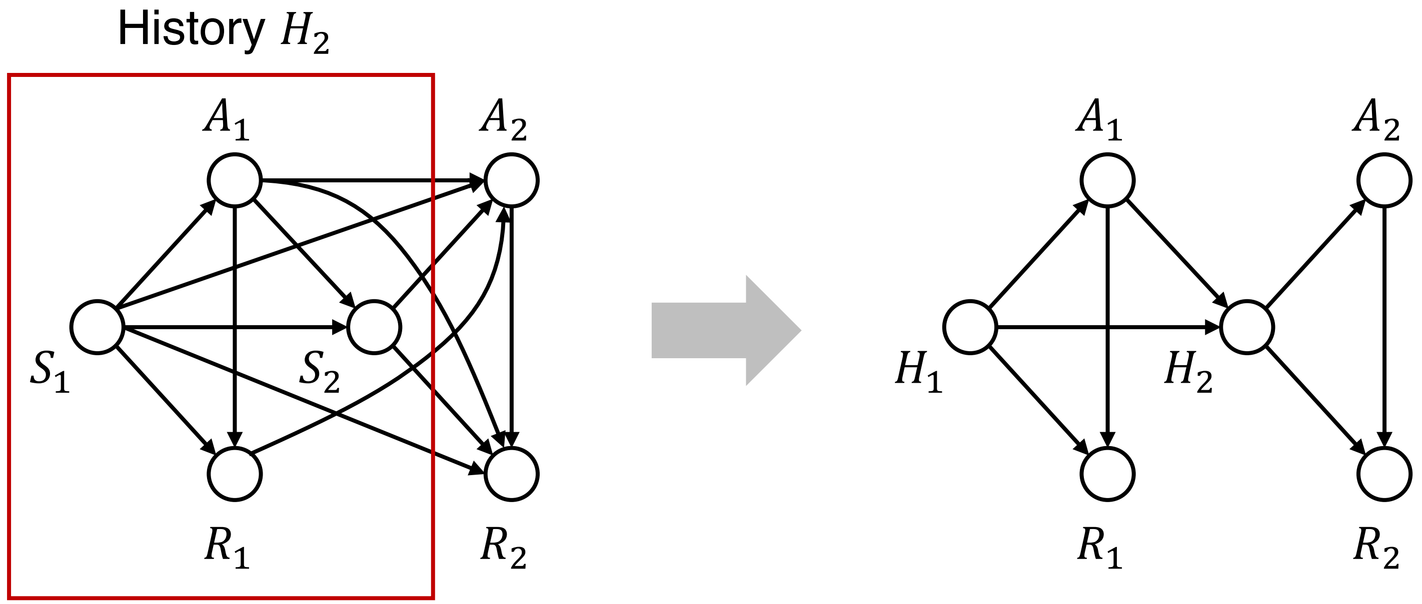
If we already tried a treatment, we might not try it again



If the last treatment was unsuccessful, it may change our next choice

State & ignorability

- To have sequential ignorability, we need to remember history!



Ignorability

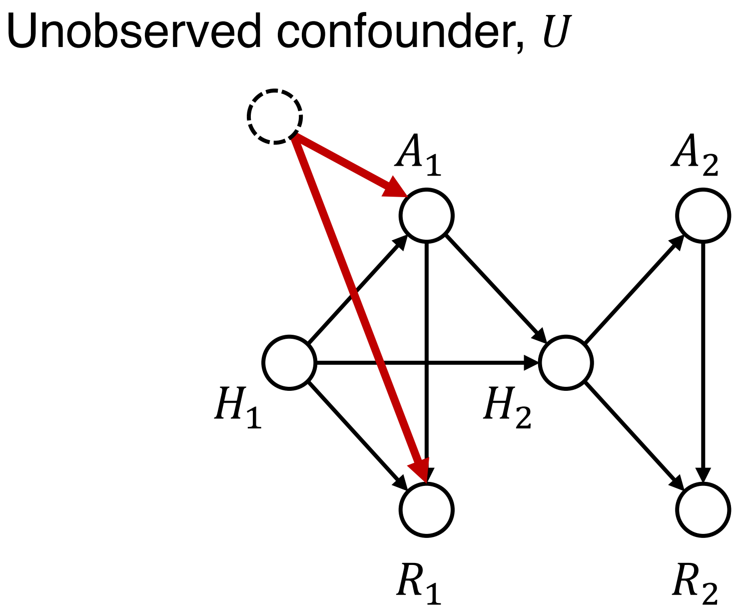
$R_t(a) \perp\!\!\!\perp A_t \mid H_t$

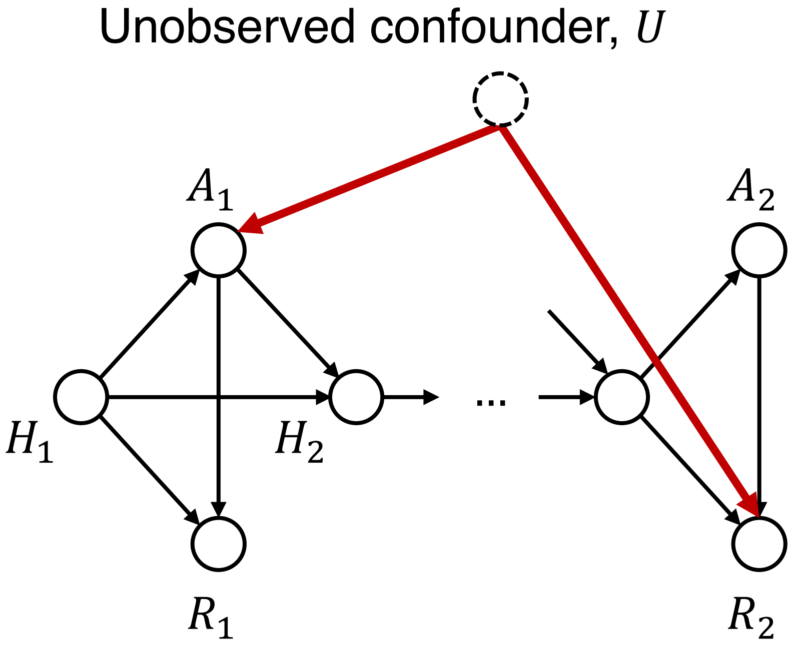
Summarizing history

- The difficulty with history is that its **size grows with time**
- A simple change of the standard MDP is to store the states and actions of a **length k window** looking backwards
- Another alternative is to **learn a summary** function that maintains what is relevant for making optimal decisions, e.g., using an RNN

State & ignorability

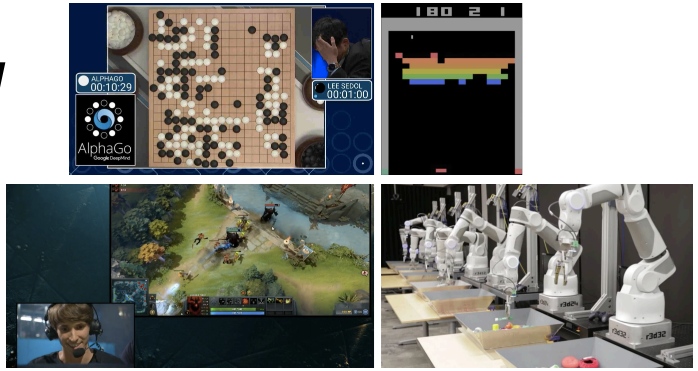
- We cannot leave out unobserved confounders





What made success possible/easier?

- **Full observability**
  Everything important to optimal action is observed



- **Markov** dynamics
  History is unimportant given recent state(s)

- Limitless **exploration** & self-play through simulation
  We can test "any" policy and observe the outcome

- **Noise-less** state/outcome (for games, specifically)

1. Decision processes
2. Reinforcement learning paradigms
3. Learning from batch (off-policy) data
4. **Reinforcement learning in healthcare. Tomorrow!**