

Reinforcement learning

Fredrik D. Johansson

Clinical ML @ MIT

Machine Learning for Healthcare

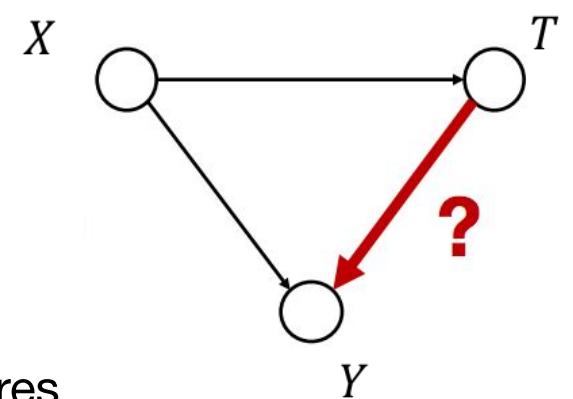
Reminder: Causal effects

- ▶ Potential outcomes under treatment and control, $Y(1), Y(0)$
- ▶ Covariates and treatment, X, T
- ▶ Conditional average treatment effect (CATE)

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

Potential outcomes

Features



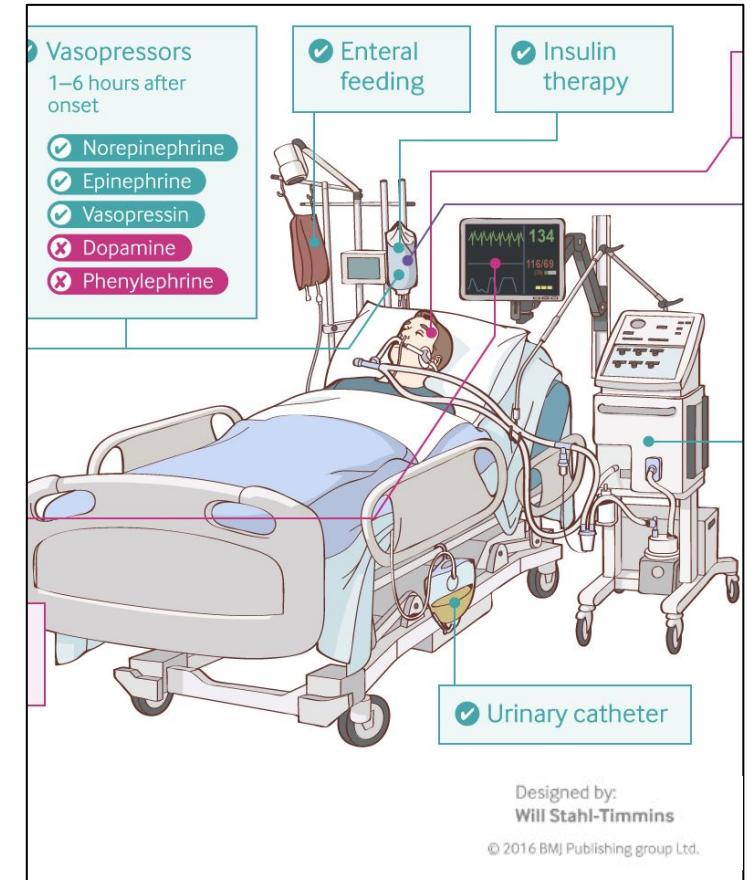
Today: Treatment policies/regimes

- ▶ A **policy π** assigns treatments to patients
(typically depending on their medical history/state)
- ▶ **Example:** For a patient with medical history x ,
$$\pi(x) = \mathbb{I}[CATE(x) > 0]$$

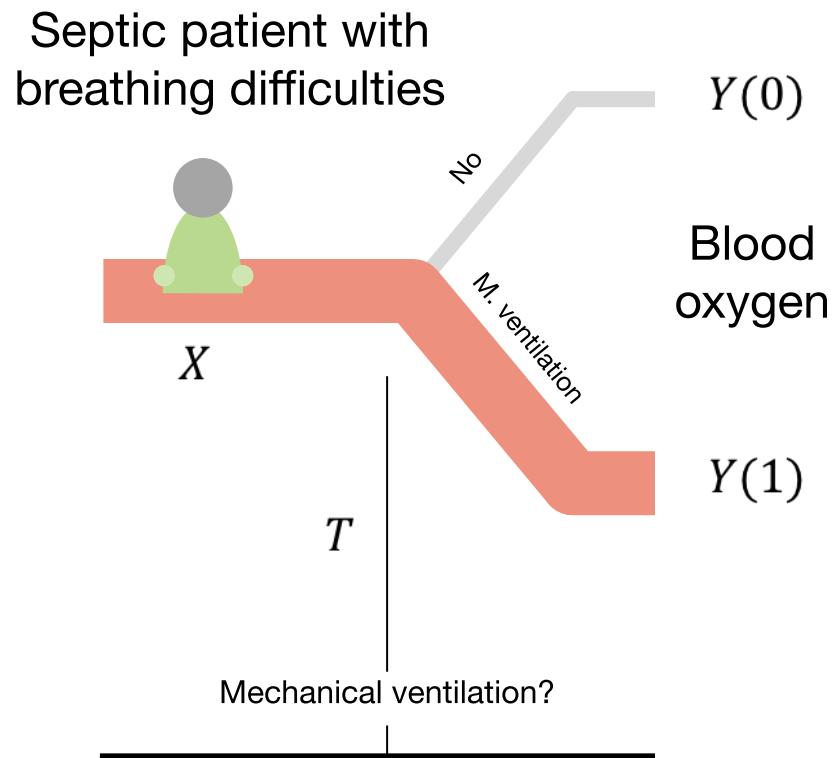
“Treat if effect is positive”
- ▶ Today we focus on policies guided by clinical outcomes
(as opposed to legislation, monetary cost or side-effects)

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, ...



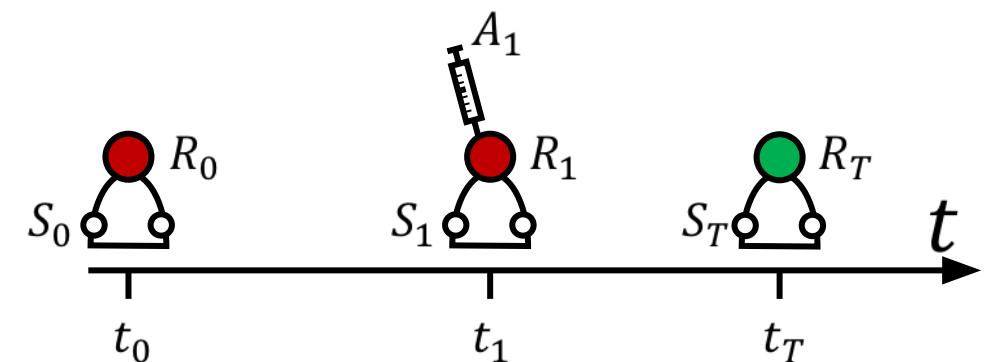
Recall: Potential outcomes



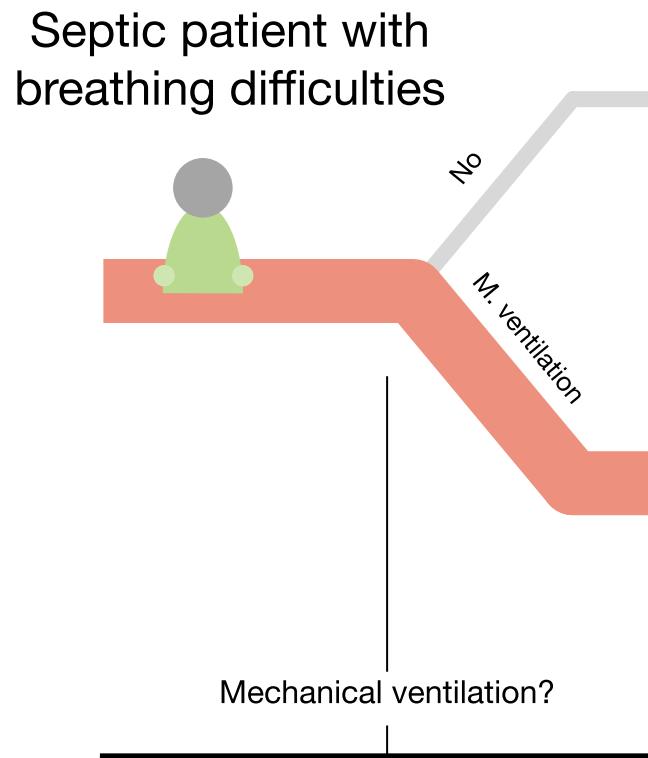
1. Should the patient be put on mechanical ventilation?

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?

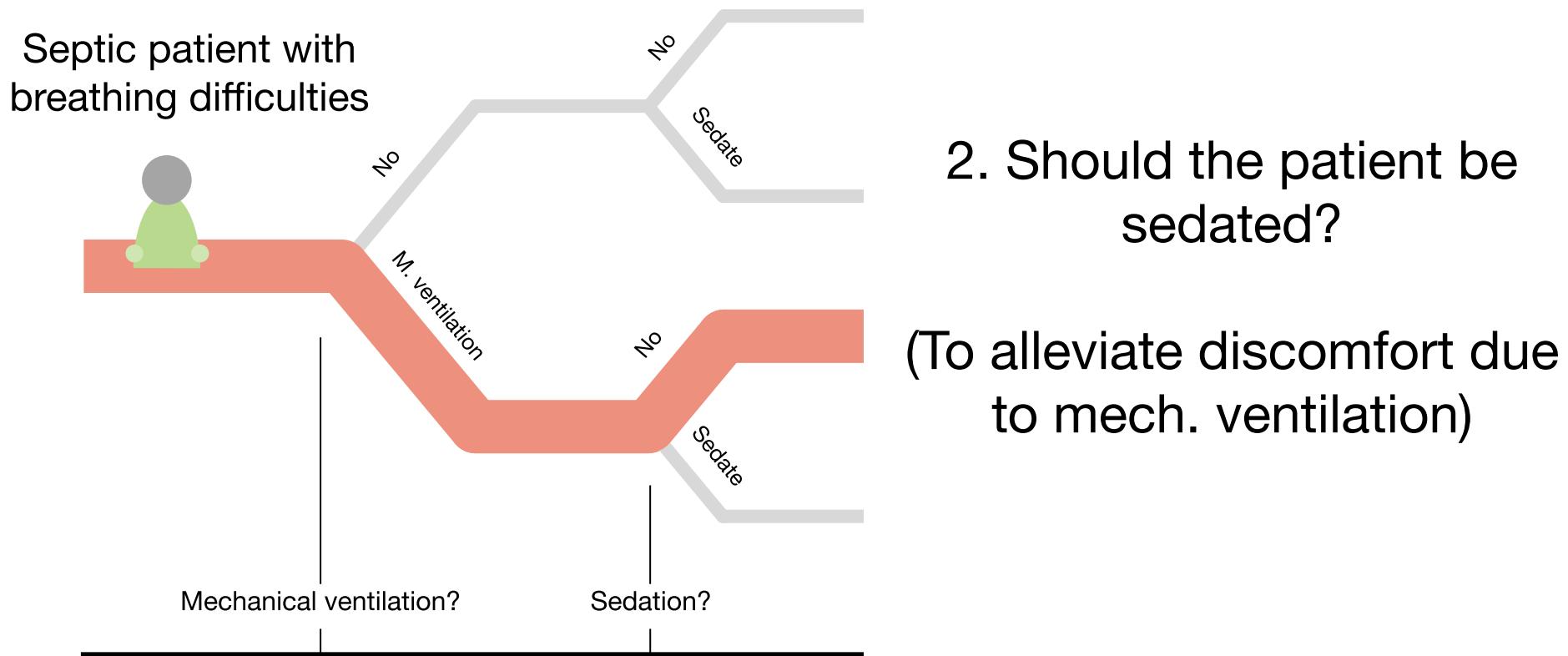


Recall: Potential outcomes

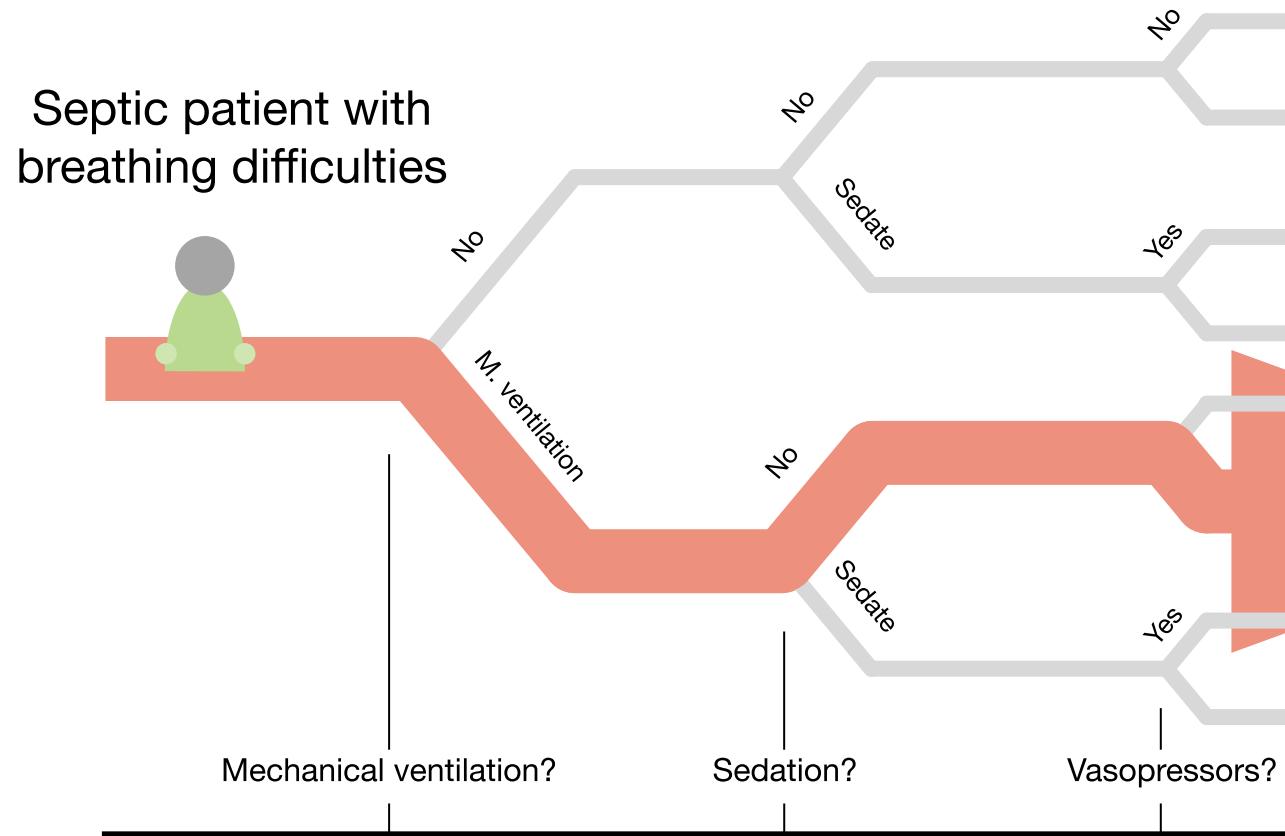


1. Should the patient be put on mechanical ventilation?

Example: Sepsis management



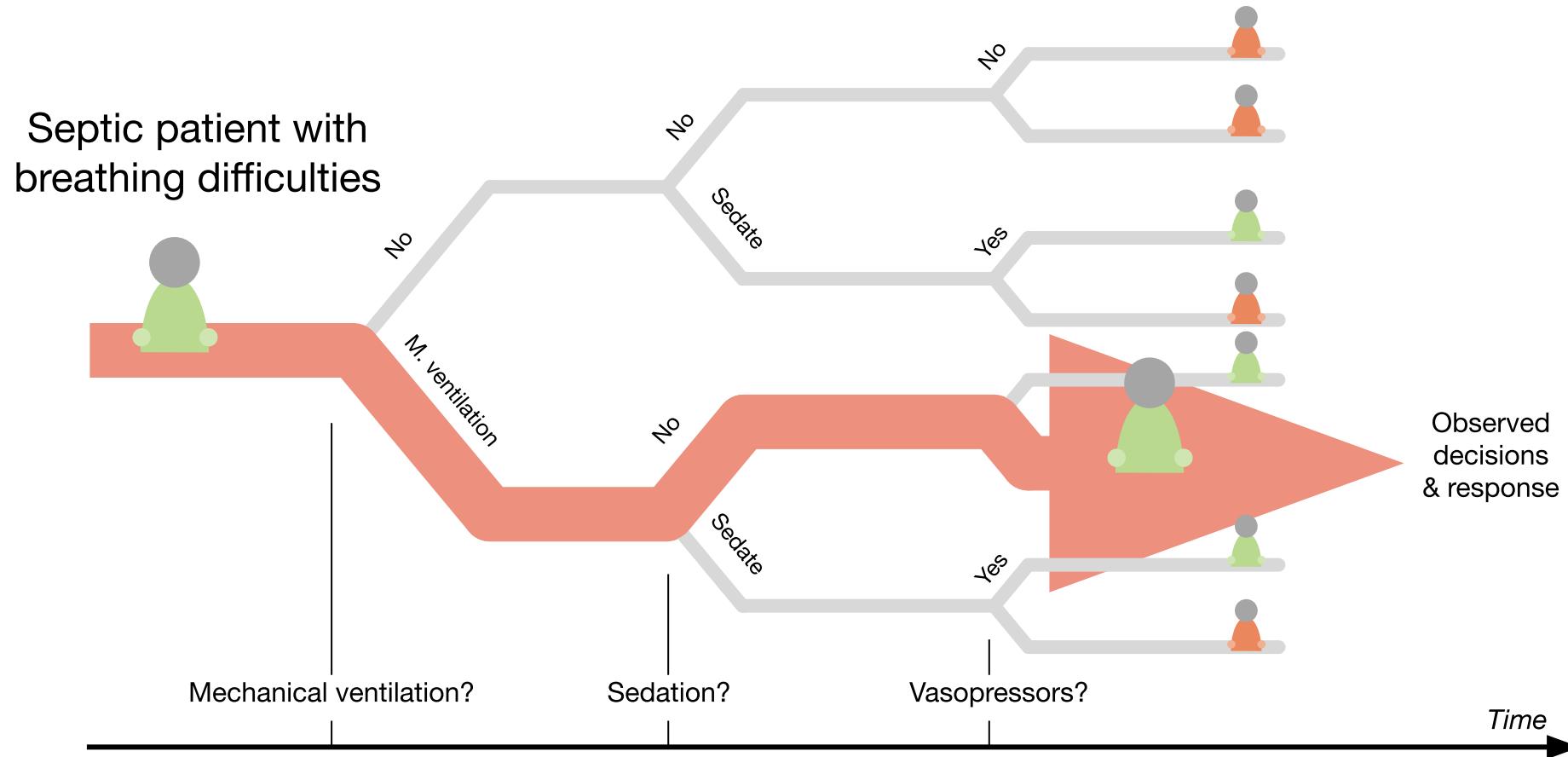
Example: Sepsis management



3. Should we artificially raise blood pressure?

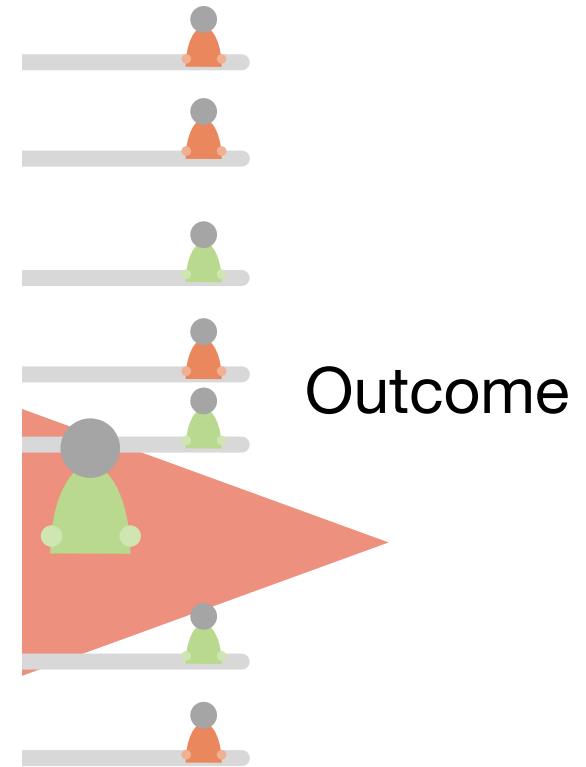
(Which may have dropped due to sedation)

Example: Sepsis management



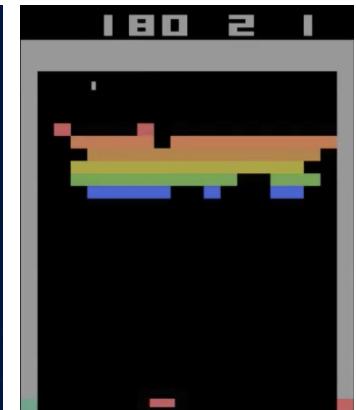
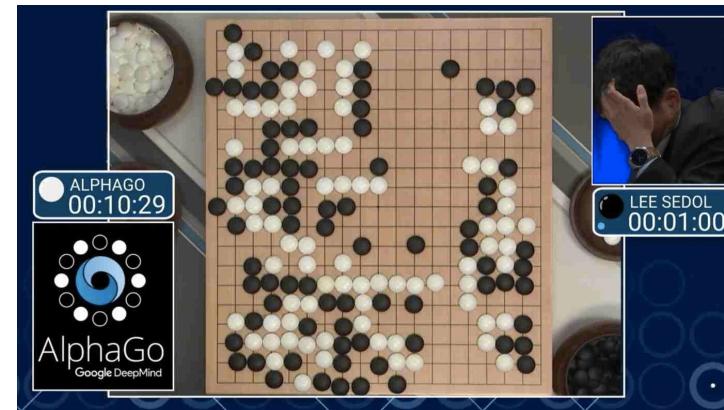
Finding optimal policies

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



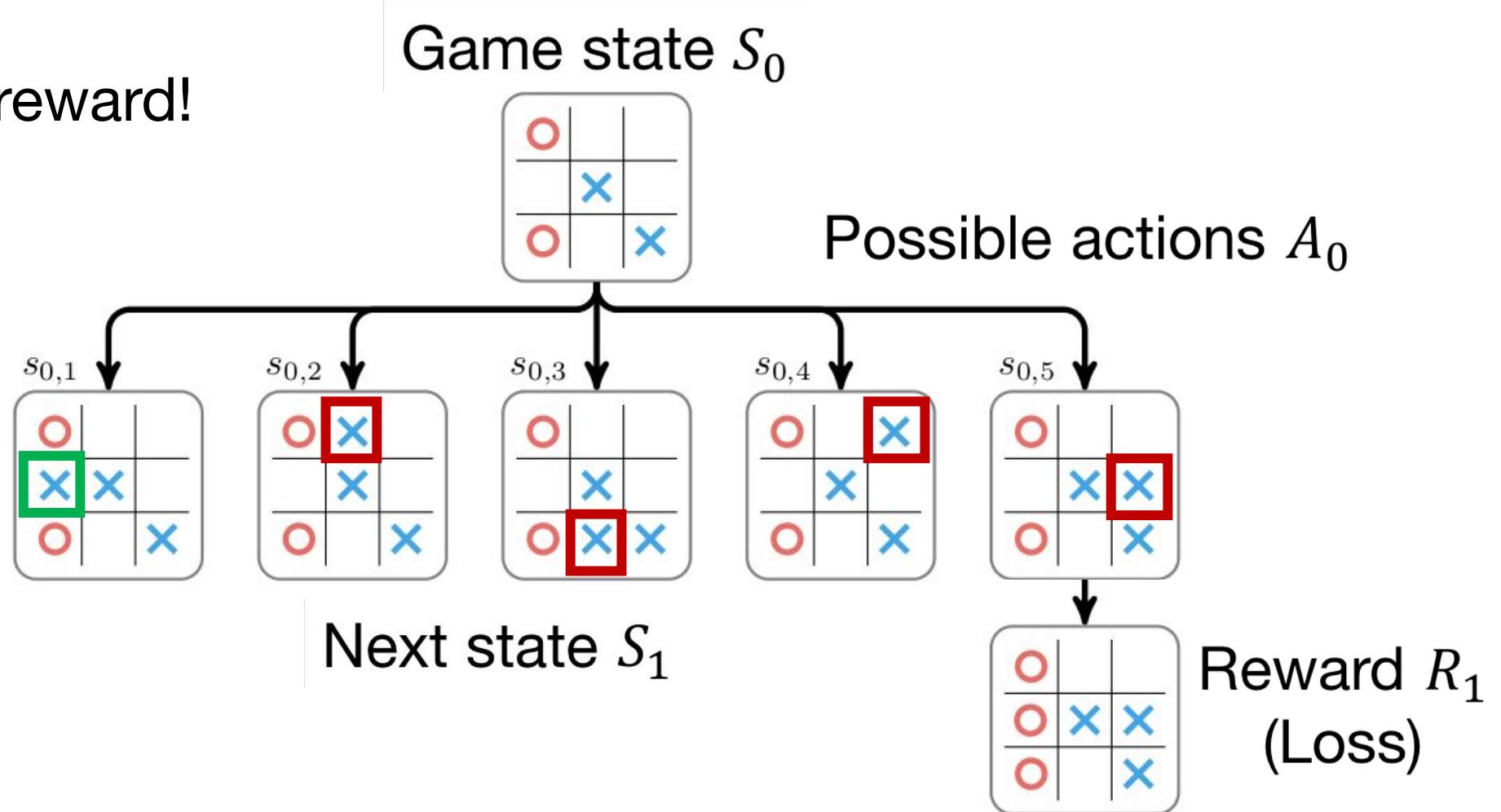
Success stories in popular press

- ▶ AlphaStar
- ▶ AlphaGo
- ▶ DQN Atari
- ▶ Open AI Five



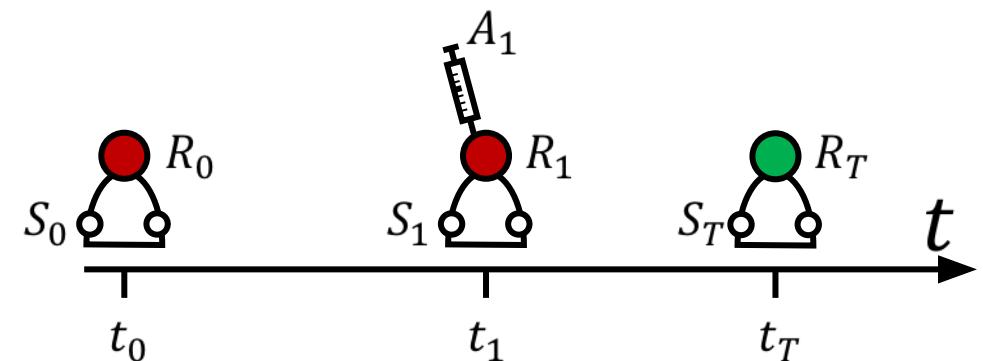
Reinforcement learning

- Maximize reward!



Great! Now let's treat patients

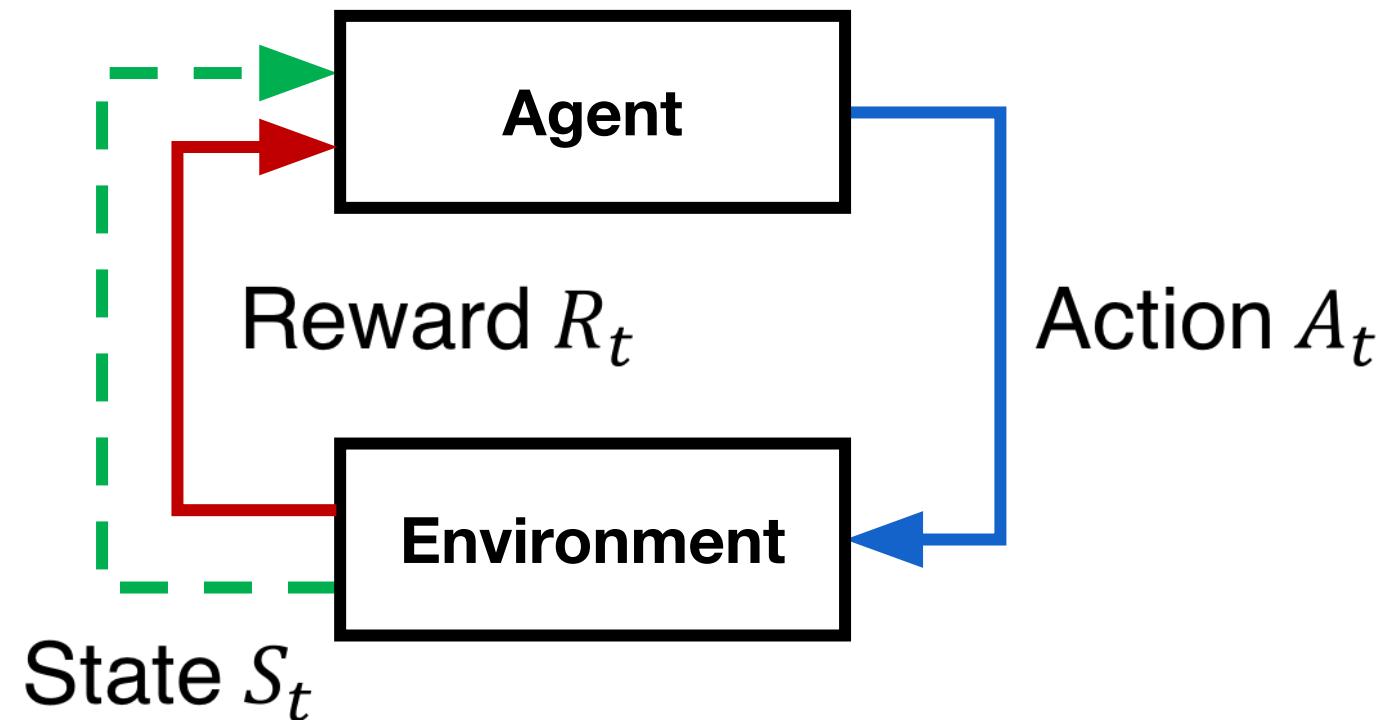
- ▶ Patient **state** at time S_t is like the game board
- ▶ Medical **treatments** A_t are like the actions
- ▶ **Outcomes** R_t are the rewards in the game
- ▶ What could **possibly** go wrong?



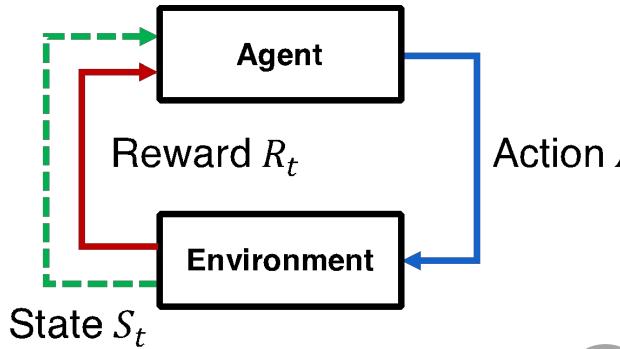
1. **Decision processes**
2. Reinforcement learning
3. Learning from batch (off-policy) data
4. Reinforcement learning in healthcare

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



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

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

Niranjan Prasad, Princeton University; Li-Fang Cheng, Princeton University; Carey Chivers, Penn Medicine; Michael Drasgolis, Penn Medicine; Barbara E. Engelhardt, Princeton University

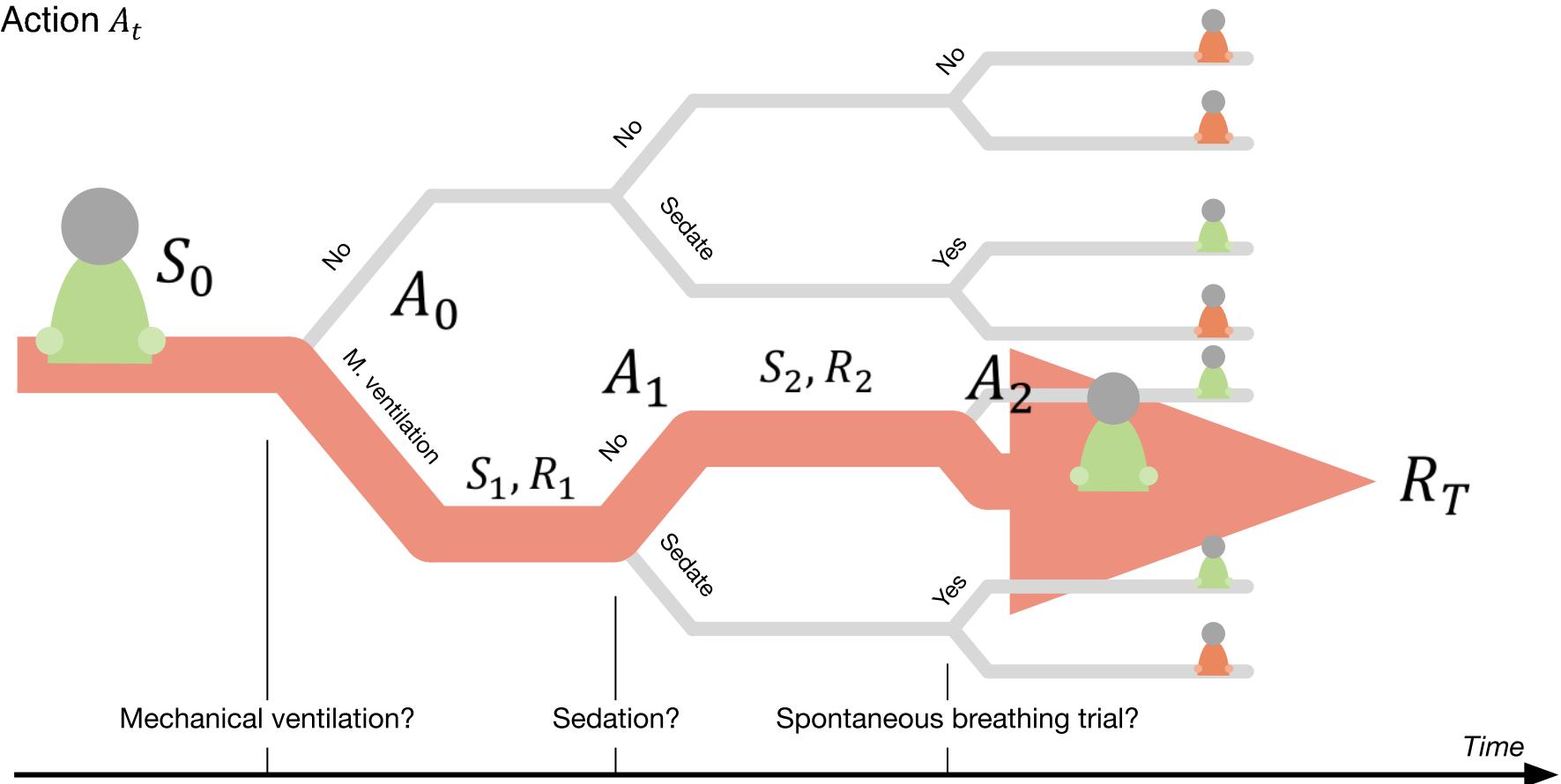
Abstract

The management of invasive mechanical ventilation, and the regulation of sedation and analgesia during ventilation, constitutes a major part of the care of patients admitted to intensive care units (ICUs). Weaning from mechanical ventilation and premature extubation are associated with increased risk of complications and hospital-acquired events, but there is no consensus best protocol for weaning patients off of a ventilator entirely. This work aims to develop a decision-making framework for weaning patients off of a ventilator, by combining information to predict time-to-extubation readiness and to recommend a personalized regime of sedation and analgesia. We propose a reinforcement learning approach to determine the best action at a given patient state. We compare treatment policies from fixed Q-tables, policy iteration, random walks, and we also demonstrate that the policies learnt show promise in recommending weaning protocols with improved outcomes, in terms of minimizing rates of re-intubation and regulating physiological stability.

1 Introduction

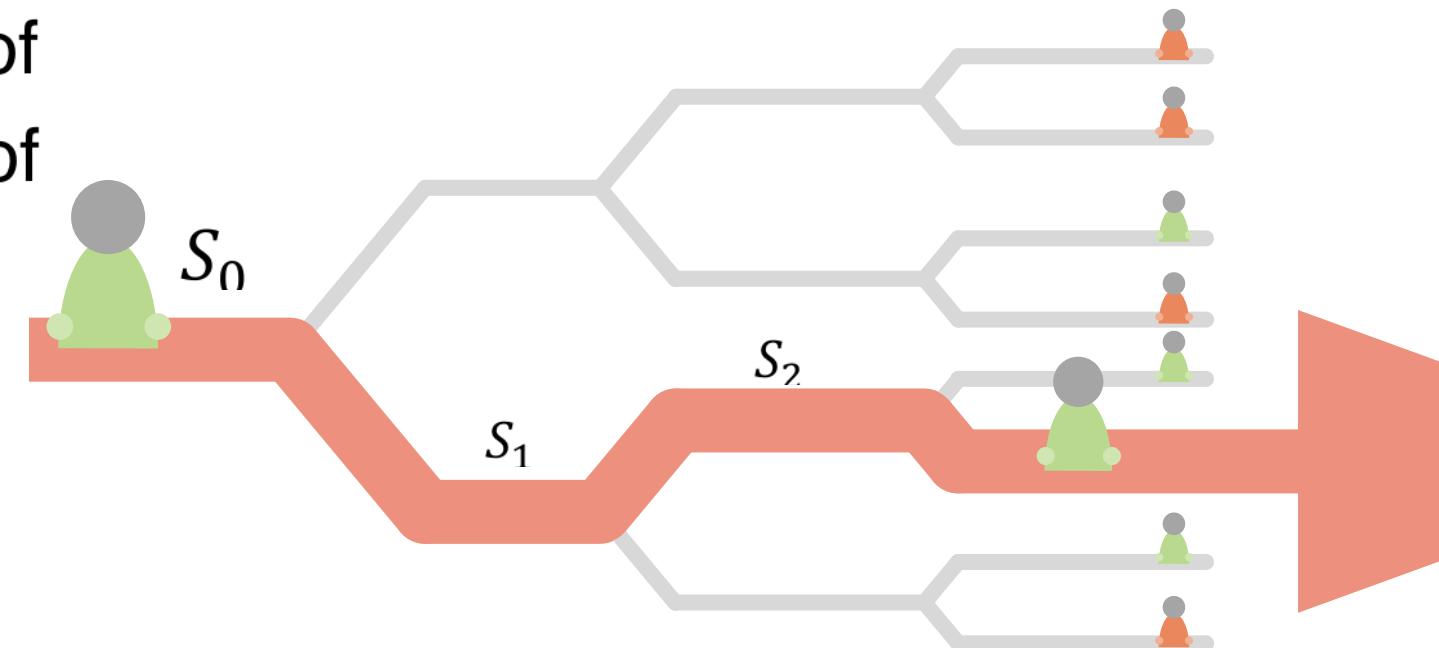
Mechanical ventilation is one of the most widely used interventions in admissions to the intensive care units (ICU); around 40% of patients in the ICU are supported on invasive mechanical ventilation at any given hour, according to the latest data from the Society of Critical Care Medicine (Ambrusso and Gabelli 2010; Wunsch et al. (2013)). These are typically patients with acute respiratory failure or compromised lung function due to conditions such as pneumonia, sepsis, or heart disease, or cases in which breathing support is necessitated by neurological disorders, impaired consciousness, or weakness.

In this work, we propose to develop a decision-support tool for weaning patients off of a ventilator in the standard ICU setting to alert clinicians when a patient is ready for initiation of weaning, and to recommend a personalized treatment protocol. We explore the use of off-policy re-



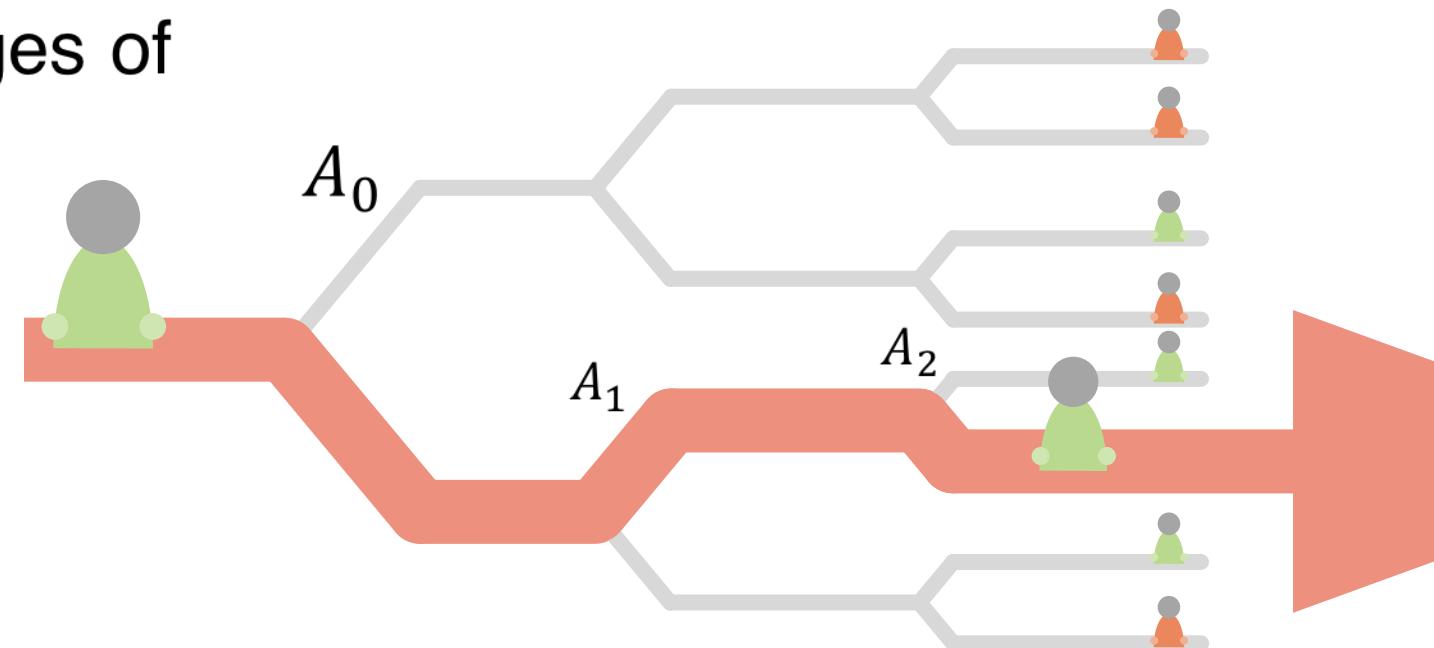
Decision process: Mechanical ventilation

- **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**: $p(S_0, \dots, S_T, A_0, \dots, A_T, R_0, \dots, R_T)$
- ▶ The agent interacts with the environment according to a **behavior policy** $\mu = p(A_t | \dots)^*$

* The ... depends on the type of agent

Markov Decision Processes

- ▶ Markov decision processes (MDPs) are a special case

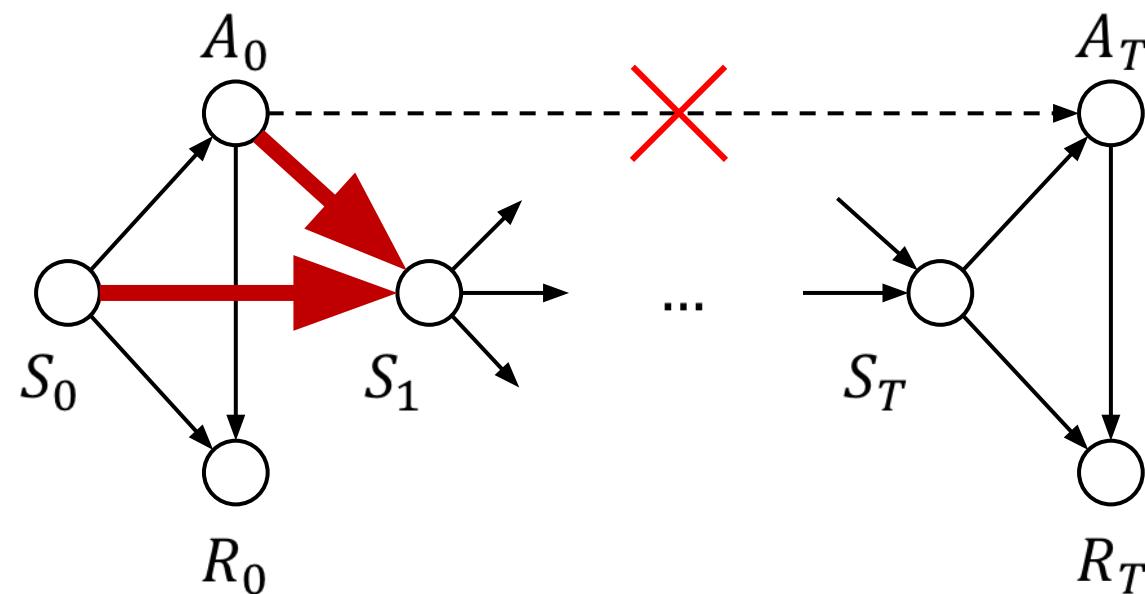
- ▶ Markov **transitions**:

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

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

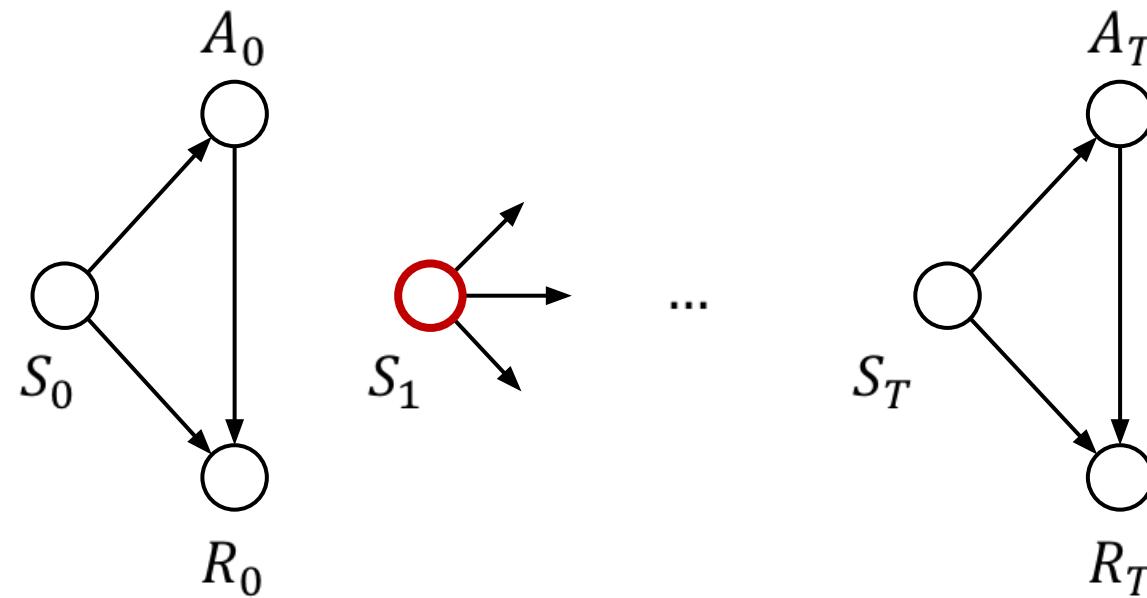
Markov assumption

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



Contextual bandits (special case)*

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



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

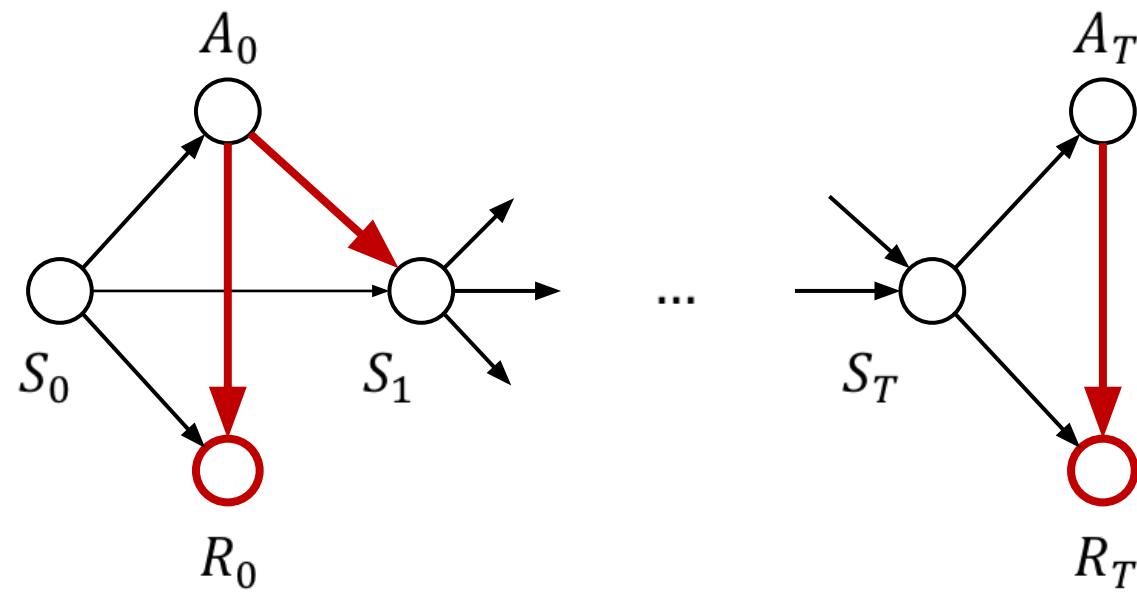
Contextual bandits & potential outcomes

- 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

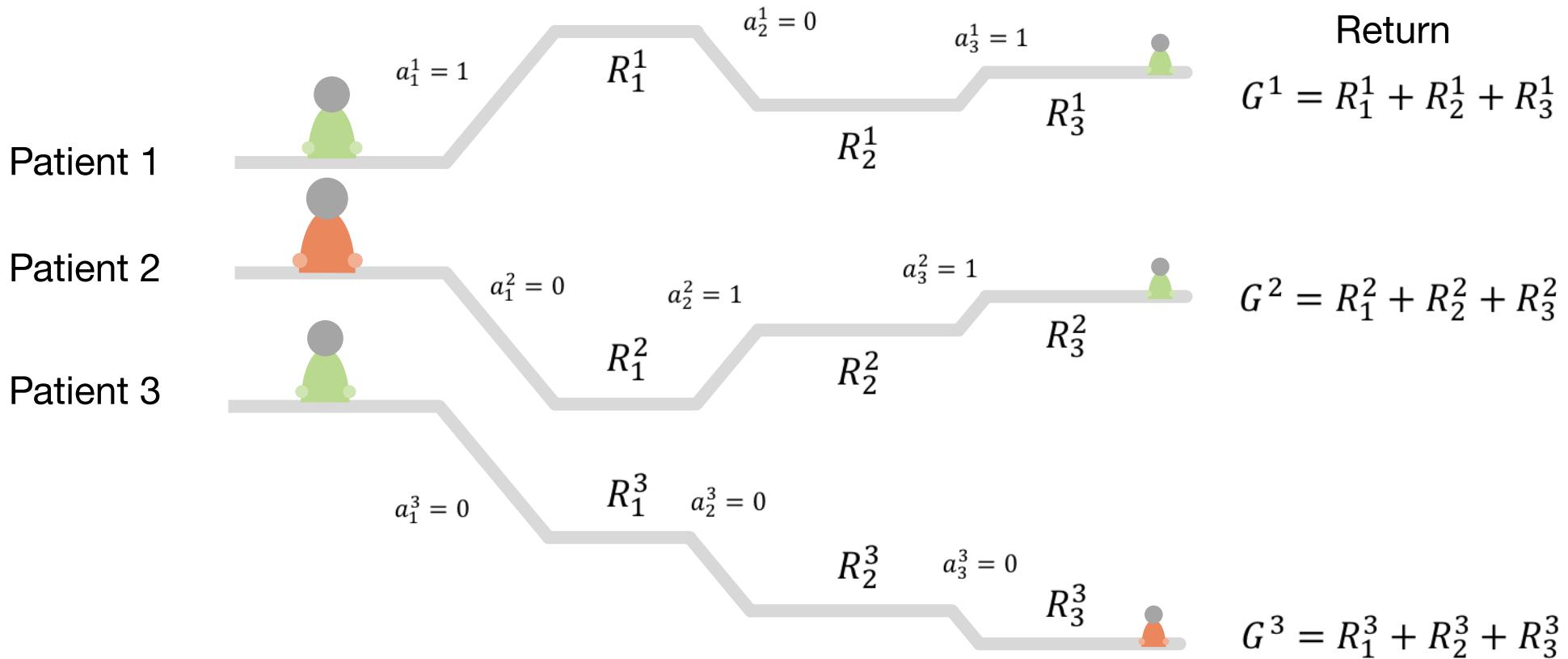


Value maximization

- ▶ 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

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



Robot in a room

- ▶ Stochastic actions



$$p(\text{Move up} \mid A = \text{"up"}) = 0.8$$

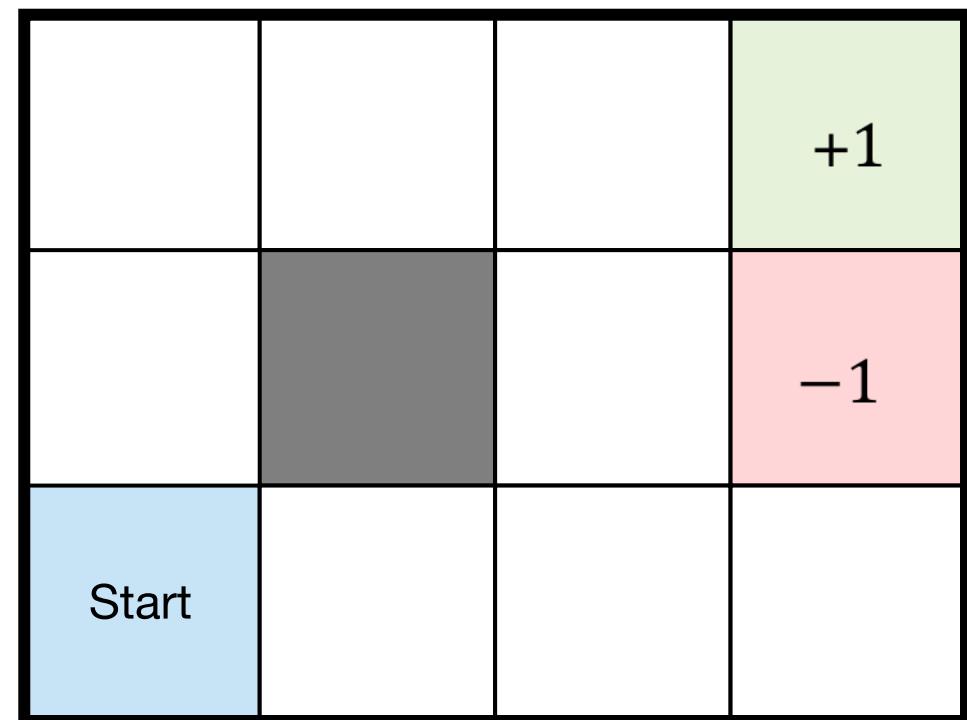
Available non-opposite moves
have uniform probability

- ▶ Rewards:

+1 at [4,3] (terminal state)

-1 at [4,2] (terminal)

-0.04 per step



Robot in a room

- ▶ Stochastic actions

$$p(\text{Move up} \mid A = \text{"up"}) = 0.8$$

Available non-opposite moves
have uniform probability

- ▶ Rewards:

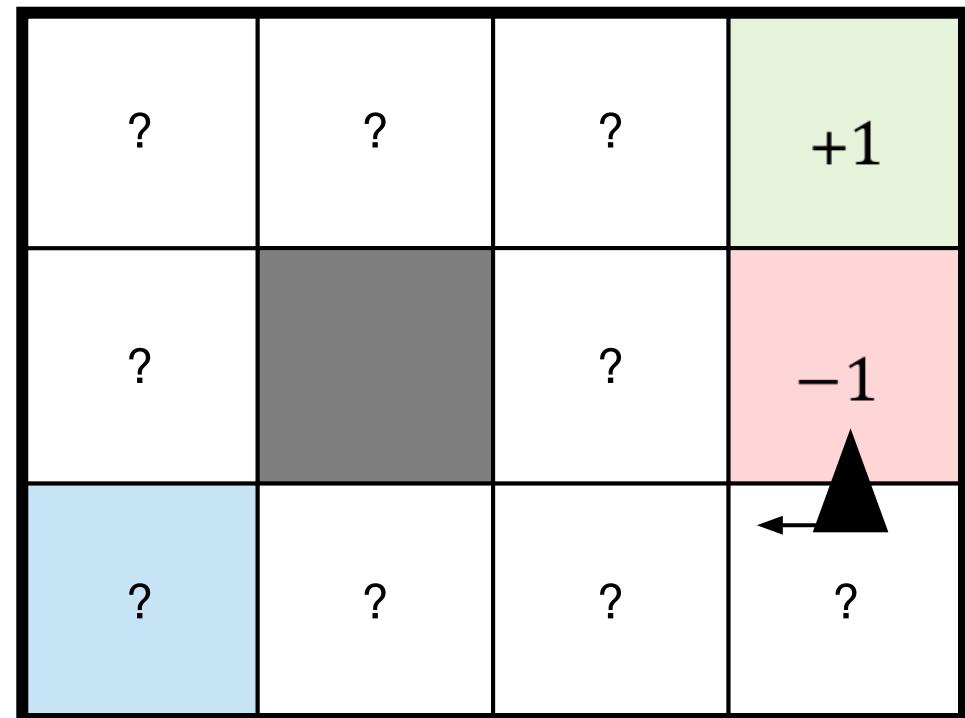
+1 at [4,3] (terminal state)

-1 at [4,2] (terminal)

-0.04 per step

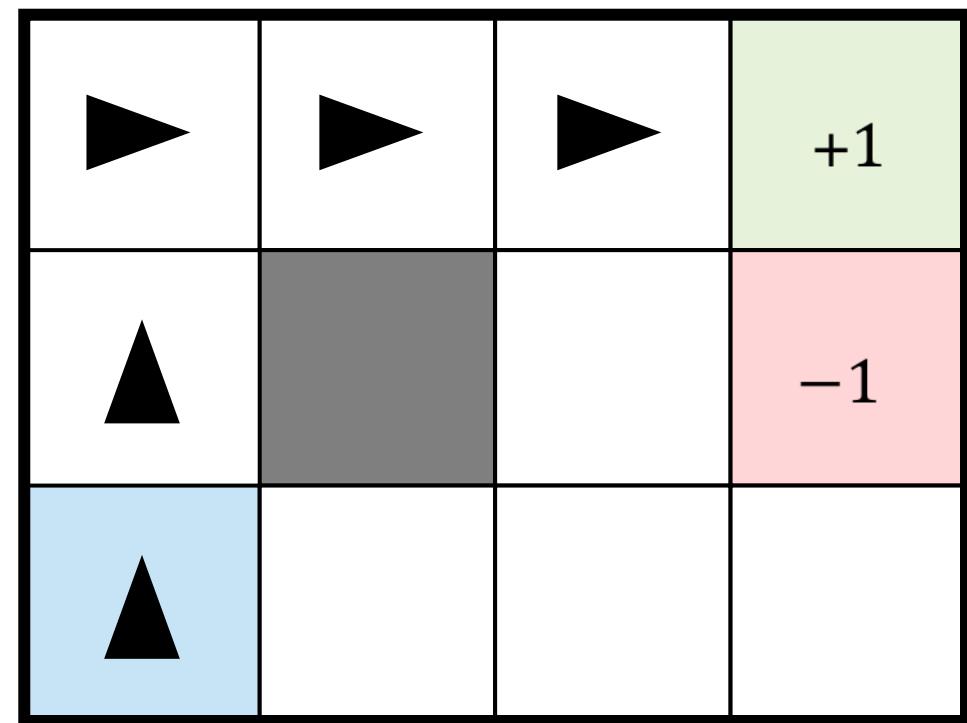


What is the optimal policy?



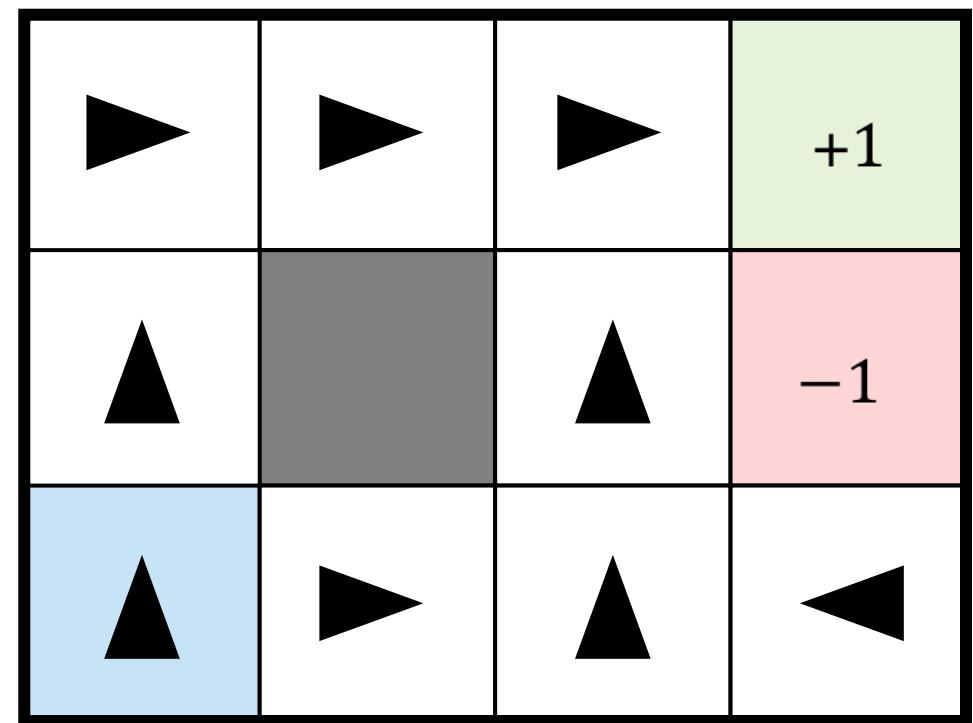
Robot in a room

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



Robot in a room

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



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

Paradigms*

► Model-based RL

Transitions

$$p(S_t | S_{t-1}, A_{t-1})$$

G-computation

MDP estimation

Value-based RL

Value/return

$$p(G_t | S_t, A_t)$$

Q-learning

G-estimation

Policy-based RL

Policy

$$p(A_t | S_t)$$

REINFORCE

Marginal structural models

*We focus on off-policy RL here

Paradigms*

► Model-based RL

Transitions

$$p(S_t | S_{t-1}, A_{t-1})$$

G-computation

MDP estimation

Value-based RL

Value/return

$$p(G_t | S_t, A_t)$$

Q-learning

G-estimation

Policy-based RL

Policy

$$p(A_t | S_t)$$

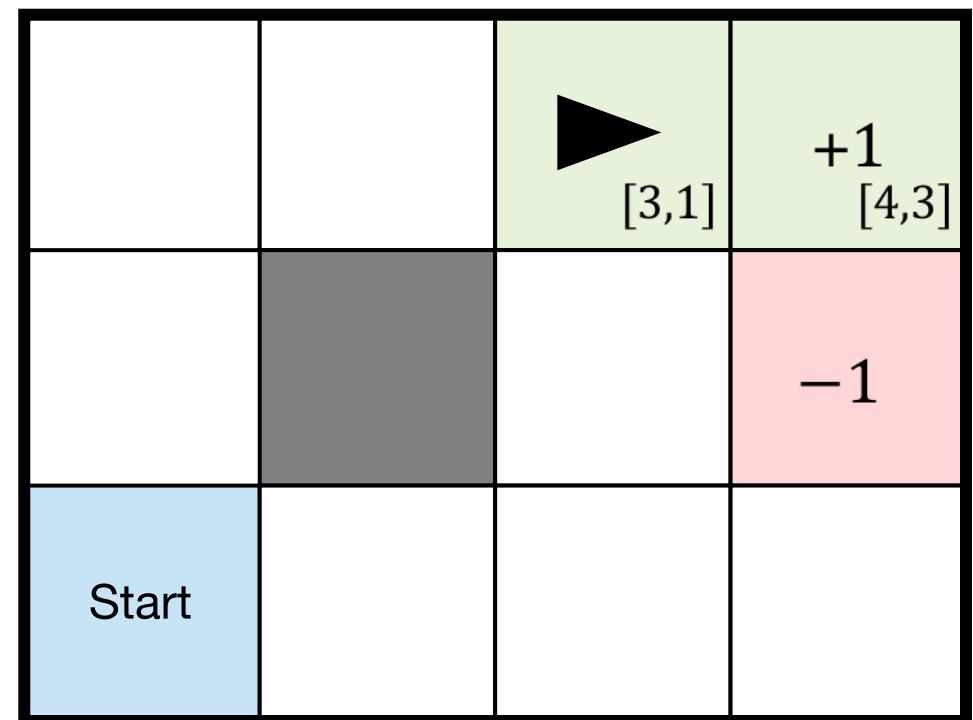
REINFORCE

Marginal structural models

*We focus on off-policy RL here

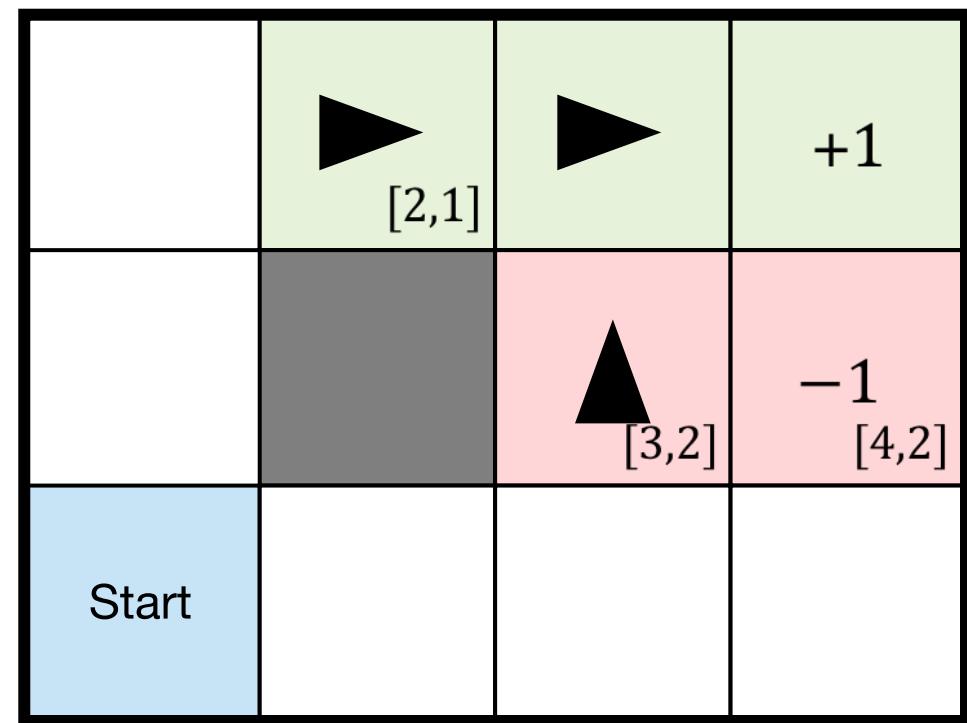
Dynamic programming

- ▶ 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]



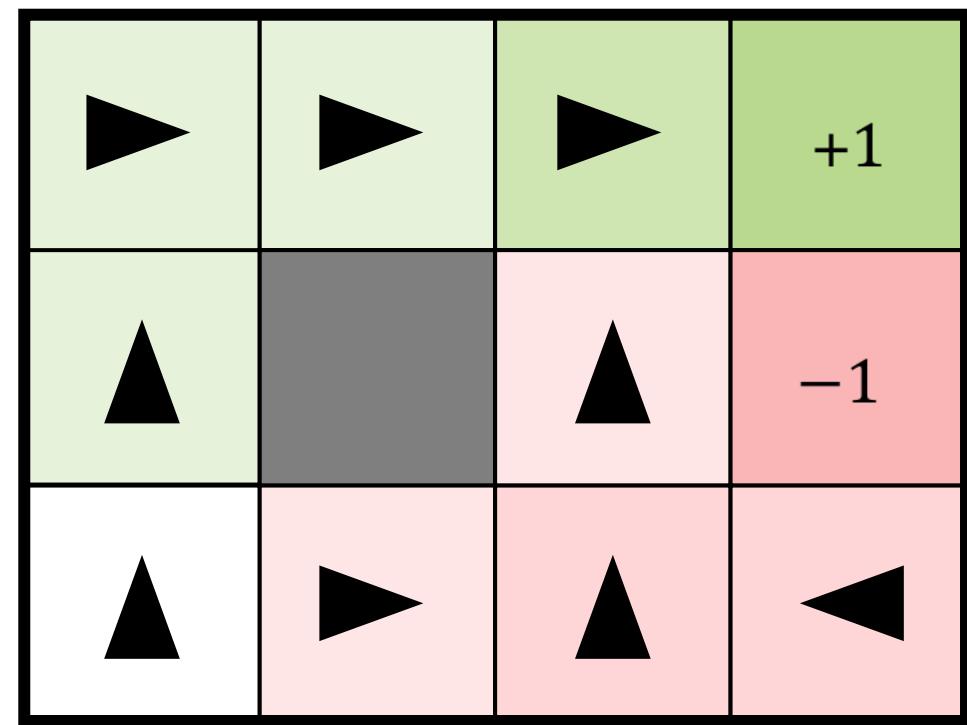
Dynamic programming

- ▶ [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]?



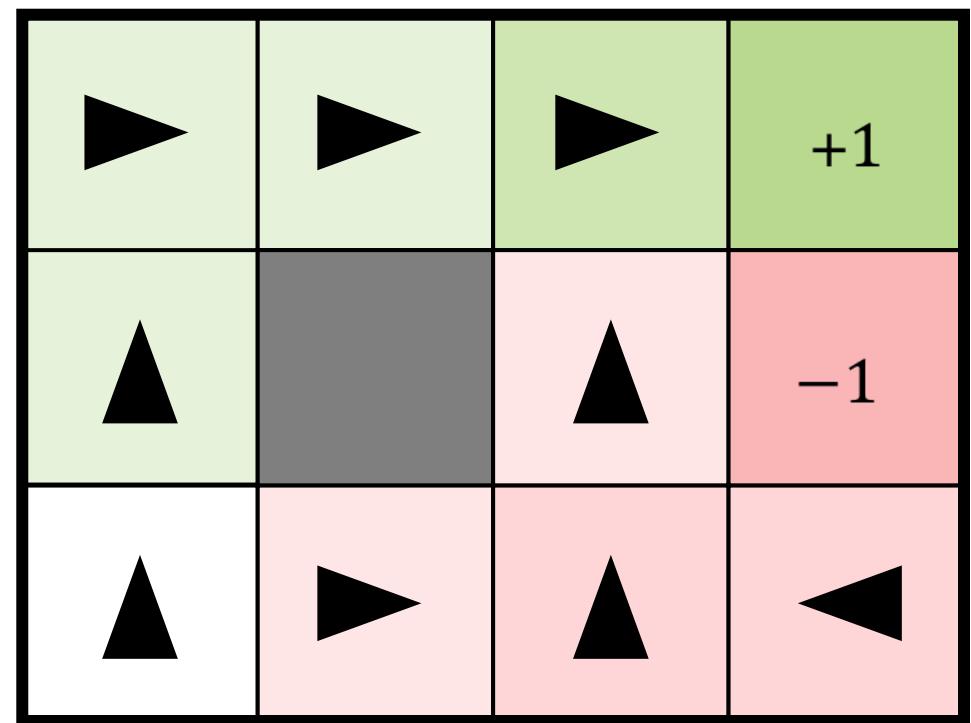
Dynamic programming

- ▶ 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



Dynamic programming

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



Q-learning

- Q-learning is a value-based reinforcement learning method
- **Recall:** The value of a state s under a policy π is

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

|
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

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

Reward discount factor*

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

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

*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

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

- 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**, $s \in \{0, \dots, K\}$, 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

- ▶ 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			
-0.04	-0.04	-0.04	0.96
-0.04		-0.04	-1.04
-0.04	-0.04	-0.04	-1.04

* We will come back to this

Q-learning with discrete states

- ▶ 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

-0.08	-0.08	0.92	-0.08	0.96	+1
-0.08	0.92	-1.04	-0.08	-1.04	-1
-0.08	-0.08	-0.08	-0.08	-0.08	-1.04
-0.08	-0.08	-0.08	-0.08	-0.08	-0.08

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

0.88	-0.12	0.92	0.88	0.96	+1
-0.12	0.88	0.92	-1.04	-1	
-0.12		-0.08	-1.04		
-0.12	-0.12	-0.12	-0.12	-0.12	-1.04

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

	0.88	0.92	0.88	0.96	+1
-0.16	0.84	0.92	0.88	-1.04	-1
0.84	-0.16	0.84	-1.04	0.84	
-0.16	-0.16	-0.16	0.84	-0.16	0.84

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

0.80	0.88	0.84	0.92	0.88	0.96	+1
0.84				0.92	-1.04	-1
-0.18				0.84		
0.80	0.80	-0.18	0.84	0.80	0.80	-1.04

Q-learning with discrete states

- ▶ 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

0.88	0.84	0.92	0.88	0.96	+1
0.80	0.84	0.92	0.88	-1.04	-1
0.84	0.76	0.84	0.84	0.84	-1.04
0.80	0.80	0.76	0.84	0.80	0.84

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

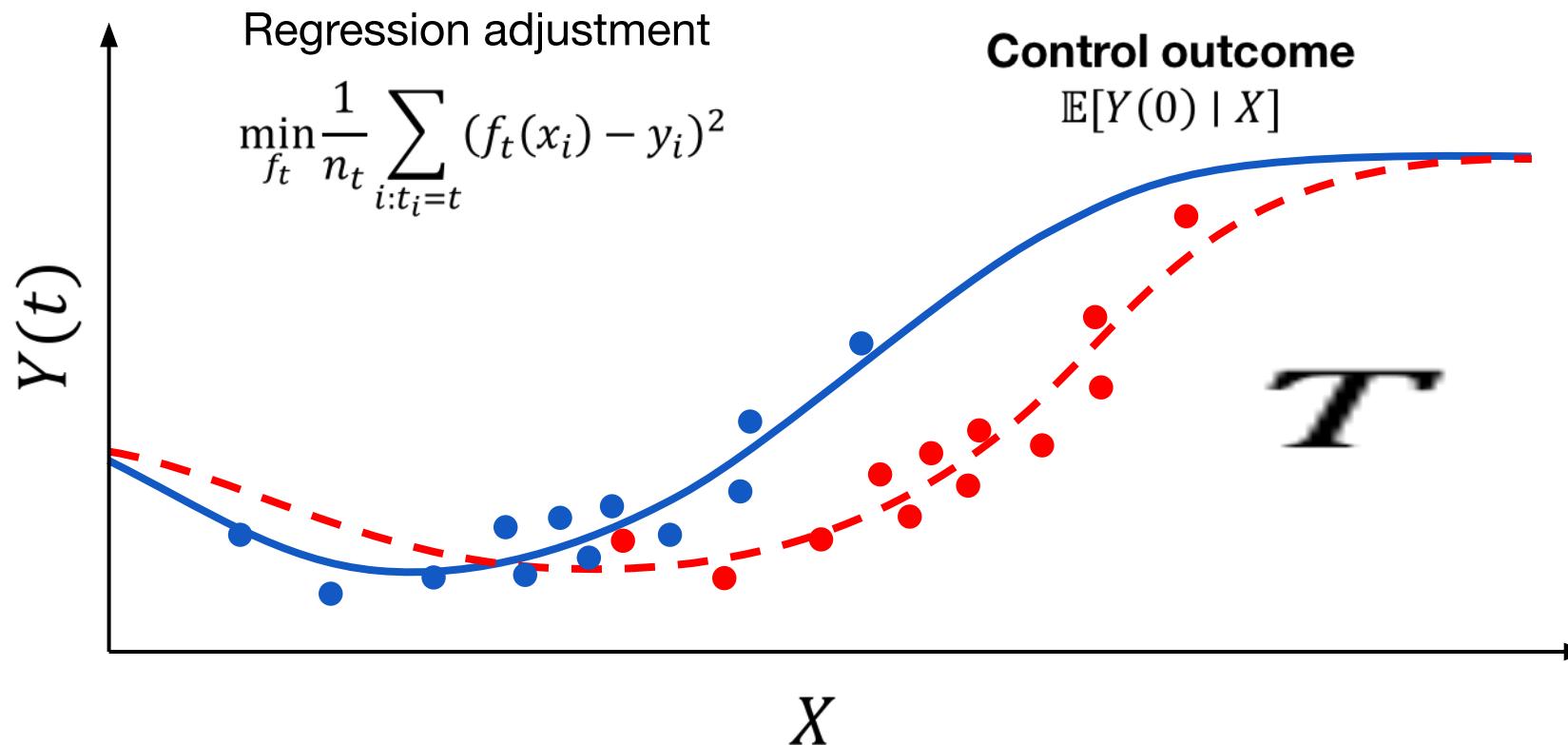
Bellman equation (one step)

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

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

- Finding $q(s, a)$ is analogous to finding expected potential outcomes $\mathbb{E}[R(a) | S = s]$ 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)

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

Choice of loss, (here squared)

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

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

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 $(s_1, a_1, r_1), \dots, (s_T, a_T, r_T)$, 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

Strong ignorability:

$$Y(0), Y(1) \perp\!\!\!\perp T \mid X$$

“No *hidden* confounders”

Sequential case

Sequential randomization:

$$G(\dots) \perp\!\!\!\perp A_t \mid \bar{S}_t, \bar{A}_{t-1}$$

“Reward indep. of policy given history”

Overlap:

$$\forall x, t: p(T = t \mid X = x) > 0$$

“All actions possible”

Positivity:

$$\forall a, t: p(A_t = a \mid \bar{S}_t, \bar{A}_{t-1}) > 0$$

“All actions possible at all times”

Assumptions for (off-policy) RL

- Sufficient conditions for identifying value function

Single-step case

Strong ignorability:

$$Y(0), Y(1) \perp\!\!\!\perp T \mid X$$

“No *hidden* confounders”

Overlap:

$$\forall x, t: p(T = t \mid X = x) > 0$$

“All actions possible”

Sequential case

Sequential randomization:

$$G(\dots) \perp\!\!\!\perp A_t \mid \bar{S}_t, \bar{A}_{t-1}$$

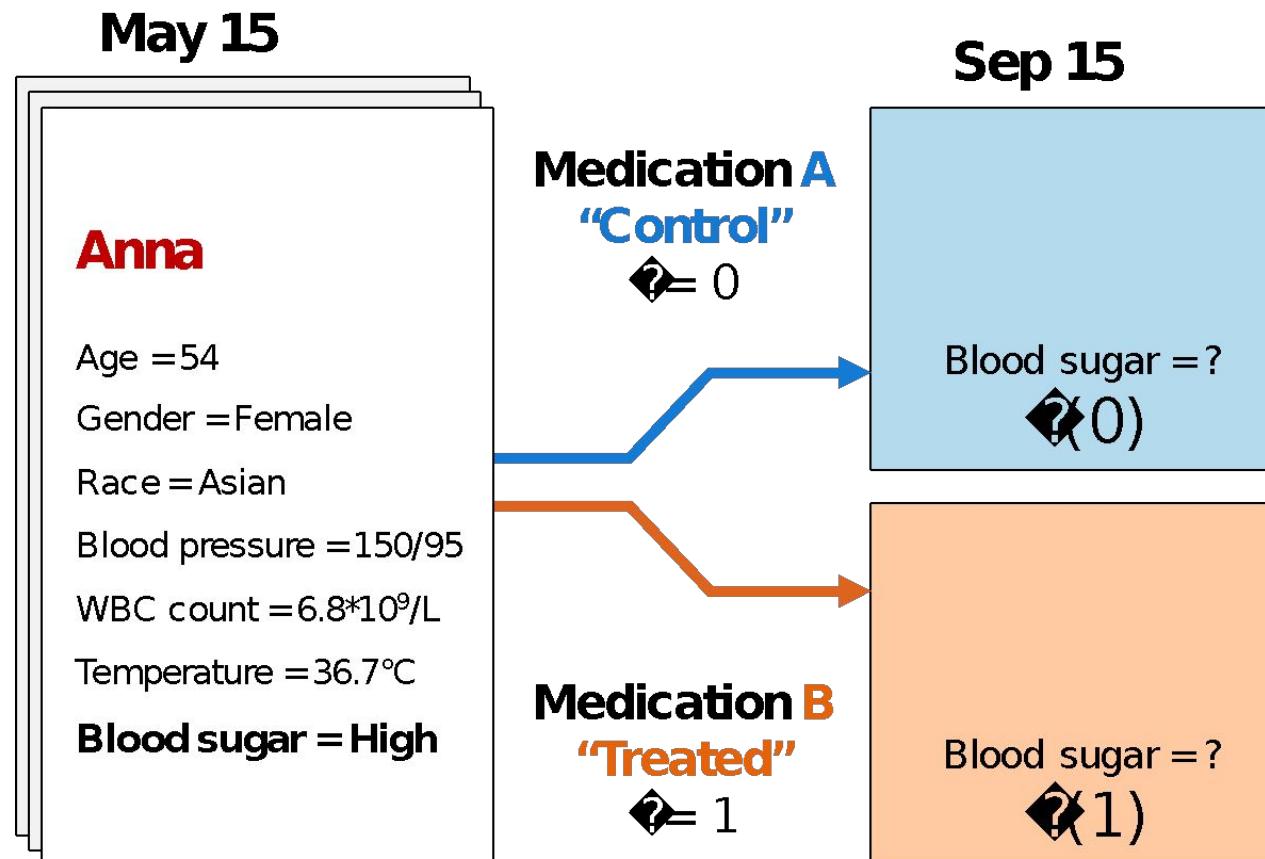
“Reward indep. of policy given history”

Positivity:

$$\forall a, t: p(A_t = a \mid \bar{S}_t, \bar{A}_{t-1}) > 0$$

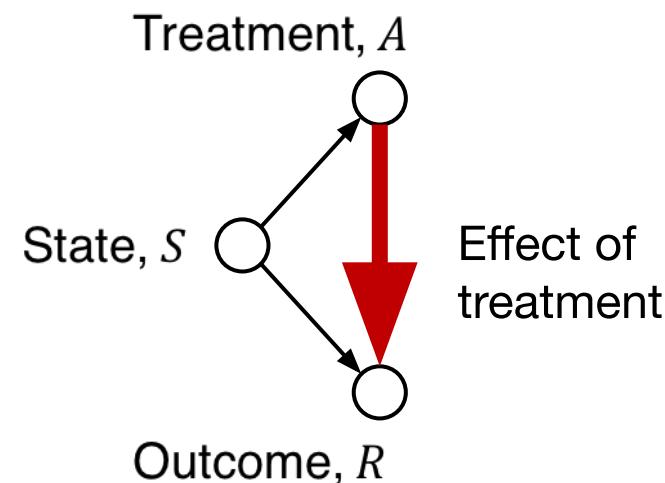
“All actions possible at all times”

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



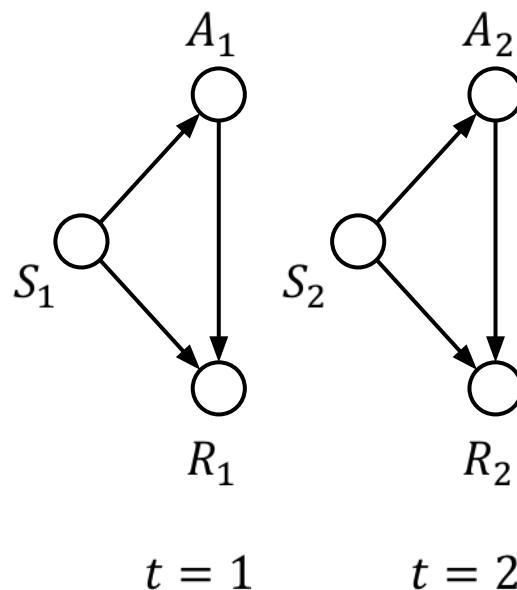
Ignorability

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

Potential outcome under
action a

Treating Anna over time

- ▶ Let's add a time point...

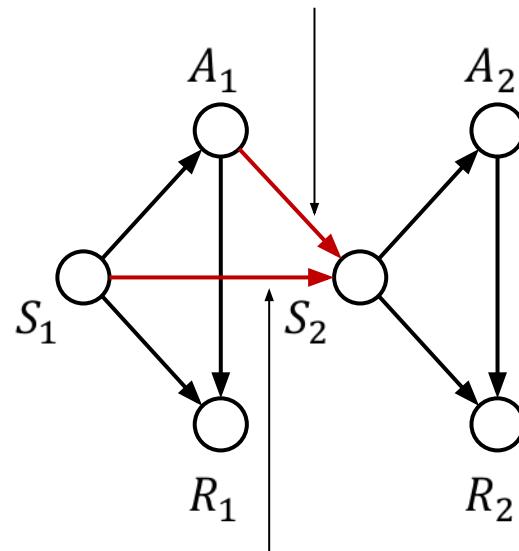


Ignorability
 $R_t(a) \perp\!\!\!\perp A_t \mid S_t$

Treating Anna over time

- ▶ What influences her state?

Anna's health status depends on how we treated her



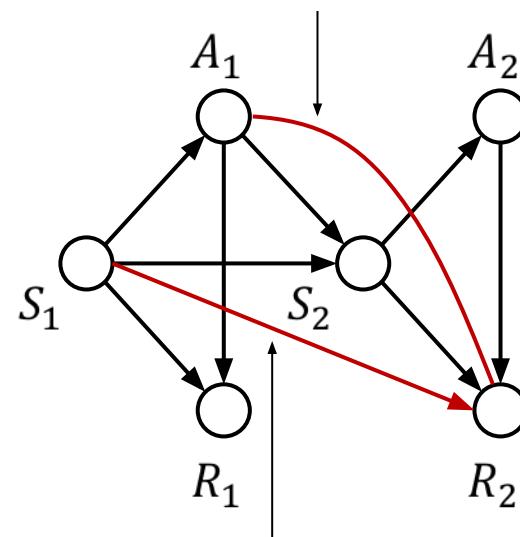
Ignorability
 $R_t(a) \perp\!\!\!\perp A_t \mid S_t$

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

Treating Anna over time

- ▶ What influences her state?

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

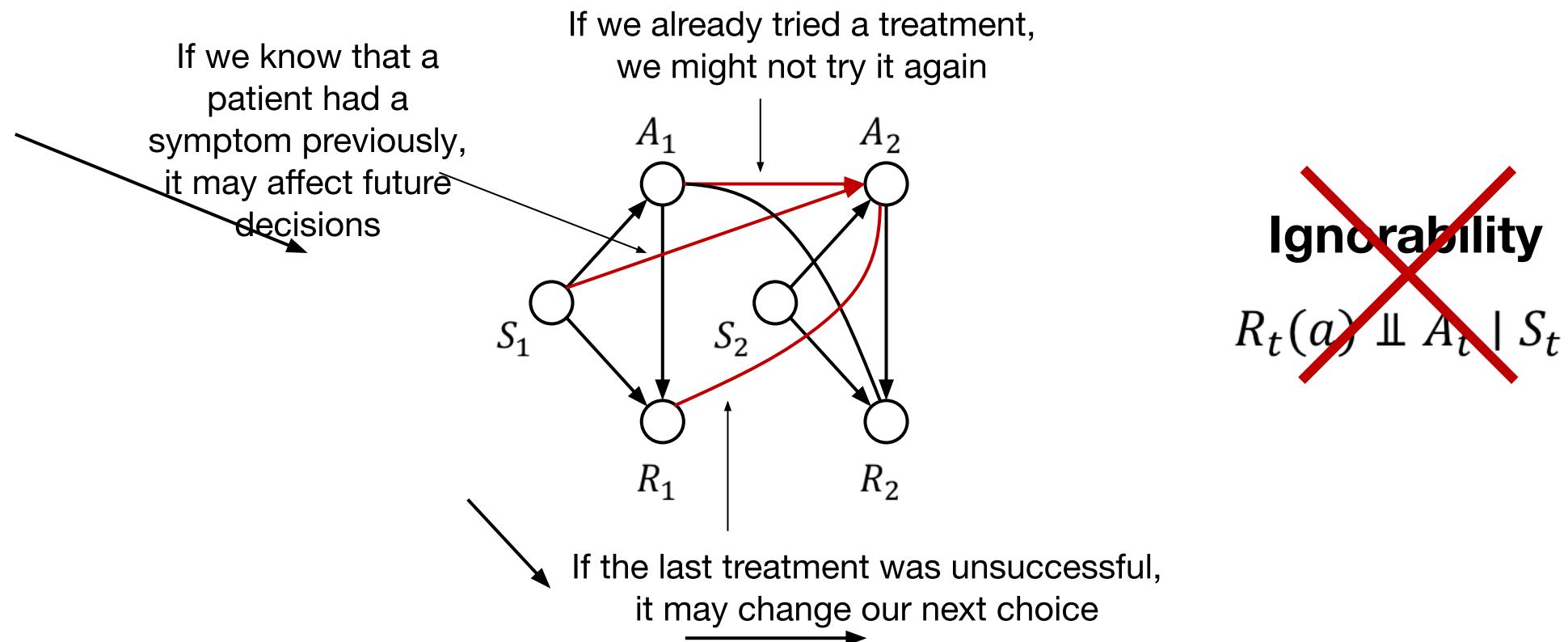


Ignorability
 $R_t(a) \perp\!\!\!\perp A_t \mid S_t$

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

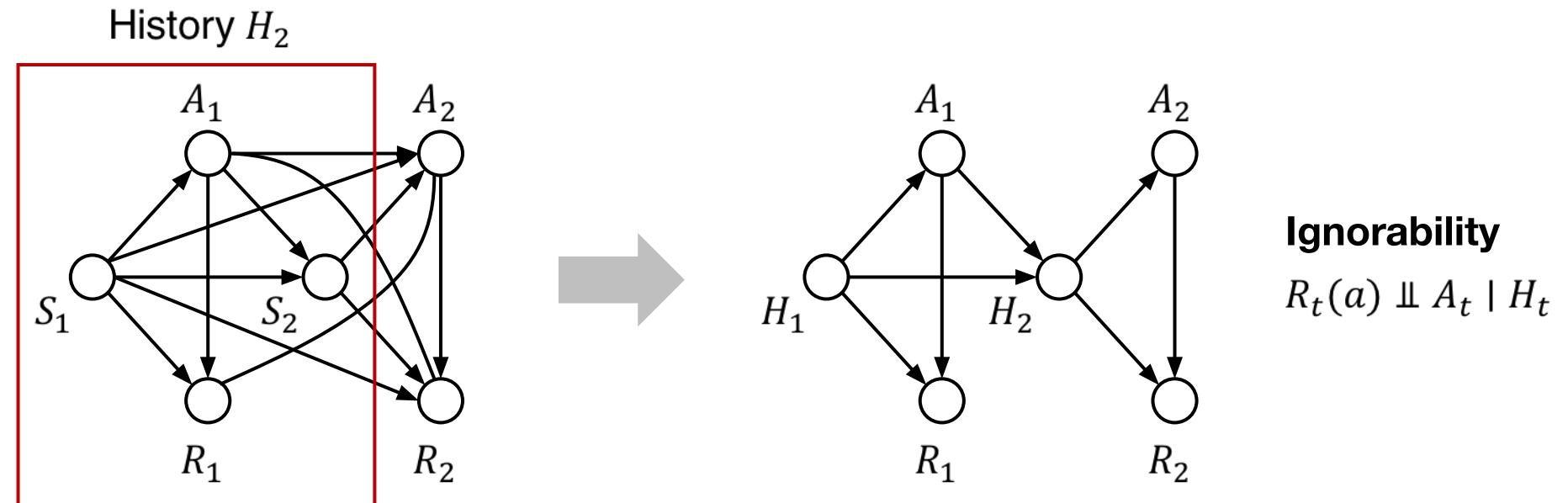
Treating Anna over time

- ▶ What influences her state?



State & ignorability

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



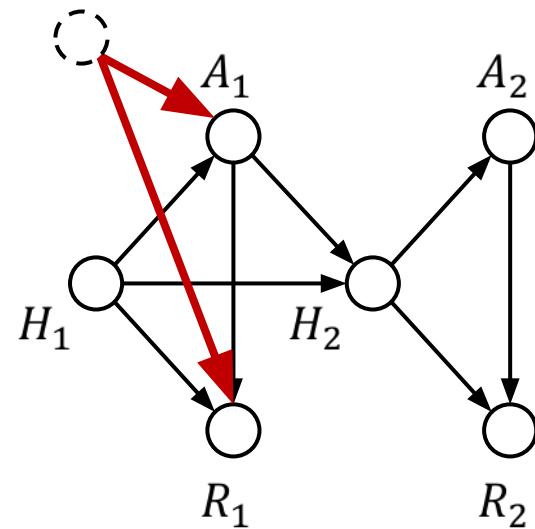
Summarizing history

- ▶ 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

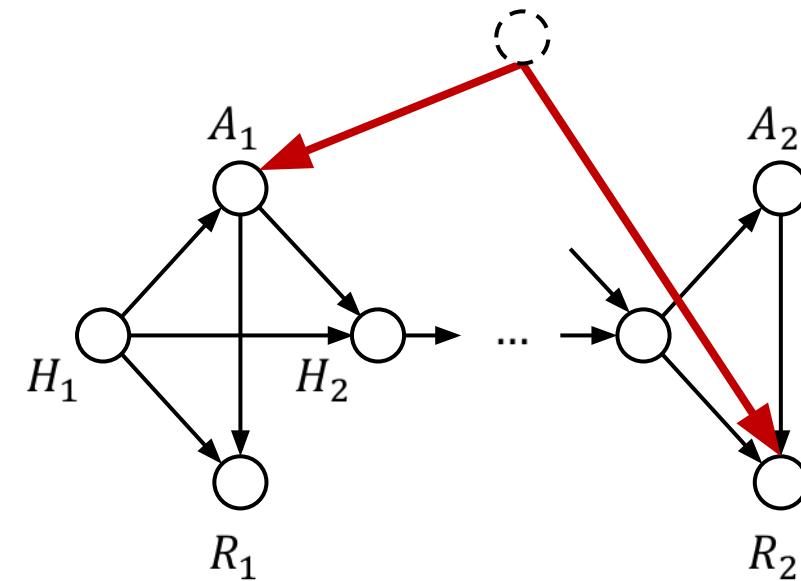
State & ignorability

- We cannot leave out unobserved confounders

Unobserved confounder, U



Unobserved confounder, U



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)