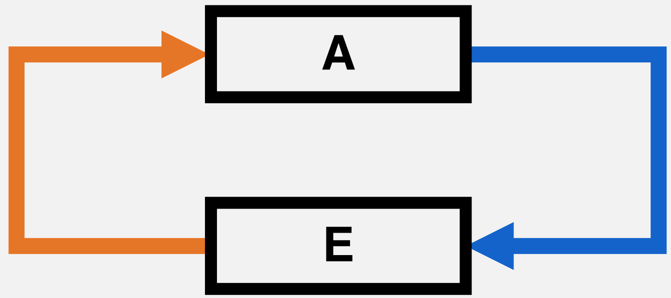


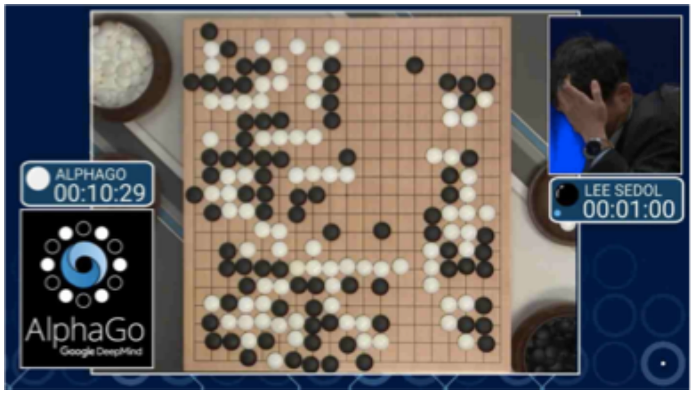
Causal Aspects of Deep Reinforcement Learning

Causal Inference & Deep Learning

MIT IAP 2018

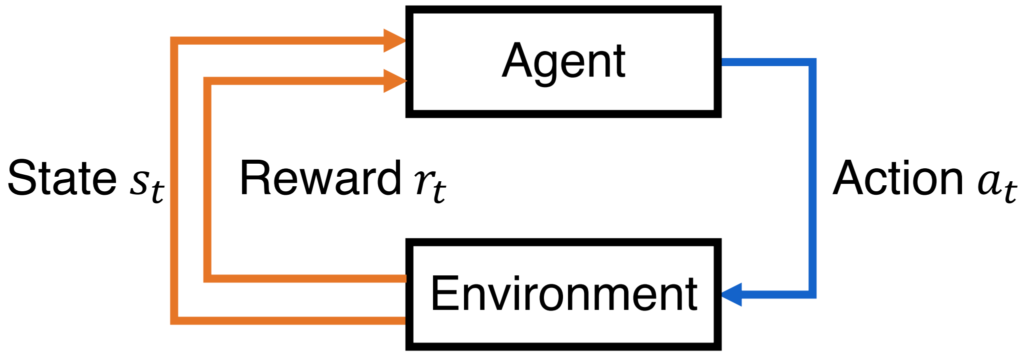
Fredrik D. Johansson

Deep reinforcement learning



Reinforcement learning in general

- Often illustrated as a loop over time $t = 0, 1, 2, \ldots$



- Continuously, the agent updates its belief of the world based on the feedback from the environment

- Learning through trial and error

Maximizing reward

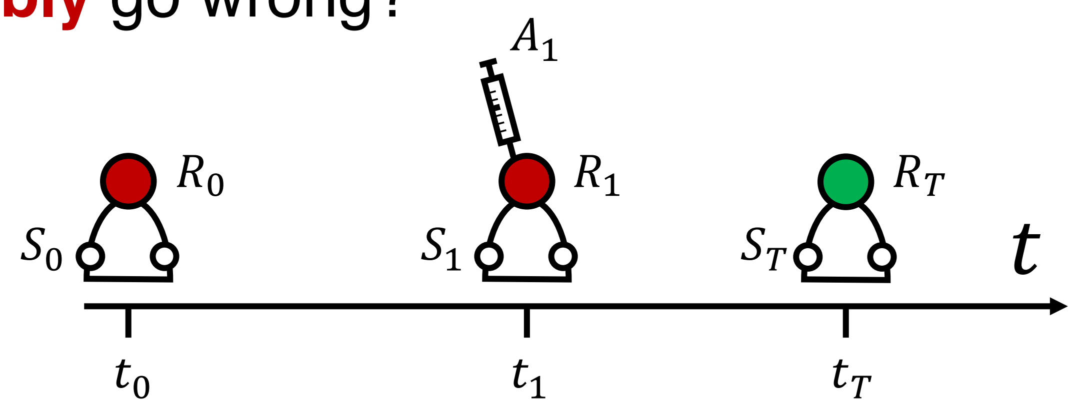
- The goal of most RL algorithms is to maximize the value, or expected return

- **Return**: $R = \sum_{t=1}^{T} r_t$ (sometimes infinite, discounted sum)
- **Value:** $V_\pi = \mathbb{E}[R]$ (sometimes conditioned on starting state)

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

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**/progression R_t are the rewards in the game
- What could **possibly** go wrong?



1. **Decision processes**

2. Learning from batch (off-policy) data

3. Reinforcement learning paradigms

4. Applications

Decision processes

- The environment-agent system is called a **decision process**
- The 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 policy $p(A_t \mid \cdots)$. (The ... depends on the type of agent)

Markov decision processes

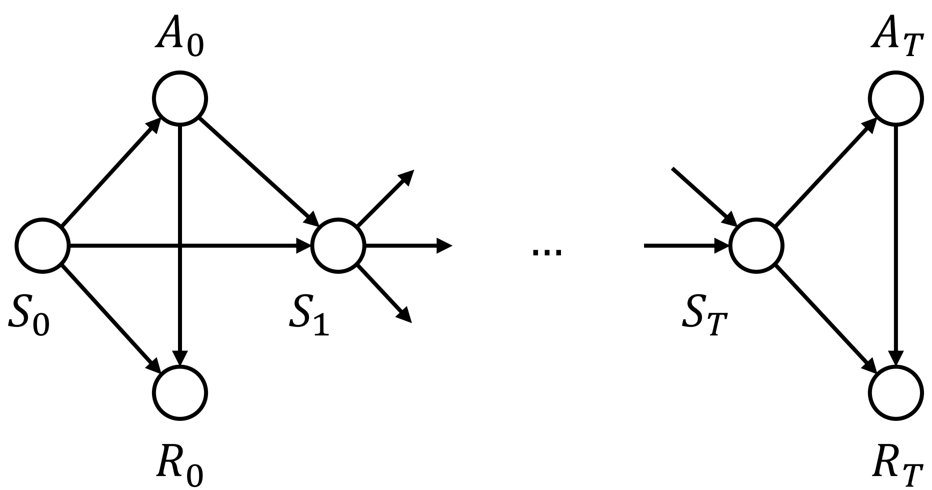
- Markov decision processes (MDPs) are a special case
- (Unknown) Markov **transitions**:

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

- (Unknown) Markov **reward** function: $p(R_t \mid S_t, A_t)$
- Markov **action** policy $p(A_t \mid S_t)$, (often denoted π or μ)

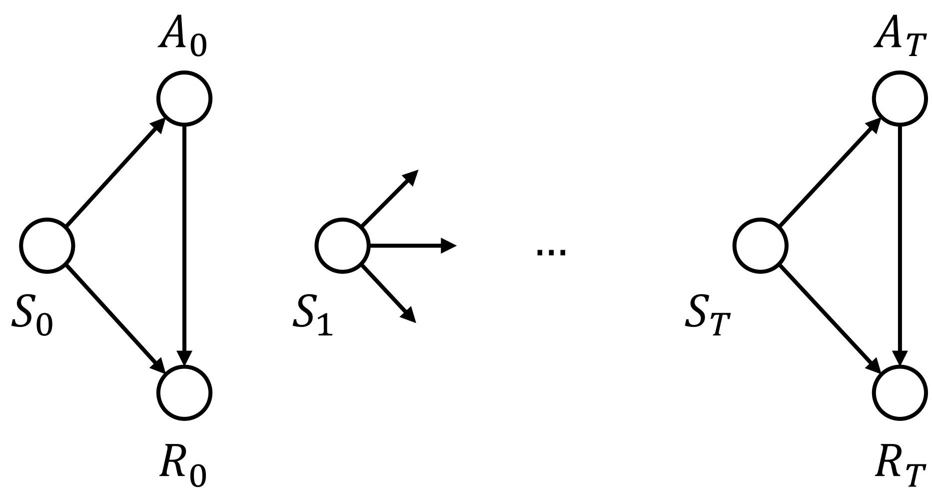
Markov assumption & MDPs

- State transitions depend only on most recent state-action
- Most common model in deep RL

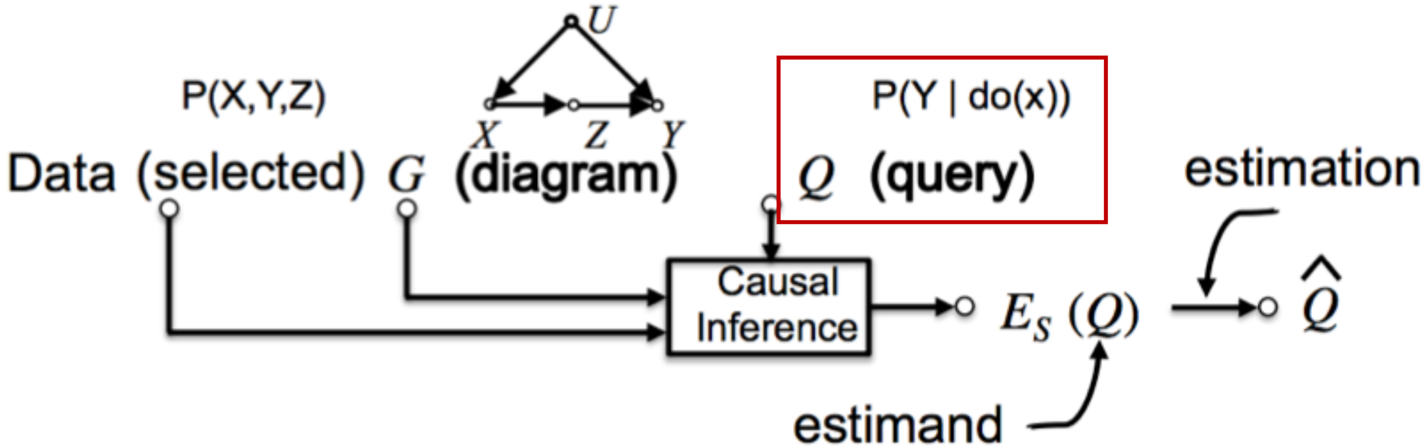


Contextual bandits

- States are independent: $p(S_t \mid S_{t-1}, A_{t-1}) = p(S_t)$
- Equivalent to single-step case, but focus often on exploration



What question are we asking?



What question are we asking?

- **Goals:**
  - What is a policy that maximizes expected reward?
  - What is the expected reward of a fixed policy π
- **Settings:**
  - **On-policy:** If I can try out my new policy π in practice, how do I find the best one quickly?
  - **Off-policy:** If I can't try out a policy π, how do I find a good one and evaluate it using observational (off-policy) data?

What question are we asking?

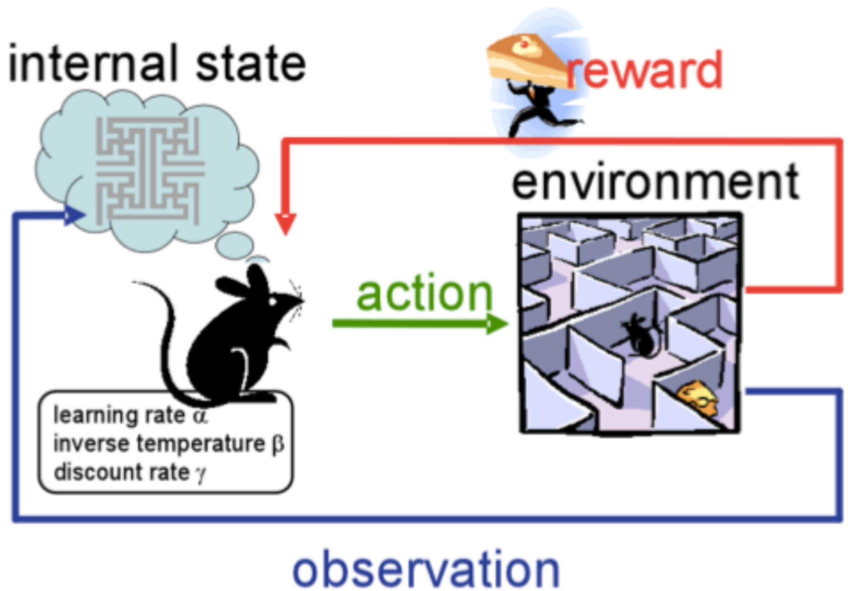
- **Goals:**
  - What is a policy that maximizes expected reward?
  - What is the expected reward of a fixed policy π

- **Settings:**
  - **On-policy:** If I can try out my new policy π in practice, how do I find the best one quickly?

Focus today

  - **Off-policy:** If I can't try out a policy π, how do I find a good one and evaluate it using observational (off-policy) data?

What question are we asking?



On-policy:
We are the rat.

Off-policy
We are learning from a video of the rat in the maze.

1. Decision processes

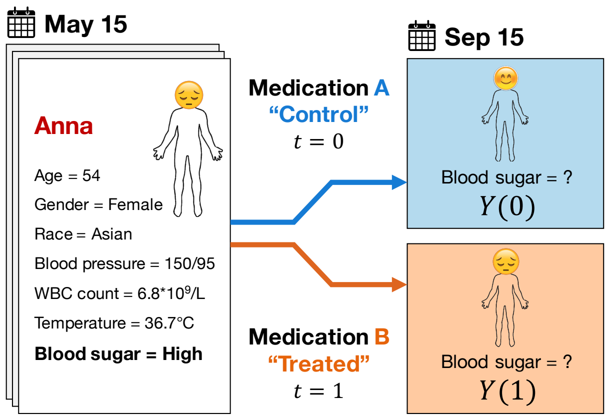
2. Learning from batch (off-policy) data

3. Reinforcement learning paradigms

4. Applications

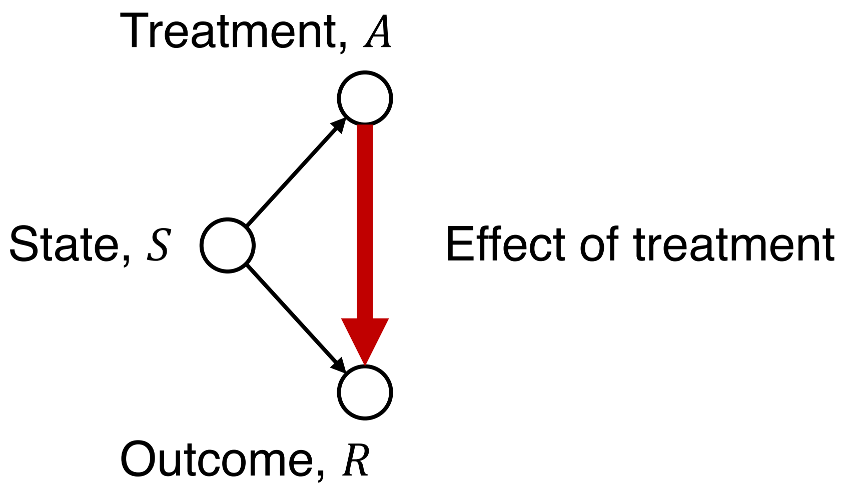
Treating Anna over time

- Remember our diabetic patient
- We had observed hers and other patient's electronic health records over time
- Based on this information, **without experimenting** further, what would be the best treatment for Anna?



Treating Anna over time

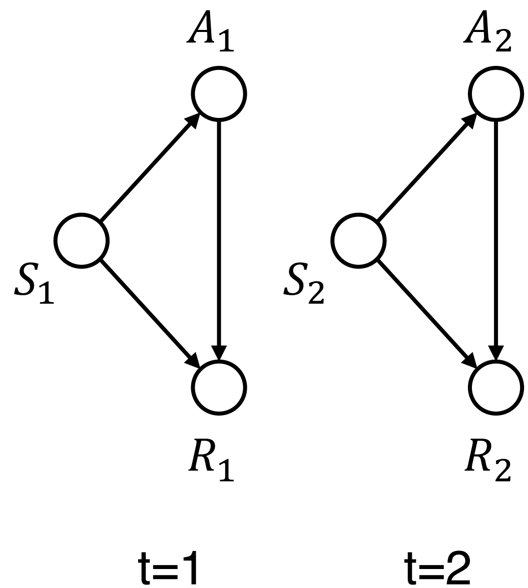
- We assumed a simple causal graph. This let us identify the causal effect of treatment on outcome from observational data



Equivalent to a single time step MDP!

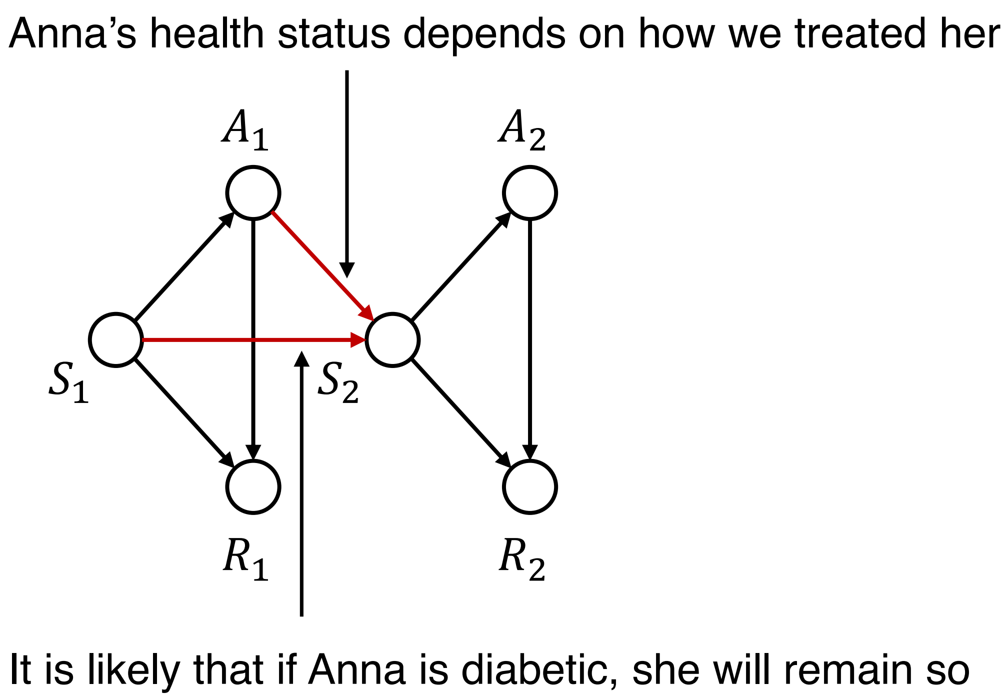
Treating Anna over time

- Let's add a time point:



Treating Anna over time

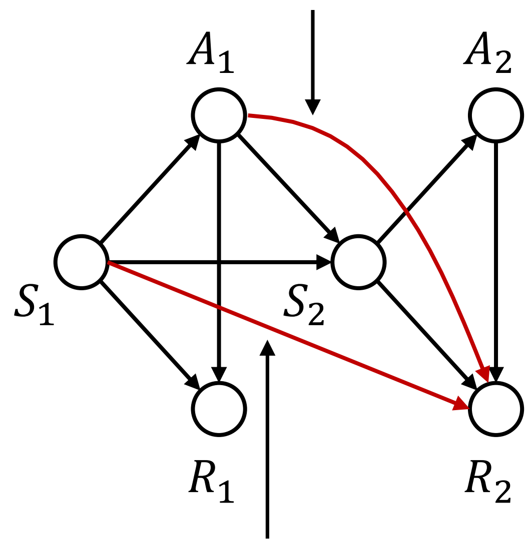
- Let's add a time point:



Treating Anna over time

- Let's add a time point:

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



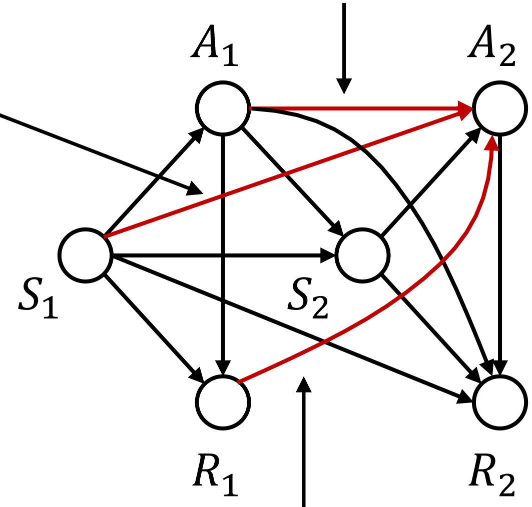
The outcome at a later time may depend on an earlier state

Treating Anna over time

- Let’s add a time point:

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

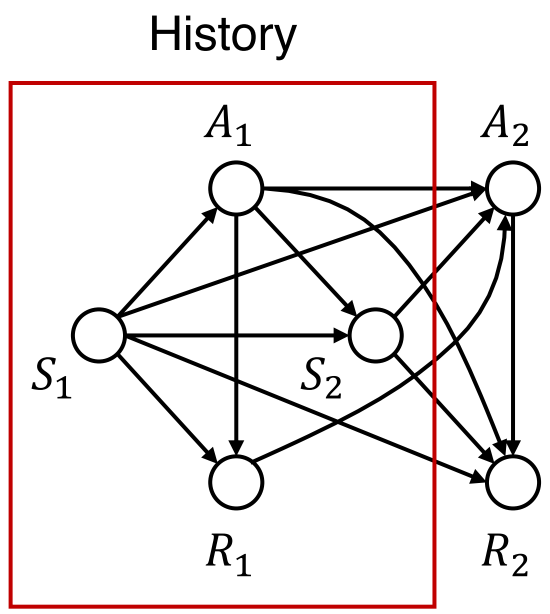
If we already tried a treatment, we might not try it again



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

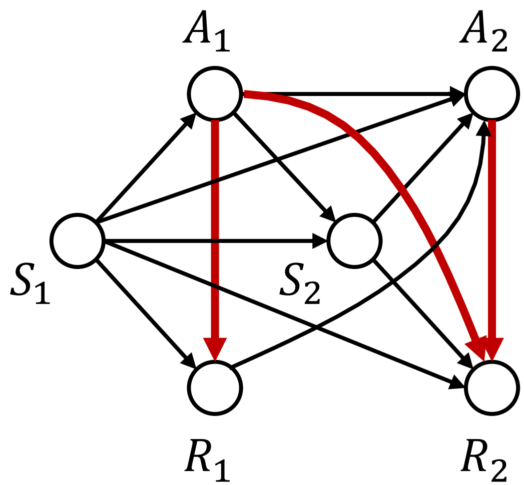
Treating Anna over time

- Our next action and outcome may depend on the whole history



Treating Anna over time

- Not only is this a complicated causal graph, it is not a Markov decision process either!

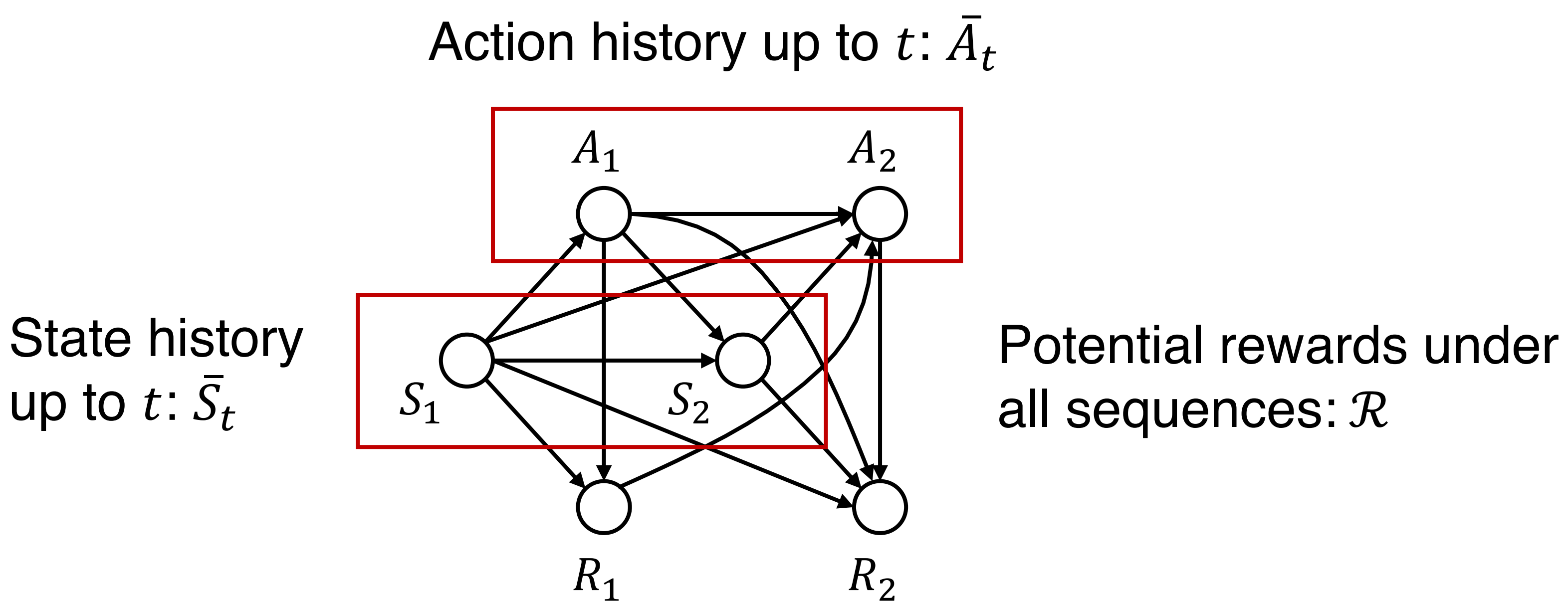


How can we find the effect of our policy on the expected reward[1]?

[1] The picture is slightly misleading: which arrows we care about depend on which effect we care about

Notation

- A little necessary notation



---

[1]The picture is slightly misleading: which arrows we care about depend on which effect we care about

Assumptions:

- Conditions for **identifiability** of potential reward:

Single-step case

Strong ignorability:

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

“No *hidden* confounders”

Overlap:

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

“All actions possible”

Sequential case

Sequential randomization:

$$\mathcal{R} \perp\!\!\!\perp A_t \mid \bar{S}_t, \bar{A}_{t-1}$$

“Reward indep. of policy given history”

Positivity:

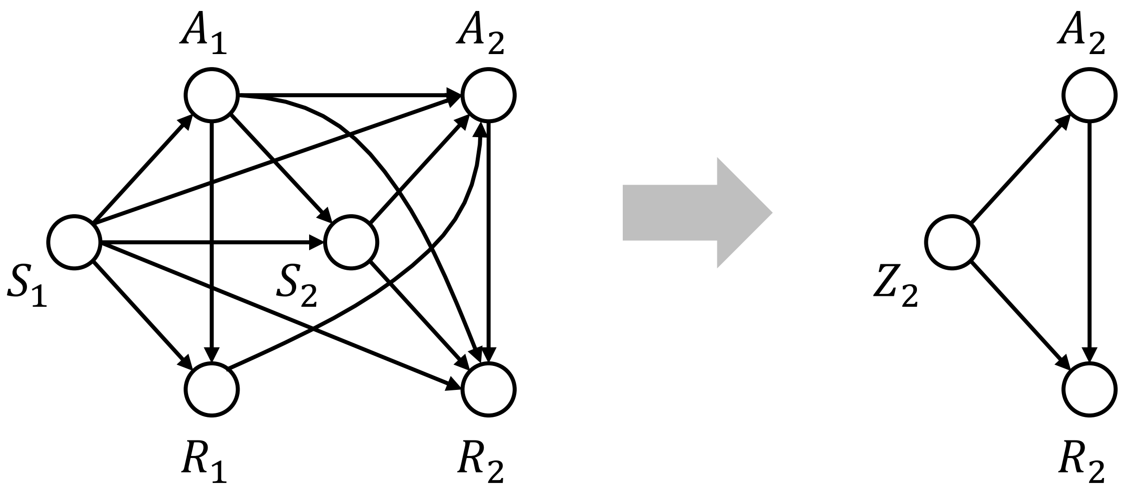
$$\forall a, t\colon\ p(A_t = a \mid \bar{S}_t, \bar{A}_{t-1}) > 0$$

“All actions possible at all times”

Summarizing history

- Conditioning on history of states and actions is algorithmically challenging: different length of history, high dimensionality etc

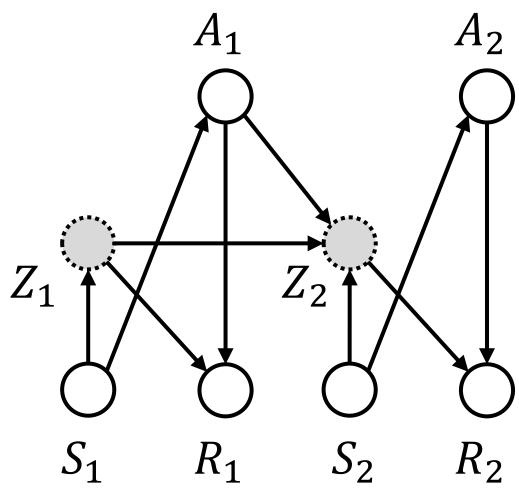
- Instead, we may attempt to summarize history in a variable Z



Summarizing history

- We can use sequence models such as recurrent neural networks and LSTMs to summarize state-action history

- For causal reasoning, we need assumptions to hold w.r.t. Z

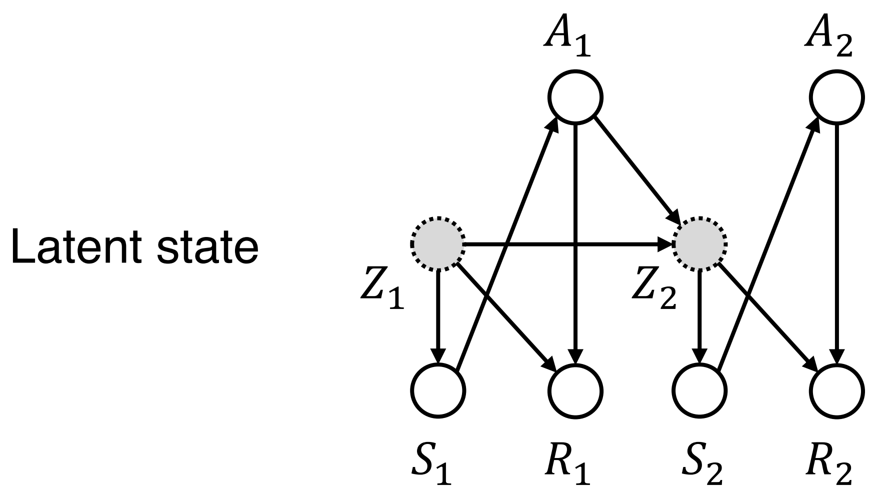


?

$$\mathcal{R} \perp\!\!\!\perp A_t \mid Z_t$$

Partially observable MDPs (POMDPs)

- A related concept are POMDPs, in which what we observe is a **partial/noisy version** of a latent Markov system



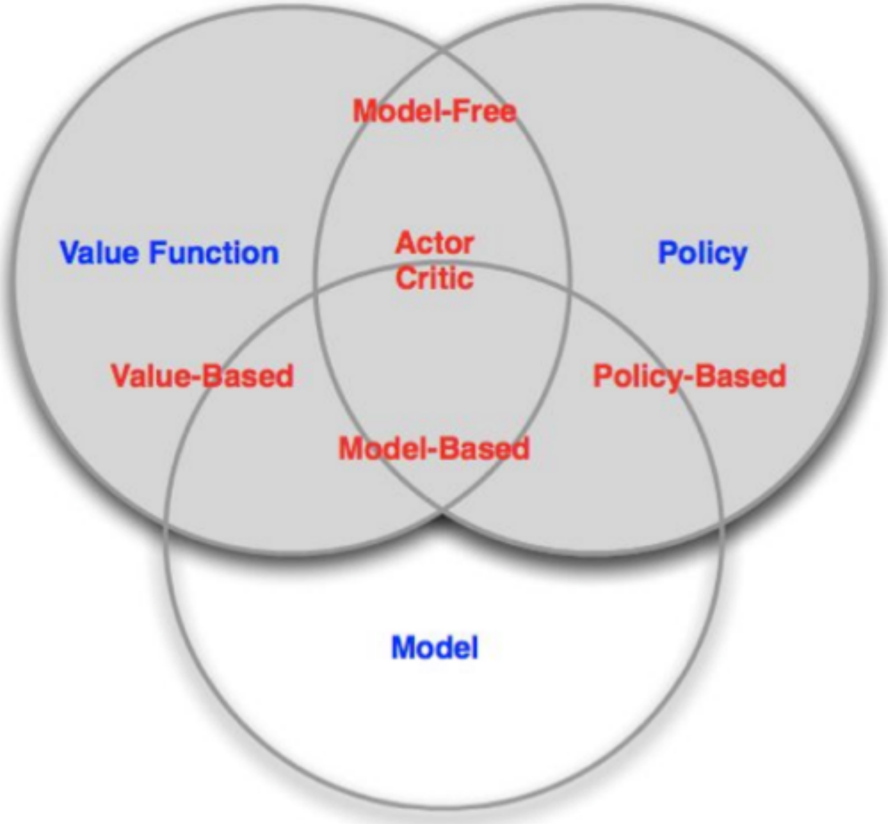
1. Decision processes

2. Learning from batch (off-policy) data

3. Reinforcement learning paradigms

4. Applications

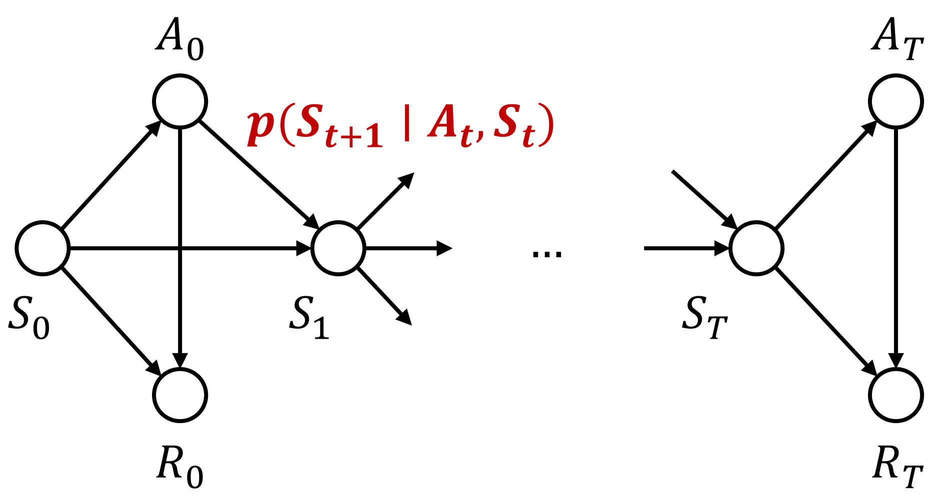
Reinforcement learning



Slide by David Silver

Model-based RL

- Explicitly model state transitions: $p(S_{t+1} \mid A_t, S_t)$
- Can be used for **planning** to discover optimal policy
- Predicts future states, and acts according to learned policy



Policy search

- Directly optimizes over the (possibly stochastic) **policy** $\pi(s_t)$, (not using a value function)
- Can be used both to learn an observed policy (e.g. man-made) and to search for optimal policies
- An example: Try an action out. If it had a good result, increase the probability of that happening again.
- Typically **on-policy** (need exploration)

Model-free RL: Q-learning

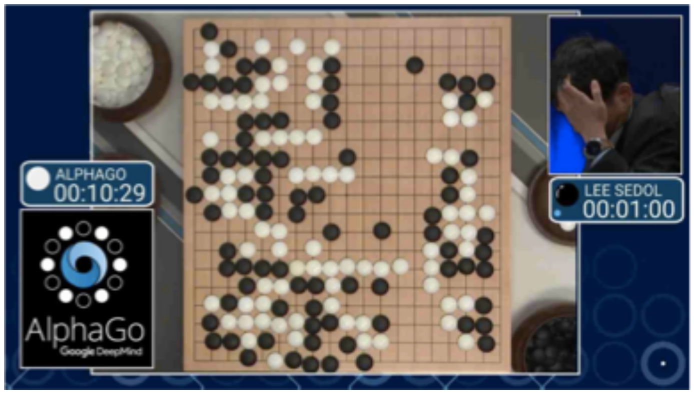
- **Off-policy, value-based** reinforcement learning method

- A Q-function $Q(s, a)$ assigns a value to each state-action pair (s, a) to represent the long-term reward of that action

- The best value function equals the expected future reward of taking an action in a state

$$Q^*(s, a) = \max_{\pi} \mathbb{E}_{\pi}[R_t \mid S_t = s, A_t = a]$$

Model-free RL



Bellman equation

- The Bellman equations states that the optimal Q-function has the property (where s' is the state after taking action a in s)

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

- Q-iteration repeatedly uses this rule to update current estimate

- If the state space is finite, Q can be represented by a table, and the optimum can be found through dynamic programming

Q-learning with function approximation

- When the state space is continuous, we have to rely on function approximation of Q (as opposed to a table)

- We can still use the Bellman equation, but are no longer guaranteed to find the optimum

$$R(Q) = \mathbb{E}_\pi\left[\left(\underbrace{\overset{\text{Reward}}{r} + \gamma \max_{a'} \hat{Q}(s', a')}_{\text{Target, typically an old estimate of } Q} - \underbrace{Q(s, a)}_{\text{Prediction}}\right)^2\right]$$

Q-learning with function approximation

- Typically proceeds iteratively:

1. Initialize target $\hat{Q}$
2. **Repeat:**
    1. Minimize Q-loss $R(Q)$ w.r.t. prediction/policy network Q
    2. After some time, update target $\hat{Q}$ with recent Q

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

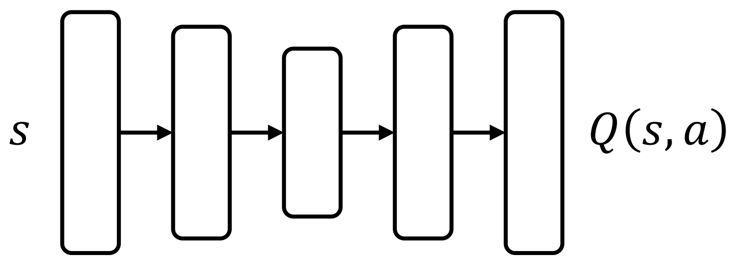
Deep Q-learning

- Function approximation with deep neural networks

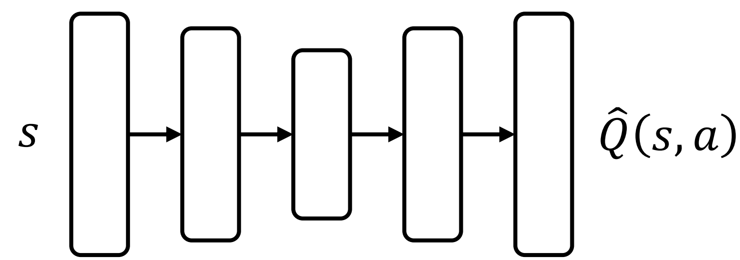
Reward

$$R(Q) = \mathbb{E}_\pi \left[\Big(\underbrace{r + \gamma \max_{a'} \hat{Q}(s', a')}_{\text{Target}} - \underbrace{Q(s,a)}_{\text{Prediction}} \Big)^2 \right]$$

Q-network, θ



Target network, $\hat{\theta}$



Same architecture, different weights

Q-learning with function approximation

- Optimization dynamics:
  - The goal post $\hat{Q}$ (target) keeps changing as we update Q
  - No guarantee to converge to optimal Q in general case

- To be causally sound in non-Markov case, we should predict Q from whole history. This is not typically done!

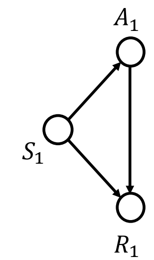
- Distributional shift (like in single-step case):
  - We don't have samples from the policy we are evaluating!
  - Our loss function is an expectation under a distribution from which we have no samples!

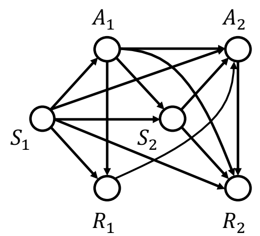
Counterfactuals are very different!

- **Single-step case:**

  Counterfactuals are changes to a single action

  (control->treated and vice versa)





- **Sequential case:**

  Counterfactuals are changes to whole sequences of actions!

Coping with distributional shift Pt. II

- Last time we discussed importance sampling estimators

$$\mathbb{E}_{\boldsymbol{q(x)}}[f(x)] = \mathbb{E}_{\boldsymbol{p(x)}}\left[f(x)\frac{q(x)}{p(x)}\right] \approx \frac{1}{n}\sum_{i=1}^{n}\frac{q(x_i)}{p(x_i)}f(x_i)$$

- Where $x_1, \ldots, x_n \sim p(x)$

- What happens when x is a sequence?

Importance sampling for RL

- First of all, we don't observe i.i.d. samples of state-action transitions, but whole sequences

- We let the loss be the expected total loss over a sequence

$$R(Q) = \mathbb{E}_\pi\left[\sum_{t=1}^{T} \underbrace{\left(r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a') - Q(s_t, a_t)\right)^2}_{L_t = L(s_t, a_t, s_{t+1})}\right]$$

Importance sampling for RL

- Product of importance weights over time

$$\mathbb{E}_{\boldsymbol{q}}\left[\sum_{t=1}^{T} L_t\right] = \mathbb{E}_{\boldsymbol{p}}\left[\sum_{t=1}^{T} L_t \prod_{t'=0}^{t} \frac{\boldsymbol{q}(a_{t'} \mid s_{t'})}{\boldsymbol{p}(a_{t'} \mid s_{t'})}\right]$$

- Could have extremely high variance if **product** of large factors (if **p** and **q** are very different)

- Effective sample size $ESS = \frac{(\sum w_i)^2}{\sum w_i^2}$, where w_i weight of sequence

Trading off bias for variance

- Even if importance sampling yields an unbiased estimator of the policy value, a biased version with smaller variance might be preferred

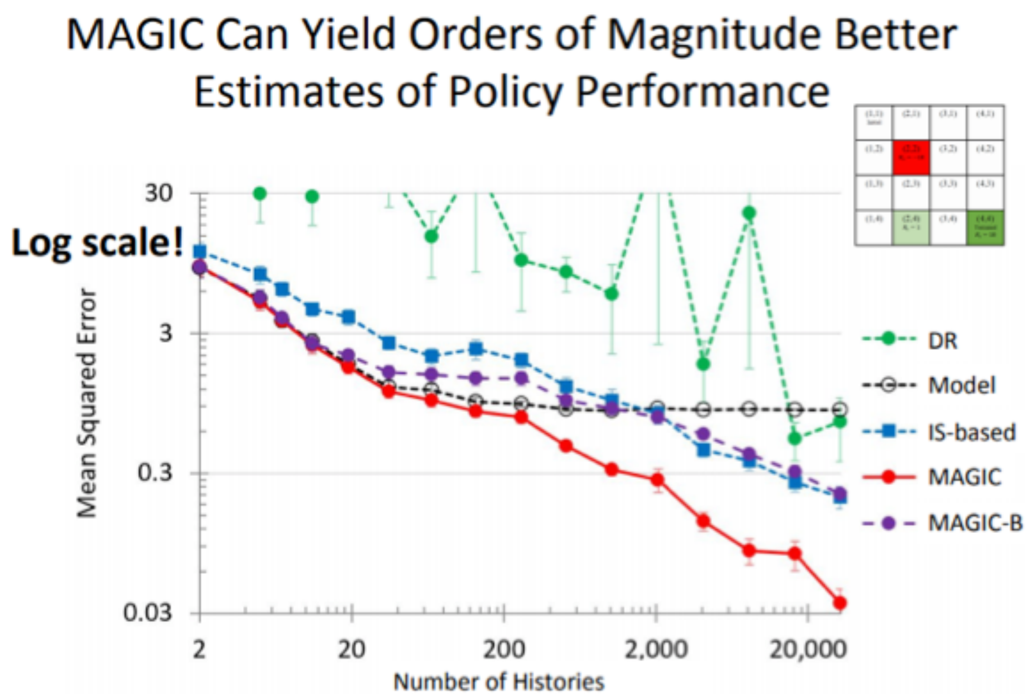
- Think of bias-variance decomposition:

$$MSE = Bias + Variance + Noise$$

- Can get smaller prediction error by trading off bias and variance

MAGIC[1]

- Combining model-based and model-free RL with IS



[1]Thomas and Brunskill, *ICML*, 2016

1. Decision processes

2. Learning from batch (off-policy) data

3. Reinforcement learning paradigms

4. Applications

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)

Many concerns

- **Can we summarize history well?**

  If we measured everything we are theoretically OK. But did we keep everything we need in our summary?

- **Is overlap/positivity enough?**

  Even if it gives us unbiased estimators, what is the sample complexity? Is it reasonable?

- **Do we have the right reward function?**

Many concerns

- **Hidden confounding**

  Did we measure all the necessary variables?

- **Expectation vs risk**

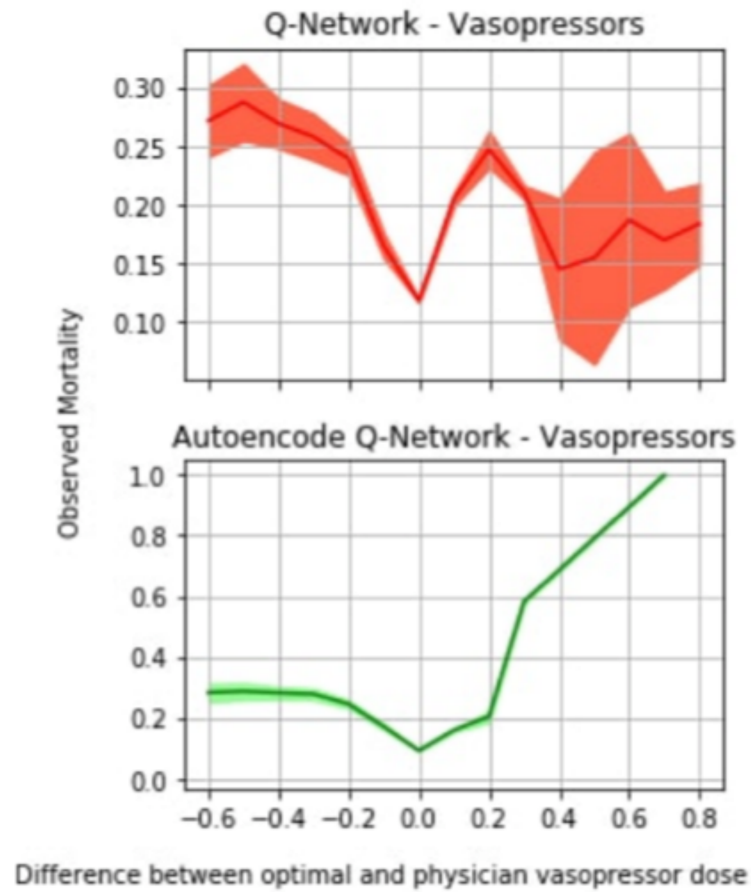
  What can we tolerate in terms of outlier behavior?

- **Evaluation**

  Can we trust our models estimates?

Sepsis treatment[1]

- Work on using Q-learning to discover the right treatment for sepsis
- Dosage of a) fluids, and b) vasopressors
- Compare found policy to physician's in terms of mortality



[1]Raghu et al. *MLHC,* 2017

Conclusions

- Off-policy reinforcement learning is **strictly** harder than counterfactual estimation
- The causal problems with standard regression are even greater
- Both conceptual/theoretical and practical challenges remain
- We need to take care that we are trying to solve a problem that is actually interesting