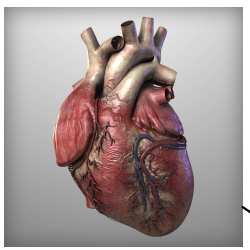
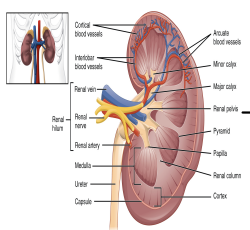
Machine Learning
Reinforcement Learning

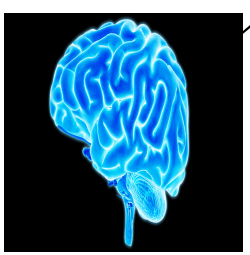
Rajesh Ranganath

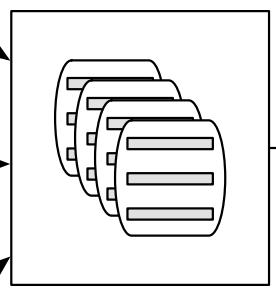
- We have a goal
- We want to take actions to reach that goal
- The actions depend on the **state**

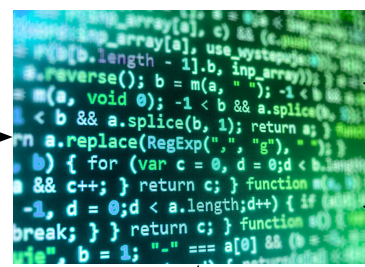
Interventions in Healthcare

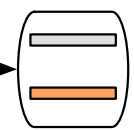








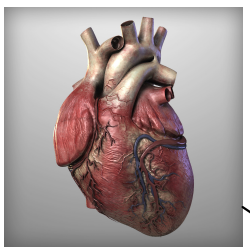


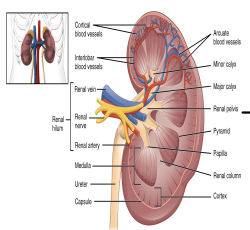


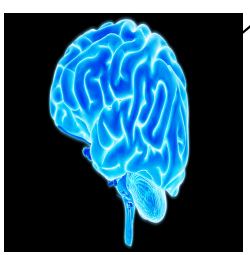
Health Status

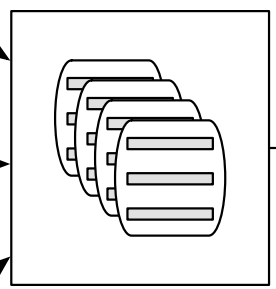
Medication A?

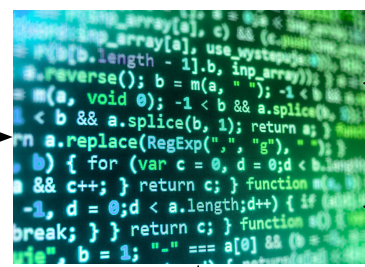
Interventions in Healthcare

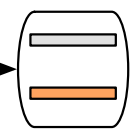








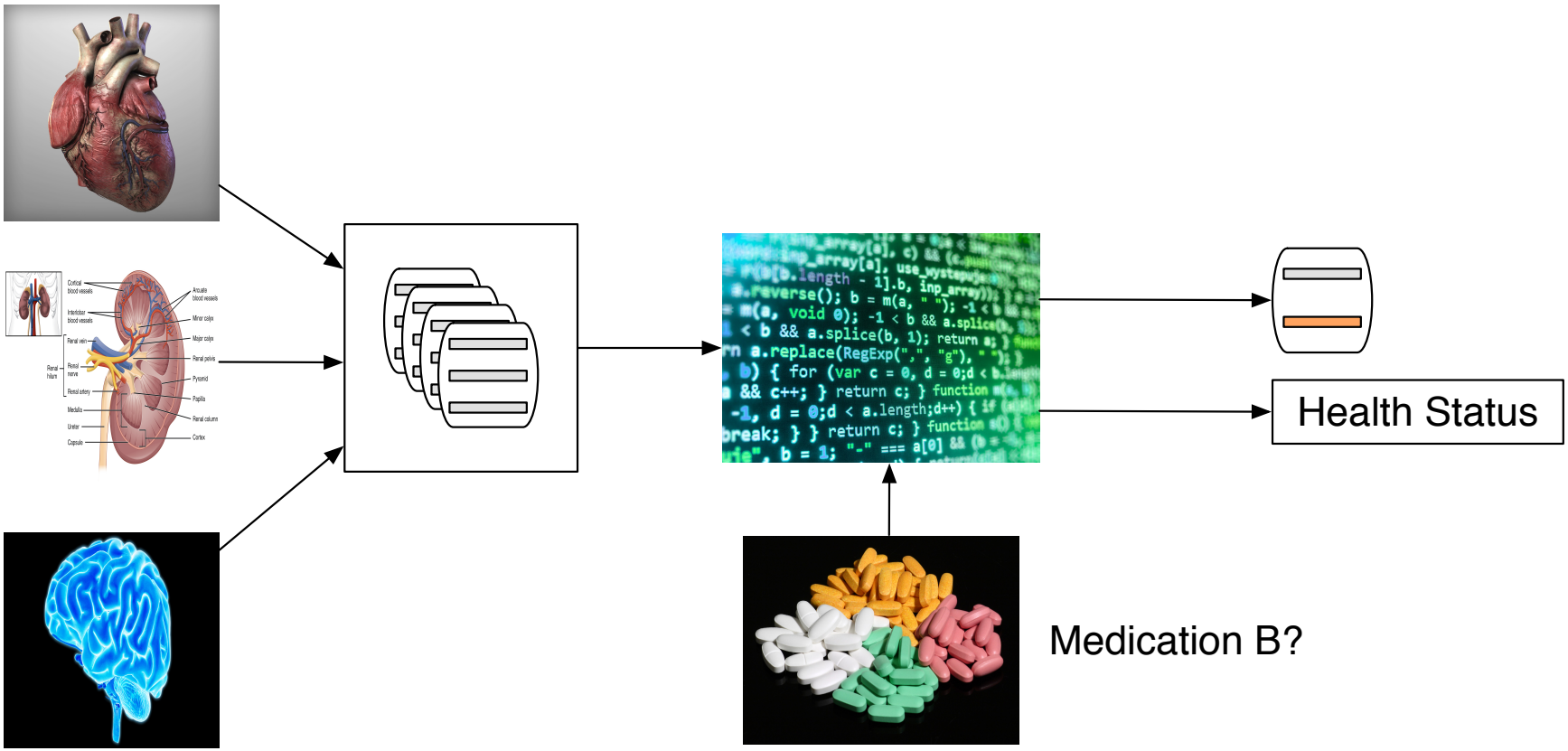




Health Status

Medication B?

Medication A or Medication B?



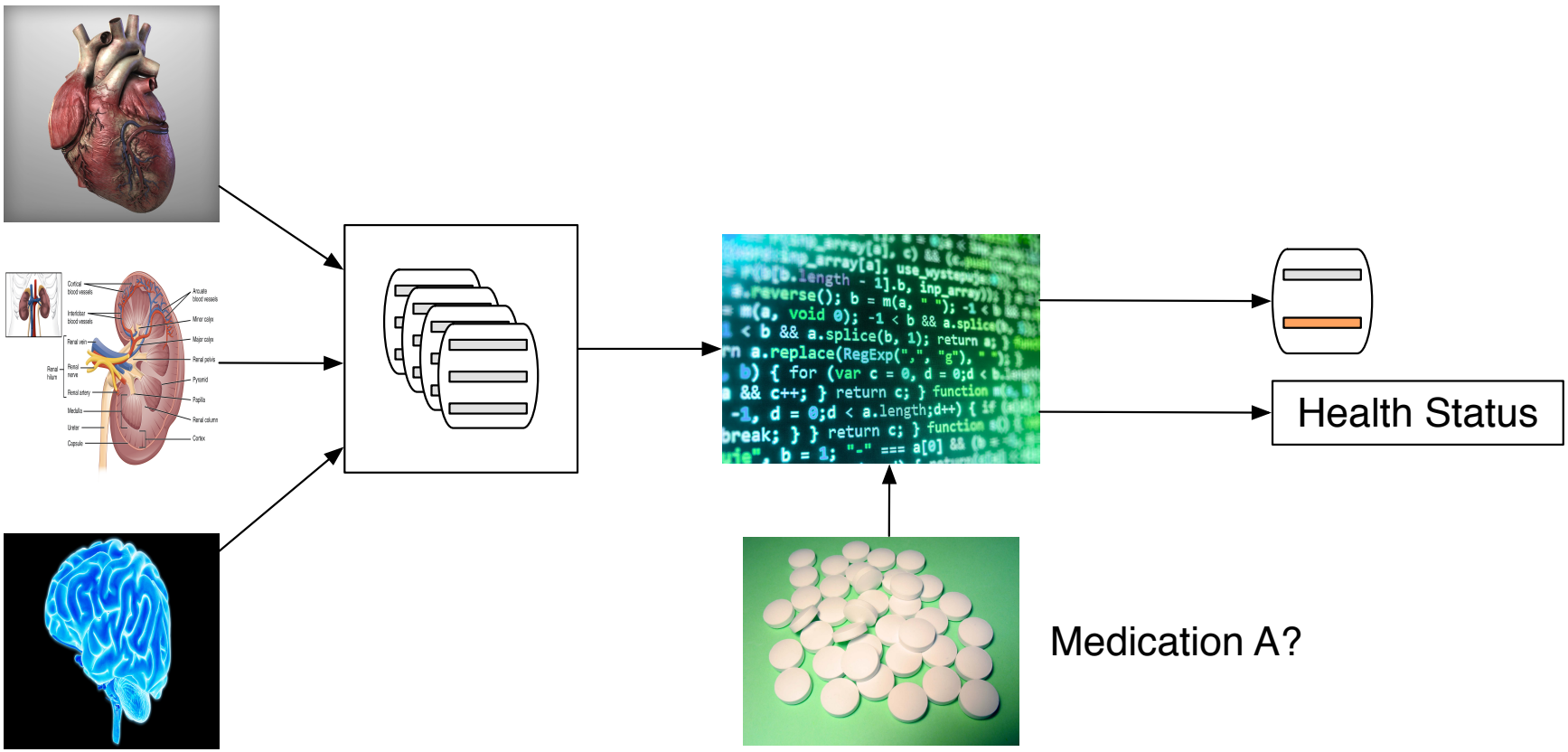
Question: Should we give medication A or medication B?

Causal Inference

Causal inference seeks to estimate the effect of an intervention

- Which statin to give to a patient with hyperlipidemia?
- Which medication to give to a depressed person?
- Which patients to give hospice care to?

Medication A Dosing?



Question: How much of medication A should we give every week?

Solutions?

- Build a time series model and make prediction?
  Why should this work?

- Do causal inference from a fixed time point?
  Sequential Decision Making?

Markov Decision Processes

A Markov Decision Process

- S: A collection of states $\mathbf{s}$
- A: A collection of actions $\mathbf{a}$ to take in each state
- $p(\mathbf{s}_{t+1} \mid \mathbf{a}_t, \mathbf{s}_t)$: State transition
- $r(r_{t+1} \mid \mathbf{a}_t, \mathbf{s}_t)$: Reward distribution
  Don't really need to be random

Markov Decision Processes

Mapping the medical example

- S: The physiological state of a person
- A: Drug A or B along with dosage
- $p(\mathbf{s}_{t+1} \mid \mathbf{a}_t, \mathbf{s}_t)$: How physiology changes with the drug/dosage given
- $r(r_{t+1} \mid \mathbf{a}_t, \mathbf{s}_t)$: Quality of life for day after medication

Why Markov?

The Markov assumptions says

Things only depend on the previous time step

Is this okay?

Why Markov?

The Markov assumptions says

Things only depend on the previous time step

Is this okay?

Can append previous states into one big one. All Markov

What about states that are hidden?

Markov Decision Processes

What's the goal of working with MDPs?

Markov Decision Processes

What's the goal of working with MDPs?

Find a policy: π

$$\pi_{\boldsymbol{\theta}}(\mathbf{a}_t \,|\, \mathbf{s}_t)$$

that maximizes total γ-weighted reward

$$\sum_t \gamma_t r_t$$

How do we do this?

First question: What do we know?

Let's assume

- We can simulate a policy
- Observe rewards for the that policy
- A finite time horizon T

Policy Gradients

Goal maximize

$$\mathscr{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} [r_{\boldsymbol{\tau}}]$$

A trajectory $\boldsymbol{\tau}$ of actions, states and rewards

$$r_{\boldsymbol{\tau}} = \sum_{t=1}^{T} \gamma_t r_t$$

What's random?

- The actions sampled from the policy
- The state transition $p(\mathbf{s}_{t+1} \,|\, \mathbf{s}_t, \mathbf{a}_t)$
- The rewards $r(r_{t+1} \,|\, \mathbf{s}_t, \mathbf{a}_t)$

Policy Gradients

Goal maximize

$$\mathscr{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} [r_{\boldsymbol{\tau}}]$$

A trajectory $\boldsymbol{\tau}$ of actions, states and rewards

$$r_{\boldsymbol{\tau}} = \sum_{t=1}^{T} \gamma_t r_t$$

How can we optimize this?

Policy Gradients

Goal maximize

$$\mathscr{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} [r_{\boldsymbol{\tau}}]$$

A trajectory $\boldsymbol{\tau}$ of actions, states and rewards

$$r_{\boldsymbol{\tau}} = \sum_{t=1}^{T} \gamma_t r_t$$

How can we optimize this?

Score function gradients

Policy Gradients

Goal maximize

$$\mathscr{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} [r_{\boldsymbol{\tau}}]$$

A trajectory $\boldsymbol{\tau}$ of actions, states and rewards

$$r_{\boldsymbol{\tau}} = \sum_{t=1}^{T} \gamma_t r_t$$

$$\nabla_{\boldsymbol{\theta}} \mathscr{L}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} [r_{\boldsymbol{\tau}}] = \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} [\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{\tau}) r_{\boldsymbol{\tau}}]$$

Policy Gradients

Goal maximize

$$\mathscr{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} [r_{\boldsymbol{\tau}}]$$

A trajectory $\boldsymbol{\tau}$ of actions, states and rewards

$$r_{\boldsymbol{\tau}} = \sum_{t=1}^{T} \gamma_t r_t$$

$$\nabla_{\boldsymbol{\theta}} \mathscr{L}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} [r_{\boldsymbol{\tau}}] = \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} [\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{\tau}) r_{\boldsymbol{\tau}}]$$

Policy Gradients

Goal maximize

$$\mathscr{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} [r_{\boldsymbol{\tau}}]$$

A trajectory $\boldsymbol{\tau}$ of actions, states and rewards

$$r_{\boldsymbol{\tau}} = \sum_{t=1}^{T} \gamma_t r_t$$

$$\hat{\nabla}_{\boldsymbol{\theta}} \mathscr{L}(\boldsymbol{\theta}) = \frac{1}{S} \sum_{s=1}^{S} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{\tau}_s) r_{\boldsymbol{\tau}^s}$$

Algorithm

1. Sample from current policy
2. Observe state transition and rewards
3. Compute Monte Carlo unbiased gradient
4. Maximize via stochastic optimization

Policy Gradients

Goal maximize

$$\mathscr{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} [r_{\boldsymbol{\tau}}]$$

A trajectory $\boldsymbol{\tau}$ of actions, states and rewards

$$r_{\boldsymbol{\tau}} = \sum_{t=1}^{T} \gamma_t r_t$$

$$\nabla_{\boldsymbol{\theta}} \mathscr{L}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} [r_{\boldsymbol{\tau}}] = \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} [\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{\tau}) r_{\boldsymbol{\tau}}]$$

Problem?

Policy Gradients

Goal maximize

$$\mathscr{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} [r_{\boldsymbol{\tau}}]$$

A trajectory $\boldsymbol{\tau}$ of actions, states and rewards

$$r_{\boldsymbol{\tau}} = \sum_{t=1}^{T} \gamma_t r_t$$

$$\nabla_{\boldsymbol{\theta}} \mathscr{L}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} [r_{\boldsymbol{\tau}}] = \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} [\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{\tau}) r_{\boldsymbol{\tau}}]$$

Problem?

Variance!

Policy Gradients

Goal maximize

$$\mathscr{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\tau}\sim\pi_{\boldsymbol{\theta}}}[r_{\boldsymbol{\tau}}]$$

A trajectory $\boldsymbol{\tau}$ of actions, states and rewards

$$r_{\boldsymbol{\tau}} = \sum_{t=1}^{T} \gamma_t r_t$$

- Can Rao-Blackwellize
- Use control variates or baseline using $\mathbb{E}[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{\tau})] = 0$

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta}) &= \mathbb{E}_{\boldsymbol{\tau}\sim\pi_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{\tau}) r_{\boldsymbol{\tau}}] \\ &= \mathbb{E}_{\boldsymbol{\tau}\sim\pi_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{\tau})(r_{\boldsymbol{\tau}} - b)]\end{aligned}$$

Actor-Critic Methods

Assume $\gamma_t = 1$ for simplicity

Actor-Critic Methods

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathscr{L}(\boldsymbol{\theta}) &= \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{\tau}) r_{\boldsymbol{\tau}} \right] \\
&= \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} \left[r_{\boldsymbol{\tau}} \sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t \,|\, \mathbf{s}_t) \right] \\
&= \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} \left[\sum_{t'=1}^{T} r_{t'} \sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t \,|\, \mathbf{s}_t) \right] \\
&= \mathbb{E} \left[\mathbb{E} \left[\sum_{t'=1}^{T} r_{t'} \sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t \,|\, \mathbf{s}_t) \,|\, \mathbf{s}_{\leq t}, \mathbf{a}_{<t} \right] \right] \\
&= \mathbb{E} \left[\mathbb{E} \left[\sum_{t=1}^{T} \sum_{t'=1}^{t} r_{t'} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t \,|\, \mathbf{s}_t) \,|\, \mathbf{s}_{\leq t}, \mathbf{a}_{<t} \right] \right] \\
&\quad + \mathbb{E} \left[\mathbb{E} \left[\sum_{t=1}^{T} \sum_{t'=t+1}^{T} r_{t'} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t \,|\, \mathbf{s}_t) \,|\, \mathbf{s}_{\leq t}, \mathbf{a}_{<t} \right] \right] \\
&= \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\boldsymbol{\theta}}} \left[\sum_{t=1}^{T} \sum_{t'=t+1}^{T} r_{t'} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t \,|\, \mathbf{s}_t) \right]
\end{aligned}
$$

Actor-Critic Methods

$$\sum_{t=1}^{T} \mathbb{E}_{\tau \sim \pi_\theta} \left[\left(\sum_{t'=t+1}^{T} r_{t'} \right) \nabla_\theta \log \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t) \right]$$

Define future reward estimate

$$\sum_{t'=t+1}^{T} r_{t'}$$

Policy gradient uses sample based future reward estimate.
Other ideas?

Actor-Critic Methods

$$\sum_{t=1}^{T} \mathbb{E}_{\tau \sim \pi_\theta} \left[\left(\sum_{t'=t+1}^{T} r_{t'} \right) \nabla_\theta \log \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t) \right]$$

Define future reward estimate

$$\sum_{t'=t+1}^{T} r_{t'}$$

Policy gradient uses sample based future reward estimate.
Other ideas?

What if the reward was estimated directly?

$$\sum_{t=1}^{T} \mathbb{E}\left[\mathbb{E}\left[\left(\sum_{t'=t+1}^{T} r_{t'}\right) | \mathbf{a}_t, \mathbf{s}_t\right] \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t | \mathbf{s}_t)\right]$$

Define the *Q-function*:

$$Q_t^{\pi}(\mathbf{a}_t, \mathbf{s}_t) = \mathbb{E}\left[\left(\sum_{t'=t+1}^{T} r_{t'}\right) | \mathbf{a}_t, \mathbf{s}_t\right]$$

$$\sum_{t=1}^{T} \mathbb{E}\left[\mathbb{E}\left[\left(\sum_{t'=t+1}^{T} r_{t'}\right) | \mathbf{a}_t, \mathbf{s}_t\right] \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t | \mathbf{s}_t)\right]$$

Take

$$Q_t^{\pi}(\mathbf{a}_t, \mathbf{s}_t) = \mathbb{E}\left[\left(\sum_{t'=t+1}^{T} r_{t'}\right) | \mathbf{a}_t, \mathbf{s}_t\right]$$

Substitute

$$\nabla_{\boldsymbol{\theta}} \mathscr{L} = \sum_{t=1}^{T} \mathbb{E}\left[Q_t^{\pi}(\mathbf{a}_t, \mathbf{s}_t) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t | \mathbf{s}_t)\right]$$

What about variance?

Define the *Value function*

$$V_t^{\pi}(\mathbf{s}_t) = \mathbb{E}_{\pi(\mathbf{a}_t \mid \mathbf{s}_t)} \mathbb{E}\left[\left(\sum_{t'=t+1}^{T} r_{t'} \right) \mid \mathbf{a}_t, \mathbf{s}_t \right]$$

Then

$$\sum_{t=1}^{T} \mathbb{E}\left[V_t^{\pi}(\mathbf{s}_t) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t \mid \mathbf{s}_t) \right] = 0$$

Can use the value function to reduce variance!

Define the *Advantage function*

$$A_t^\pi(\mathbf{a}_t, \mathbf{s}_t) = Q_t^\pi(\mathbf{a}_t, \mathbf{s}_t) - V_t^\pi(\mathbf{s}_t)$$

- Measures action $\mathbf{a}_t$ better than the average policy action
- It's the advantage of action $\mathbf{a}_t$ over average under π

$$\nabla_{\boldsymbol{\theta}} \mathscr{L} = \sum_{t=1}^{T} \mathbb{E}\left[A_t^\pi(\mathbf{a}_t, \mathbf{s}_t) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t \,|\, \mathbf{s}_t)\right]$$

How do we make practical use of this decomposition?

Rewriting

$$Q_t^\pi(\mathbf{a}_t, \mathbf{s}_t) = \mathbb{E}\left[\left(\sum_{t'=t+1}^{T} r_{t'}\right) | \, \mathbf{a}_t, \mathbf{s}_t\right]$$

with

$$V_t^\pi(\mathbf{s}_t) = \mathbb{E}_{\pi(\mathbf{a}_t \,|\, \mathbf{s}_t)}\mathbb{E}\left[\left(\sum_{t'=t+1}^{T} r_{t'}\right) | \, \mathbf{a}_t, \mathbf{s}_t\right],$$

yields

$$\begin{aligned} Q_t^\pi(\mathbf{a}_t, \mathbf{s}_t) &= \mathbb{E}\left[\left(\sum_{t'=t+1}^{T} r_{t'}\right) | \, \mathbf{a}_t, \mathbf{s}_t\right] = \mathbb{E}\left[r_{t+1} + \left(\sum_{t'=t+2}^{T} r_{t'}\right) | \, \mathbf{a}_t, \mathbf{s}_t\right] \\ &= \mathbb{E}\left[r_{t+1} + \mathbb{E}\left[\left(\sum_{t'=t+2}^{T} r_{t'}\right) | \, \mathbf{a}_{t+1}, \mathbf{s}_{t+1}\right] | \, \mathbf{a}_t, \mathbf{s}_t\right] \\ &= \mathbb{E}\left[r_{t+1} + V_{t+1}^\pi(\mathbf{s}_{t+1}) | \, \mathbf{a}_t, \mathbf{s}_t\right] \end{aligned}$$

Advantage Actor-Critic

Simplify

$$A_t^\pi(\mathbf{a}_t, \mathbf{s}_t) = \mathbb{E}[r_{t+1} + V_{t+1}^\pi(\mathbf{s}_{t+1}) \,|\, \mathbf{a}_t, \mathbf{s}_t] - V_t^\pi(\mathbf{s}_t)$$

Only need to estimate $V_t^\pi(\mathbf{s}_t)$. How?

Advantage Actor-Critic

Simplify

$$A_t^\pi(\mathbf{a}_t, \mathbf{s}_t) = \mathbb{E}[r_{t+1} + V_{t+1}^\pi(\mathbf{s}_{t+1}) \,|\, \mathbf{a}_t, \mathbf{s}_t] - V_t^\pi(\mathbf{s}_t)$$

Only need to estimate $V_t^\pi(\mathbf{s}_t)$. How?

Monte Carlo

$$V_t^\pi(\mathbf{s}_t) \approx \frac{1}{J} \sum_{j=1}^{J} \sum_{t'=t+1}^{T} r_{t'}^j$$

Needs to sample sub-trajectories

Advantage Actor-Critic

Simplify

$$A_t^\pi(\mathbf{a}_t, \mathbf{s}_t) = \mathbb{E}[r_{t+1} + V_{t+1}^\pi(\mathbf{s}_{t+1}) \,|\, \mathbf{a}_t, \mathbf{s}_t] - V_t^\pi(\mathbf{s}_t)$$

Only need to estimate $V_t^\pi(\mathbf{s}_t)$. How?

Single Monte Carlo Estimate

$$V^\pi(\mathbf{s}_t) \approx \sum_{t'=t+1}^{T} r_{t'}$$

Advantage Actor-Critic

Simplify

$$A_t^\pi(\mathbf{a}_t, \mathbf{s}_t) = \mathbb{E}[r_{t+1} + V_{t+1}^\pi(\mathbf{s}_{t+1}) \,|\, \mathbf{a}_t, \mathbf{s}_t] - V_t^\pi(\mathbf{s}_t)$$

Only need to estimate $V_t^\pi(\mathbf{s}_t)$. How?

What about regression?

Advantage Actor-Critic

Simplify

$$A_t^\pi(\mathbf{a}_t, \mathbf{s}_t) = \mathbb{E}[r_{t+1} + V_{t+1}^\pi(\mathbf{s}_{t+1}) \,|\, \mathbf{a}_t, \mathbf{s}_t] - V_t^\pi(\mathbf{s}_t)$$

Only need to estimate $V_t^\pi(\mathbf{s}_t)$. How?

Data

$$\left(s_t^j, \sum_{t'=t+1}^{T} r_{t'}^j \right)_{j=1}^{J}$$

Take a parametric function $f_{\boldsymbol{\phi}}$

$$min_{\boldsymbol{\phi}} \sum_{j=1}^{J} \left(\sum_{t'=t+1}^{T} r_{t'}^j - f_{\boldsymbol{\phi}}(s_t^j) \right)^2$$

Advantage Actor-Critic

Simplify

$$A_t^\pi(\mathbf{a}_t, \mathbf{s}_t) = \mathbb{E}[r_{t+1} + V_{t+1}^\pi(\mathbf{s}_{t+1}) \,|\, \mathbf{a}_t, \mathbf{s}_t] - V_t^\pi(\mathbf{s}_t)$$

Only need to estimate $V_t^\pi(\mathbf{s}_t)$. How?

Can we reuse the previous estimate?

Advantage Actor-Critic

Simplify

$$A_t^\pi(\mathbf{a}_t, \mathbf{s}_t) = \mathbb{E}[r_{t+1} + V_{t+1}^\pi(\mathbf{s}_{t+1}) \,|\, \mathbf{a}_t, \mathbf{s}_t] - V_t^\pi(\mathbf{s}_t)$$

Only need to estimate $V_t^\pi(\mathbf{s}_t)$. How?

Split

$$V^\pi(\mathbf{s}_t) = V^\pi(\mathbf{s}_{t+1}) + r_{t+1}$$

Use current reward along with previous estimate of future rewards

Advantage Actor-Critic

Simplify

$$A_t^{\pi}(\mathbf{a}_t, \mathbf{s}_t) = \mathbb{E}[r_{t+1} + V_{t+1}^{\pi}(\mathbf{s}_{t+1}) \,|\, \mathbf{a}_t, \mathbf{s}_t] - V_t^{\pi}(\mathbf{s}_t)$$

Only need to estimate $V_t^{\pi}(\mathbf{s}_t)$. How?

$$min_{\boldsymbol{\phi}} \sum_{j=1}^{J} (f_{\boldsymbol{\phi}^*}(s_{t+1}^j) + r_{t+1}^j - f_{\boldsymbol{\phi}}(s_t^j))^2$$

- Trades bias for variance
- Can put together into an algorithm

Start with initial policy $\pi_{\boldsymbol{\theta}}$ and value function $f_{\boldsymbol{\phi}}$

Run until convergence

1. Sample from policy $\pi_{\boldsymbol{\theta}}$
2. Observe states and rewards
3. Update policy with policy gradients

$$\nabla_{\boldsymbol{\theta}} \mathscr{L} = \sum_{t=1}^{T} \mathbb{E}\left[A^{\pi}(\mathbf{a}_t, \mathbf{s}_t) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t \,|\, \mathbf{s}_t)\right]$$

4. Update value function with stochastic gradients

$$\nabla_{\boldsymbol{\phi}} = \sum_{j=1}^{J} (f_{\boldsymbol{\phi}^*}(s^j_{t+1}) + r^j_{t+1} - f_{\boldsymbol{\phi}}(s^j_t)) \nabla_{\boldsymbol{\phi}} f_{\boldsymbol{\phi}}(s^j_t)$$

Actor-Critic with Advantage function

What's the actor? What's the critic?

- Actor: Policy
- Critic: Advantage, Q, Value functions

Can we learn a policy with just critics?

- Actor: Policy

- Critic: Advantage, Q, Value functions

Can we learn a policy with just critics?

- $Q^{\pi}(\mathbf{s}, \mathbf{a})$: Reward under policy π for $\mathbf{a}$ when in $\mathbf{s}$

- Use Q^{π} to find a better π^*

$$\pi^*(\mathbf{a} \mid \mathbf{s}) = \delta_{\mathbf{a}=\text{argmax}_{\mathbf{a}'} Q(\mathbf{a}',\mathbf{s})}$$

  Pick the best action

Fitted Value Iteration

Assume reward is $\sum_{t=0}^{\infty} \gamma^t r_t$, then

$$
\begin{aligned}
Q^{\pi}(\mathbf{s}, \mathbf{a}) &= \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \,|\, \mathbf{s}, \mathbf{a}\right] \\
&= \mathbb{E}_{\pi}\left[r(\mathbf{s}, \mathbf{a}) + \sum_{t=1}^{\infty} \gamma^t r_t \,|\, \mathbf{s}, \mathbf{a}\right] \\
&= r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' \,|\, \mathbf{s}, \mathbf{a})} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \,|\, \mathbf{s}'\right] \\
&= r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' \,|\, \mathbf{s}, \mathbf{a})}[V(\mathbf{s}')]
\end{aligned}
$$

Fitted Value Iteration

Assume reward is $\sum_{t=0}^{\infty} \gamma^t r_t$, then

$$
\begin{aligned}
Q^{\pi}(\mathbf{s}, \mathbf{a}) &= \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \,|\, \mathbf{s}, \mathbf{a}\right] \\
&= \mathbb{E}_{\pi}\left[r(\mathbf{s}, \mathbf{a}) + \sum_{t=1}^{\infty} \gamma^t r_t \,|\, \mathbf{s}, \mathbf{a}\right] \\
&= r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' \,|\, \mathbf{s}, \mathbf{a})} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \,|\, \mathbf{s}'\right] \\
&= r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' \,|\, \mathbf{s}, \mathbf{a})}[V(\mathbf{s}')]
\end{aligned}
$$

Can use this to build an algorithm

- $Q(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' \,|\, \mathbf{s}, \mathbf{a})}[V(\mathbf{s}')]$
- $V(s) = \underbrace{\max_a Q(s, a)}_{\text{Choose Best Policy}}$

Fitted Value Iteration

Assume reward is $\sum_{t=0}^{\infty} \gamma^t r_t$, then

$$
\begin{aligned}
Q^{\pi}(\mathbf{s}, \mathbf{a}) &= \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \,|\, \mathbf{s}, \mathbf{a}\right] \\
&= \mathbb{E}_{\pi}\left[r(\mathbf{s}, \mathbf{a}) + \sum_{t=1}^{\infty} \gamma^t r_t \,|\, \mathbf{s}, \mathbf{a}\right] \\
&= r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' \,|\, \mathbf{s}, \mathbf{a})} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \,|\, \mathbf{s}'\right] \\
&= r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' \,|\, \mathbf{s}, \mathbf{a})}[V(\mathbf{s}')]
\end{aligned}
$$

Storing all of V can be hard. Use a function $f_{\boldsymbol{\phi}}$

- $y_j = \max_{a'} r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' \,|\, \mathbf{s}_j, \mathbf{a}')}[f_{\boldsymbol{\phi}}(\mathbf{s}')]$
- $\min_{\boldsymbol{\phi}} \sum_{j=1}^{J} (y_j - f_{\boldsymbol{\phi}}(\mathbf{s}_j))^2$

Needs knowledge of dynamics

Fitted Q Iteration

Data comes in tuples $\mathbf{s}', \mathbf{s}, \mathbf{a}, r$

Idea: Use Q-function to approximate the Value function

$$V(\mathbf{s}') = \max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}')$$

Use function to learn Q

- Step one:

$$\begin{aligned} y_j &= r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' \mid \mathbf{s}_j, \mathbf{a}_j)}[f_{\boldsymbol{\phi}}(\mathbf{s}')] \\ &= r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}'_j, \mathbf{a}') \end{aligned}$$

- $\min_{\boldsymbol{\phi}} \sum_{j=1}^{J} (y_j - Q_{\boldsymbol{\phi}}(\mathbf{s}_j, \mathbf{a}_j))^2$

Doesn't require samples from a particular policy!

Fitted Q Iteration

Data comes in tuples $\mathbf{s}', \mathbf{s}, \mathbf{a}, r$

Idea: Use Q-function to approximate the Value function

$$V(\mathbf{s}') = \max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}')$$

Use function to learn Q

- Step one:

$$\begin{aligned} y_j &= r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' \mid \mathbf{s}_j, \mathbf{a}_j)}[f_{\boldsymbol{\phi}}(\mathbf{s}')] \\ &= r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}'_j, \mathbf{a}') \end{aligned}$$

- $\min_{\boldsymbol{\phi}} \sum_{j=1}^{J} (y_j - Q_{\boldsymbol{\phi}}(\mathbf{s}_j, \mathbf{a}_j))^2$

Doesn't require samples from a particular policy!

- How many y_j's to collect?
- What if the collection is too big?

Online Q Learning

Collection a single tuple $\mathbf{s}', \mathbf{s}, \mathbf{a}, r$ and update Q with it

- Take some action $\mathbf{a}_j$ in $\mathbf{s}_j$, observe $\mathbf{s}'_j, r(\mathbf{s}_j, \mathbf{a}_j)$

- Step one:

$$\begin{aligned} y_j &= r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' \mid \mathbf{s}_j, \mathbf{a}_j)}[f_{\boldsymbol{\phi}}(\mathbf{s}')] \\ &= r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}'_j, \mathbf{a}') \end{aligned}$$

- $\boldsymbol{\phi} = \boldsymbol{\phi} + \alpha \nabla_{\boldsymbol{\phi}} Q_{\boldsymbol{\phi}}(\mathbf{s}_j, \mathbf{a}_j)(y_j - Q_{\boldsymbol{\phi}}(\mathbf{s}_j, \mathbf{a}_j))$

Only needs a single sample at a time

Online Q Learning

Collection a single tuple $\mathbf{s}', \mathbf{s}, \mathbf{a}, r$ and update Q with it

- Take some action $\mathbf{a}_j$ in $\mathbf{s}_j$, observe $\mathbf{s}'_j, r(\mathbf{s}_j, \mathbf{a}_j)$

- Step one:

$$\begin{aligned} y_j &= r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' \mid \mathbf{s}_j, \mathbf{a}_j)}[f_{\boldsymbol{\phi}}(\mathbf{s}')] \\ &= r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}'_j, \mathbf{a}') \end{aligned}$$

- $\boldsymbol{\phi} = \boldsymbol{\phi} + \alpha \nabla_{\boldsymbol{\phi}} Q_{\boldsymbol{\phi}}(\mathbf{s}_j, \mathbf{a}_j)(y_j - Q_{\boldsymbol{\phi}}(\mathbf{s}_j, \mathbf{a}_j))$

Only needs a single sample at a time

Can something go wrong?

What if the first time in state s, we take an action that rarely gives a bad future state/reward

Exploration vs Exploitation

Exploration:

- Take new actions to understand if they are good

Exploitation:

- Take actions that are known to be good

Tradeoff when trying to learn, but still be good while learning

Online q-iteration exploits too much

Epsilon Greedy and Entropy

Two ways:

- Epsilon Greedy: Sample a random action with probably ϵ otherwise choose maximum

- Entropy Sampling: $\pi(\mathbf{a}_t \,|\, \mathbf{s}_t) \propto \exp\left(\frac{1}{T} Q_{\boldsymbol{\phi}}(\mathbf{a}_t, \mathbf{s}_t)\right)$

Helps trade exploration and exploitation

More trouble?

Why does it work?

What is online Q learning?

What is online Q learning?

Not a gradient method

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

Objective

$$\min(y_j - Q_{\boldsymbol{\phi}}(\mathbf{a}_j, \mathbf{s}_j))^2$$

Same as

$$\min(r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \max_{\mathbf{a}'} Q_{\boldsymbol{\phi}}(\mathbf{s}'_j, \mathbf{a}') - Q_{\boldsymbol{\phi}}(\mathbf{a}_j, \mathbf{s}_j))^2$$

Online Q-iteration

$$\boldsymbol{\phi} = \boldsymbol{\phi} + \alpha \nabla_{\boldsymbol{\phi}} Q_{\boldsymbol{\phi}}(\mathbf{s}_j, \mathbf{a}_j)(y_j - Q_{\boldsymbol{\phi}}(\mathbf{s}_j, \mathbf{a}_j))$$

is not gradient!

More trouble?

Collection a single tuple $\mathbf{s}', \mathbf{s}, \mathbf{a}, r$ and update Q with it

- Take some action $\mathbf{a}_j$ in $\mathbf{s}_j$, observe $\mathbf{s}'_j, r(\mathbf{s}_j, \mathbf{a}_j)$
- Step one:

$$\begin{aligned} y_j &= r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' \mid \mathbf{s}_j, \mathbf{a}_j)}[f_{\boldsymbol{\phi}}(\mathbf{s}')] \\ &= r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}'_j, \mathbf{a}') \end{aligned}$$

- $\boldsymbol{\phi} = \boldsymbol{\phi} + \alpha \nabla_{\boldsymbol{\phi}} (y_j - Q(\mathbf{a}_j, \mathbf{s}_j))$

How are the tuples collected?

- Start in state $\mathbf{s}$
- Take action $\mathbf{a}$
- Go to state $\mathbf{s}'$
- Set $\mathbf{s} = \mathbf{s}'$

Samples are not iid, standard regression unhappy!

Fixes

Correlated Samples:

- Store old samples in a buffer
- Update Q function with that buffer
- Can be parallelized

Not a proper gradient:

- Use a second Q function

$$\min(r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \max_{\mathbf{a}'} Q_{\hat{\boldsymbol{\phi}}}(\mathbf{s}'_j, \mathbf{a}') - Q_{\boldsymbol{\phi}}(\mathbf{a}_j, \mathbf{s}_j))^2$$

Guarantees?

Deep Q Learning

Deep Q Learning

- Save $\boldsymbol{\phi}$ as $\hat{\boldsymbol{\phi}}$
- Collect dataset $(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s'}_j, r_j)_{j=1}^{J}$
- For K iterations:
  - Sample a batch of data from the data set
  - Update parameters

$$\boldsymbol{\phi} = \boldsymbol{\phi} + \alpha \nabla_{\boldsymbol{\phi}} Q(\mathbf{s}_j, \mathbf{a}_j)(r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \max_{\mathbf{a}'} Q_{\hat{\boldsymbol{\phi}}}(\mathbf{s}'_j, \mathbf{a}') - Q_{\boldsymbol{\phi}}(\mathbf{a}_j, \mathbf{s}_j))$$

- Use Deep Networks for $Q_{\boldsymbol{\phi}}$

Deep Q Learning

Deep Q Learning

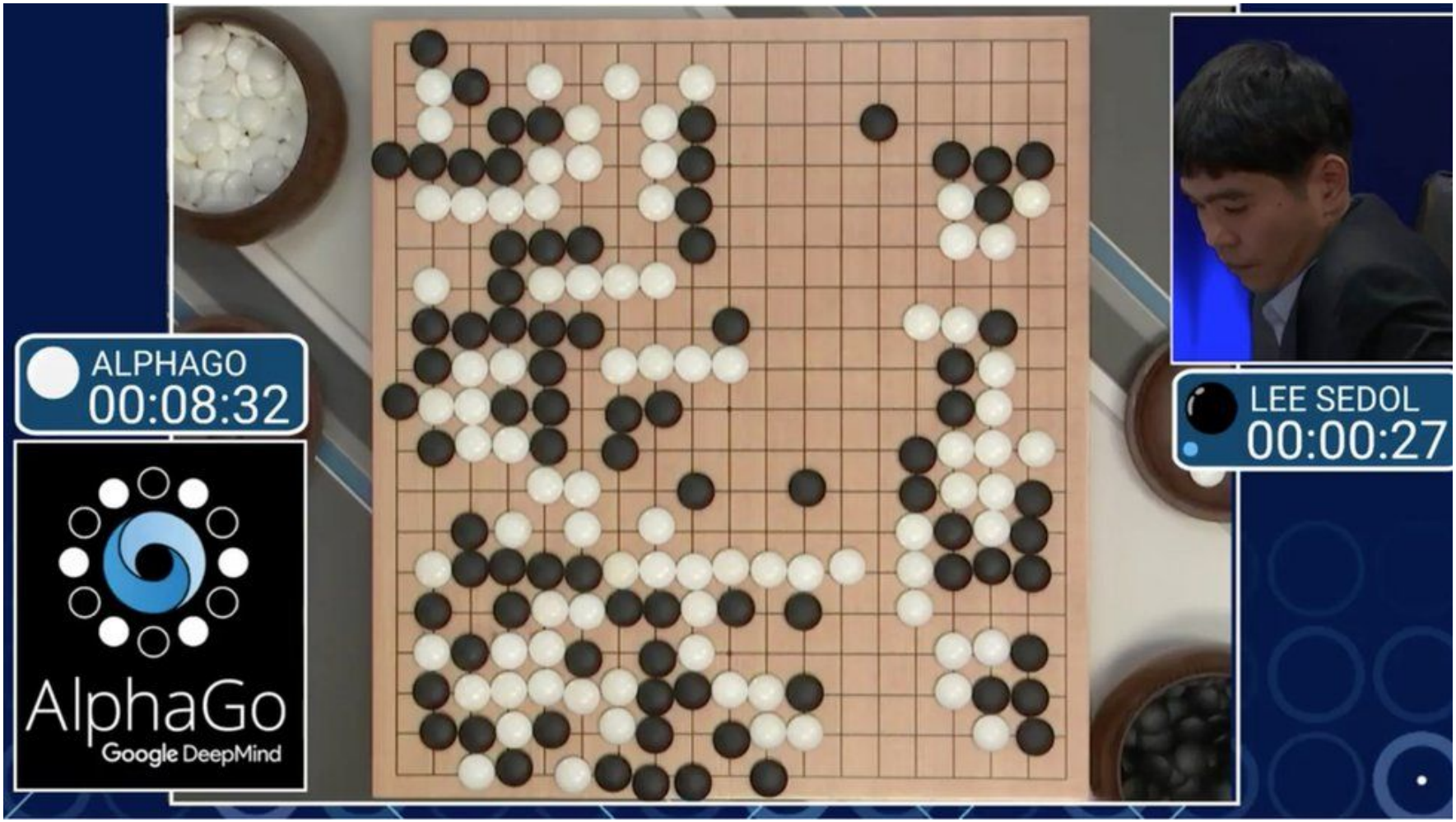
- Save $\boldsymbol{\phi}$ as $\hat{\boldsymbol{\phi}}$
- Collect dataset $(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s'}_j, r_j)_{j=1}^{J}$
- For K iterations:
  - Sample a batch of data from the data set
  - Update parameters

$$\boldsymbol{\phi} = \boldsymbol{\phi} + \alpha \nabla_{\boldsymbol{\phi}} Q(\mathbf{s}_j, \mathbf{a}_j)(r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \max_{\mathbf{a}'} Q_{\hat{\boldsymbol{\phi}}}(\mathbf{s}'_j, \mathbf{a}') - Q_{\boldsymbol{\phi}}(\mathbf{a}_j, \mathbf{s}_j))$$

- Use Deep Networks for $Q_{\boldsymbol{\phi}}$

Guarantees?

This has led to lots of fun



[BBC]

This has led to lots of fun

[OpenAI Five]

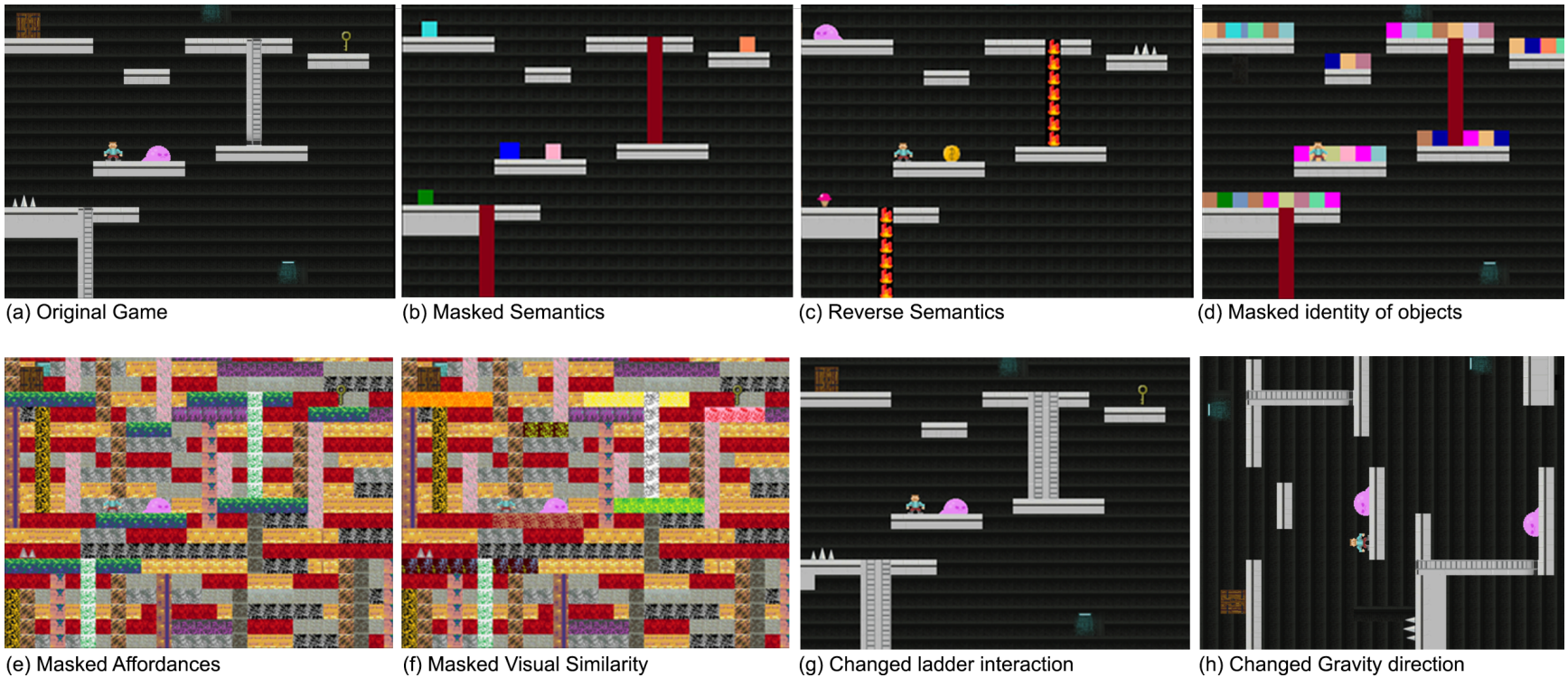
We were expecting to need sophisticated algorithmic ideas, such as hierarchical reinforcement learning, but we were surprised by what we found: the fundamental improvement we needed for this problem was scale. Achieving and utilizing that scale wasn't easy and was the bulk of our research effort!

[OpenAI Five]

- Today's class went from foundations to deep q learning quickly
- Variants plus massive scale led to some "super-human" performance

Is everything solved?

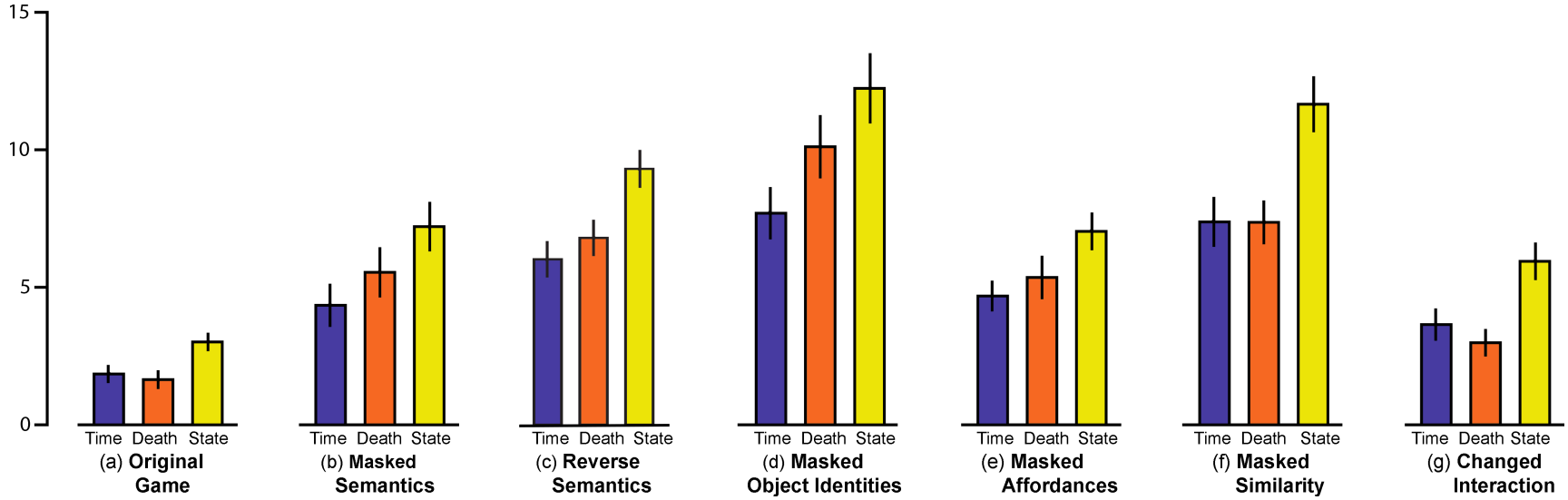
Investigating Human Priors for Playing Video Games

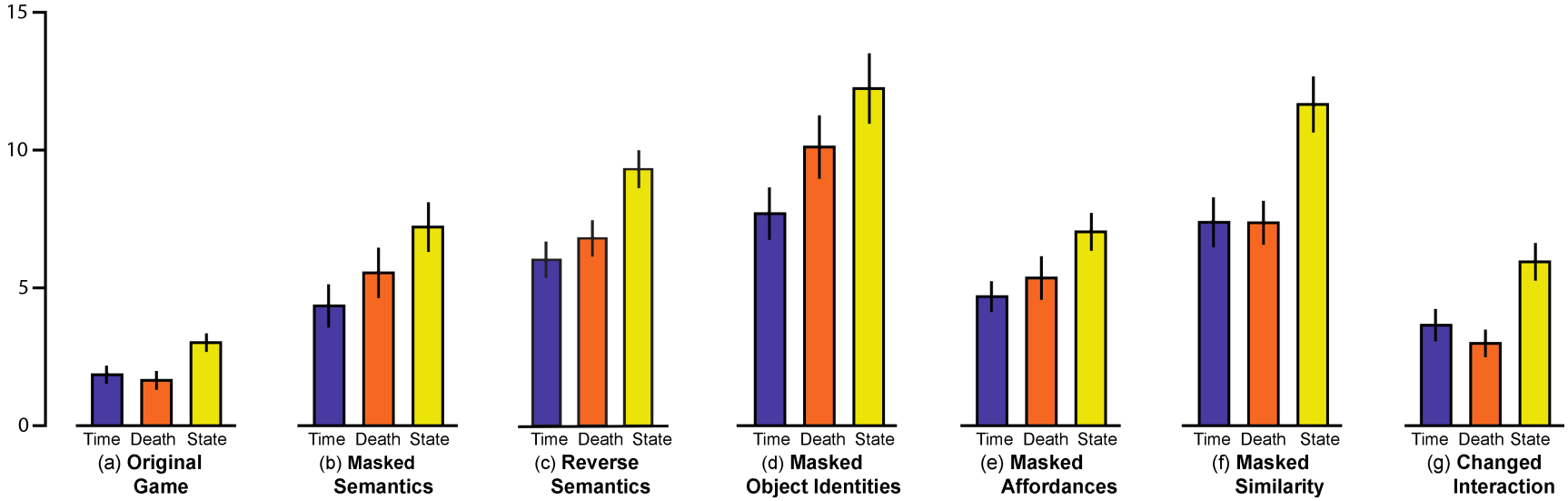


(a) Original Game (b) Masked Semantics (c) Reverse Semantics (d) Masked identity of objects

(e) Masked Affordances (f) Masked Visual Similarity (g) Changed ladder interaction (h) Changed Gravity direction

[Dubey+ 2018]


15
10
5
0
Time Death State
(a) Original Game
Time Death State
(b) Masked Semantics
Time Death State
(c) Reverse Semantics
Time Death State
(d) Masked Object Identities
Time Death State
(e) Masked Affordances
Time Death State
(f) Masked Similarity
Time Death State
(g) Changed Interaction


15
10
5
0
Time Death State
(a) Original Game
(b) Masked Semantics
(c) Reverse Semantics
(d) Masked Object Identities
(e) Masked Affordances
(f) Masked Similarity
(g) Changed Interaction


Average # of steps
10
5
0
Original
Semantics
Object
Affordance
Similarity
(e) Performance of RL agent

[Dubey+ 2018]

A Large Question

How to integrate, retain, and refine knowledge?

Learned about Sequential Decision Making: Markov Decision Processes

Policy Gradients: Directly update a policy of the world

Actor-Critic: Use estimates of future value of state to update policy

Q-learning: Estimate future value of state to define implicit policy

Deep Q-learning: Estimate future value of state to define an implicit policy with deep learning

What's the underlying principles?
Feels like a collection of bandaids

Where does the reward come from?