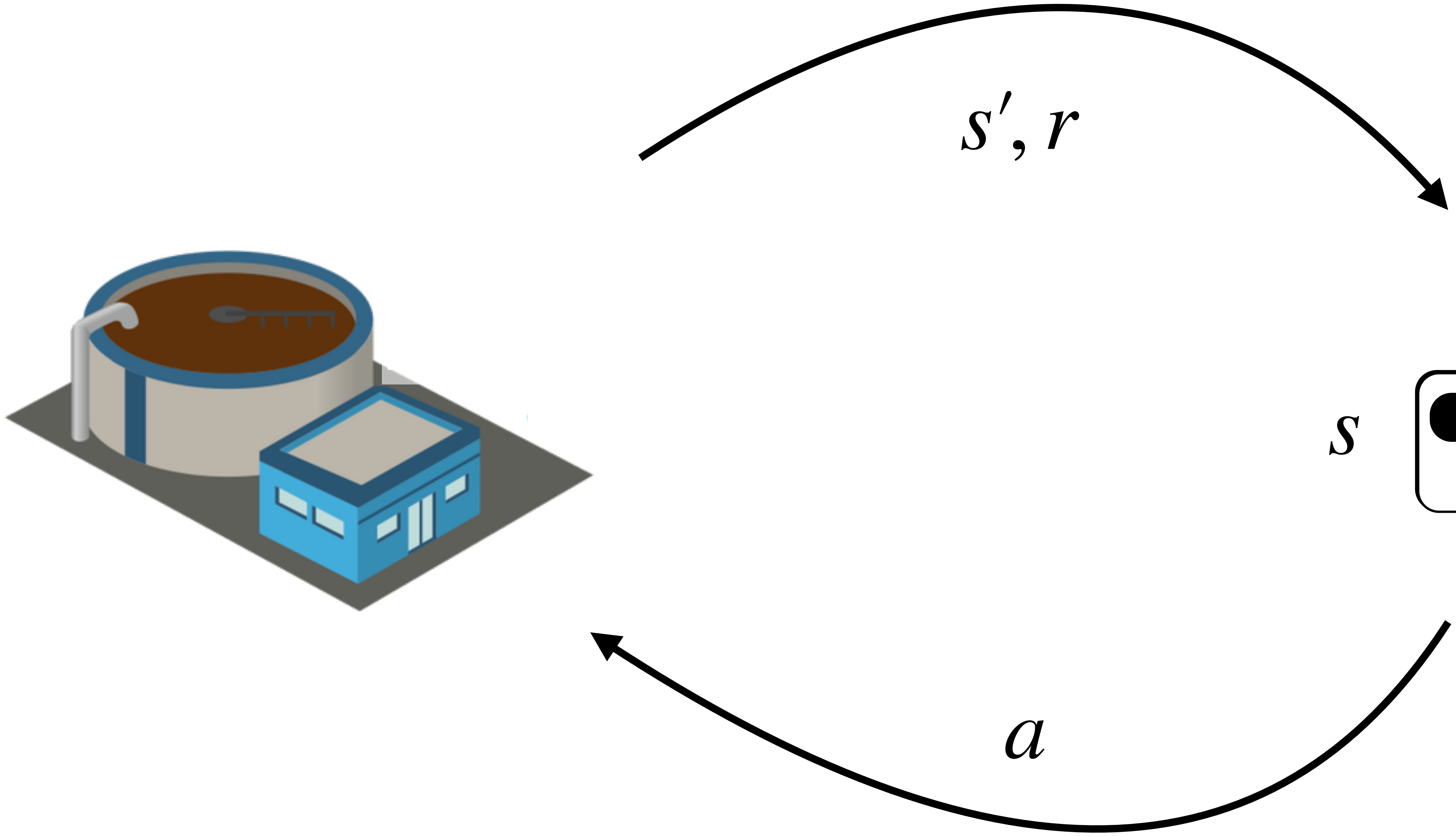
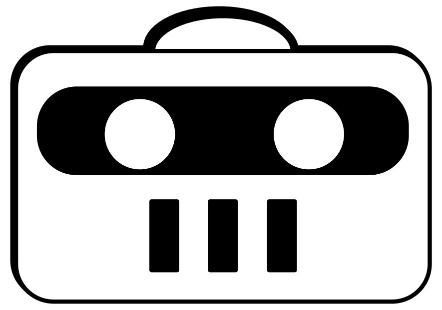
Advances in Value Estimation in Reinforcement Learning

Martha White
Associate Professor
University of Alberta
Canada CIFAR AI Chair

Problem Setting: Reinforcement Learning

- An agent interacts with the environment, to maximize reward





Agent learns policy $\pi(a \mid s)$
to maximize expected return

$$\mathbb{E}_\pi[G_t \mid S_t = s]$$

with $G_t = R_{t+1} + \gamma R_{t+2} + \dots$
and $\gamma > 0$

$s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, \dots$

Value Estimation is Central to Reinforcement Learning

- A value function v_π tells us the expected return from a state s, under policy π
  - $v_\pi(s) = \mathbb{E}_\pi[G_t \,|\, S_t = s] = \mathbb{E}_\pi[R + \gamma v_\pi(S') \,|\, S = s]$
- Action-value q_π allows us to improve the policy, by taking greedy actions
  - $q_\pi(s, a) = \mathbb{E}_\pi[G_t \,|\, S_t = s, A_t = a]$
  - $\pi'(s) = \arg\max_{a \in \mathscr{A}} q_\pi(s, a)$ obtains as good or higher return in each state
- Can also directly estimate q^*, the action-values for the optimal policy
- Value estimates critical for policy gradient methods (e.g., Actor Critic)

The Value Estimation Problem

- **Problem**: typically cannot exactly represent v_π
  - instead use $\hat{v}(s, \mathbf{w})$ parameterized value function (e.g., linear function, NN)
- **Goal**: Find approximate values $\hat{v}(\cdot, \mathbf{w}) \in \mathcal{F}$ to minimize the **value error**

$$\sum_s d(s) \ \left(\hat{v}(s, \mathbf{w}) - v_\pi(s)\right)^2$$

* for simplicity we use finite states

The Value Estimation Problem

- **Problem**: typically cannot exactly represent v_π
  - instead use $\hat{v}(s, \mathbf{w})$ parameterized value function (e.g., linear function, NN)
- **Goal**: Find approximate values $\hat{v}(\,\cdot\,, \mathbf{w}) \in \mathcal{F}$ to minimize the **value error**

$$\sum_s d(s) \, \left(\hat{v}(s, \mathbf{w}) - v_\pi(s)\right)^2$$

- **Issue**: Hard to directly optimize

* for simplicity we use finite states

Optimizing Value Error

- **Option 1**: Monte Carlo samples of return
  - If we can get samples of return G under policy π from state S we can update using regression to these samples
  - $\mathbf{w} \leftarrow \mathbf{w} + \eta \delta \nabla \hat{v}(S, \mathbf{w})$ for $\delta = G - \hat{v}(S, \mathbf{w})$
- **Issue**: Need to get samples of G, which can (a) be high variance and (b) delay online updating

Using Bootstrapped Return Estimates with Temporal Difference Learning

- **Option 2**: TD methods (including Q-learning)
  - this has been the **standard approach**
- The TD update is $\mathbf{w} \leftarrow \mathbf{w} + \eta\delta\nabla\hat{v}(S, \mathbf{w})$
  - where $\delta = \underbrace{R + \gamma\hat{v}(S', \mathbf{w})}_{\approx G} - \hat{v}(S, \mathbf{w})$ for transition (S, A, R, S')

Issues with Temporal Difference Learning

- The standard approach has been to use TD methods (including Q-learning)
- The TD update is $\mathbf{w} \leftarrow \mathbf{w} + \eta\delta \nabla \hat{v}(S, \mathbf{w})$
  - where $\delta = R + \gamma\hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$
- **Issue**: TD is only sound on-policy under linear function approximation
  - Can **diverge** under **off-policy** sampling
  - Can **diverge** under **nonlinear** function approximation (e.g., neural networks)

Why is TD not sound?

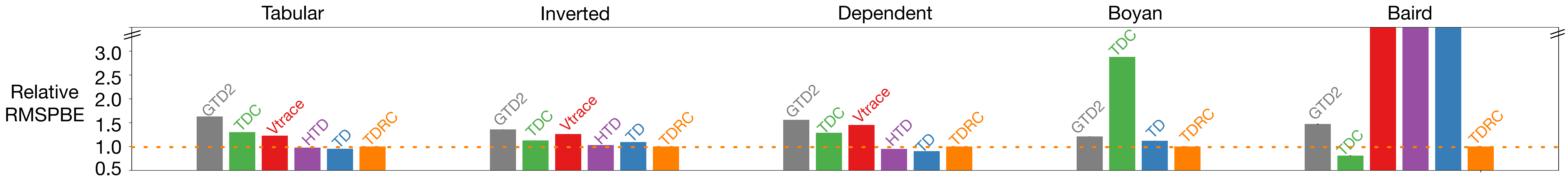
- The TD update is **not** the gradient of any objective function
  - Recall: $\mathbf{w} \leftarrow \mathbf{w} + \eta\delta\nabla\hat{v}(S, \mathbf{w})$ for $\delta = R + \gamma\hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$
- It is **not** the gradient of the squared TD error δ^2
  - $\nabla\delta^2 = \delta\left(\gamma\nabla\hat{v}(S', \mathbf{w}) - \nabla\hat{v}(S, \mathbf{w})\right)$
  - TD update omits $\delta\gamma\nabla\hat{v}(S', \mathbf{w})$

Why is TD not sound?

- The TD update is **not** the gradient of any objective function
  - Recall: $\mathbf{w} \leftarrow \mathbf{w} + \eta\delta\nabla\hat{v}(S, \mathbf{w})$ for $\delta = R + \gamma\hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$
- It is **not** the gradient of the squared TD error δ^2
  - $\nabla\delta^2 = \delta\left(\gamma\nabla\hat{v}(S', \mathbf{w}) - \nabla\hat{v}(S, \mathbf{w})\right)$
  - TD update omits $\delta\gamma\nabla\hat{v}(S', \mathbf{w})$
- Rather, with linear function approximation, TD can be seen as a stochastic update to solve a linear system of equations (iterative system solver)

What does this look like in practice?

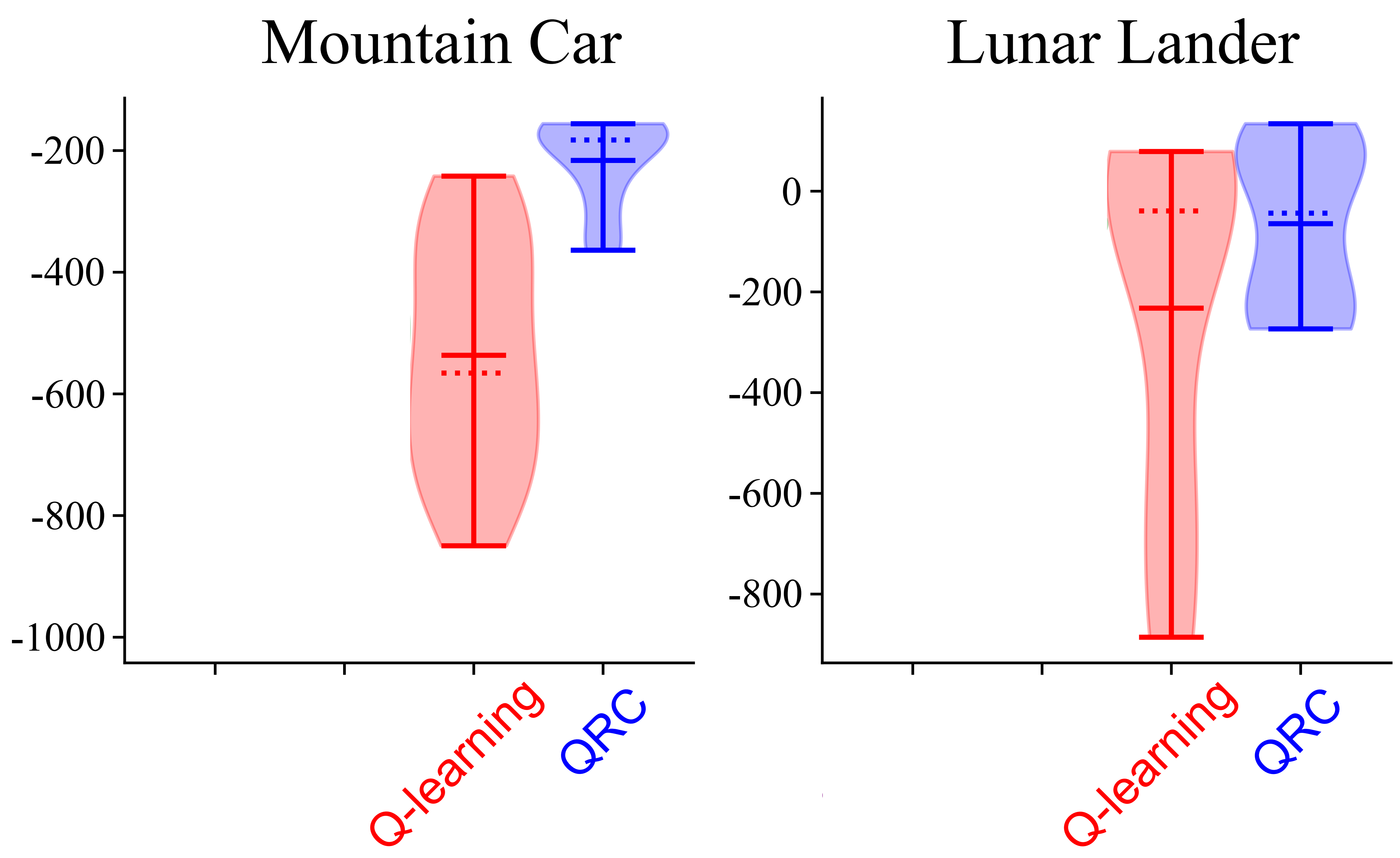
- TD generally performs very well…until it doesn't



- Our algorithm in this talk is TDRC (a better gradient TD method)
- GTD2 and TDC are standard (sound) gradient methods, that have been generally avoided because they seemed not to work too well
- TD diverges on Baird's counterexample (rightmost)

How about in control, with Q-learning?

- Might be manifesting primarily as sensitivity to hyperparameters
- May also explain the need for target networks (speculative)



How do we improve on TD methods?

There is a long history and a plethora of approaches for value estimation

Most correspond to minimizing one of two typical objectives

Outline for What's Coming Up

- A brief **history** of value estimation
  - particularly by explaining the **two key objectives**
- An explanation of our **generalized objective**
  - any why this generalization clarifies extensions to the **nonlinear setting**
- The naive **algorithm**, and how to **improve** on it significantly
  - aka, how we actually got gradient TD methods to work well

Squared Bellman Error

- The true values v_π satisfy the **Bellman equation**
  - $v_\pi = Tv_\pi$ for Bellman operator $(Tv_\pi)(s) \doteq \mathbb{E}_\pi[R + \gamma v_\pi(S') \mid S = s]$
  - i.e., $v_\pi(s) = \mathbb{E}_\pi[R + \gamma v_\pi(S') \mid S = s]$ for all states s

Squared Bellman Error

- The true values v_π satisfy the **Bellman equation**
  - $v_\pi = Tv_\pi$ for Bellman operator $(Tv_\pi)(s) \doteq \mathbb{E}_\pi[R + \gamma v_\pi(S') \mid S = s]$
  - i.e., $v_\pi(s) = \mathbb{E}_\pi[R + \gamma v_\pi(S') \mid S = s]$ for all states s
- Under function approximation, may not be able to find $v = Tv$

$$\overline{\mathrm{BE}}(\mathbf{w}) = \sum_{s \in \mathcal{S}} d(s)(T\hat{v}(\cdot, \mathbf{w})(s) - \hat{v}(s, \mathbf{w}))^2$$

$$= \sum_{s \in \mathcal{S}} d(s)\mathbb{E}_\pi[\delta(\mathbf{w}) \mid S = s]^2$$

Recall
$\delta(\mathbf{w}) = R + \gamma\hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$

Squared Bellman Error

- The true values v_π satisfy the **Bellman equation**
  - $v_\pi = Tv_\pi$ for Bellman operator $(Tv_\pi)(s) \doteq \mathbb{E}_\pi[R + \gamma v_\pi(S') \mid S = s]$
- Under function approximation (FA), may not be able to find $v = Tv$
- $\overline{\mathrm{BE}}(\mathbf{w}) = \sum_{s \in \mathcal{S}} d(s) \mathbb{E}_\pi[\delta(\mathbf{w}) \mid S = s]^2$
- **Issue**: **double sampling** problem
  - to get an unbiased sample of the gradient of this objective for a state, need two independent samples of next state and reward from that state

More on the double sampling problem

$$\overline{\mathrm{BE}}(\mathbf{w}) = \sum_{s \in \mathcal{S}} d(s) \mathbb{E}_\pi[\delta(\mathbf{w}) \,|\, S = s]^2$$

$$\nabla \overline{\mathrm{BE}}(\mathbf{w}) = 2 \sum_{s \in \mathcal{S}} d(s) \mathbb{E}_\pi[\delta(\mathbf{w}) \,|\, S = s] \mathbb{E}_\pi[\nabla \delta(\mathbf{w}) \,|\, S = s]$$

For a state S with sampled R and S', $\delta(\mathbf{w}) \nabla \delta(\mathbf{w})$ is not an unbiased sample:

$$\mathbb{E}_\pi[\delta(\mathbf{w}) \nabla \delta(\mathbf{w}) \,|\, S = s] \neq \mathbb{E}_\pi[\delta(\mathbf{w}) \,|\, S = s] \mathbb{E}_\pi[\nabla \delta(\mathbf{w}) \,|\, S = s]$$

Recall: $\delta(\mathbf{w}) = R + \gamma \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$

An Aside: Why not use Squared TD Error?

$$\overline{\text{TDE}}(\mathbf{w}) = \sum_{s \in \mathcal{S}} d(s) \mathbb{E}_\pi[\delta(\mathbf{w})^2 \,|\, S = s]$$

$$\nabla \overline{\text{TDE}}(\mathbf{w}) = 2 \sum_{s \in \mathcal{S}} d(s) \mathbb{E}_\pi[\delta(\mathbf{w}) \nabla \delta(\mathbf{w}) \,|\, S = s]$$

Then $\delta(\mathbf{w}) \nabla \delta(\mathbf{w})$ is an unbiased sample of this gradient

Reason: the resulting solution is typically **bad**

Linear Projected Bellman error

- Objective underlying **Temporal Difference (TD)** learning
- For linear FA, TD finds v that satisfies projected fixed point $v = \Pi T v$
  - Projection Π projects Tv back to the linear function space
- Objective: $\overline{\text{PBE}}(\mathbf{w}) = \sum_{s \in \mathcal{S}} d(s) \Big((\Pi T \hat{v}(\,\cdot\,, \mathbf{w}))(s) - \hat{v}(\,\cdot\,, \mathbf{w})(s) \Big)^2$
- **Issue**: **restricted to the linear** setting
  - Plus sometimes it can produce poor solutions
  - BE is better connected to the value error

Summary of Motivation and History

- TD can diverge under off-policy sampling and nonlinear function approximation
- Significant **progress** since the introduction of the linear $\overline{\text{PBE}}$ and the resulting gradient TD algorithms, which ensure convergence (2009)
- $\overline{\text{PBE}}$ primarily for the **linear** setting
    - nonlinear $\overline{\text{PBE}}$ relatively complex, with Hessian-vector products
- $\overline{\text{BE}}$ **difficult to optimize** due to the double-sampling problem
    - plus, it has **identifiability** issues
    - recent **positive** developments for double-sampling using a conjugate form

Key Points for this Talk

- We use the same **conjugate form** to develop a **Generalized $\overline{\text{PBE}}$**
- Exploit **insights** from the literature, for linear $\overline{\text{PBE}}$ and $\overline{\text{BE}}$, to obtain
  - new theoretical results on the solution quality of the value estimate
  - new algorithmic approaches to optimize the $\overline{\text{PBE}}$

- "A Generalized Projected Bellman Error for Off-policy Value Estimation in Reinforcement Learning", JMLR, 2022
- Paper on arXiv about extension to Huber losses

Andrew Patterson

Let’s start by **deriving** the Generalized $\overline{\text{PBE}}$

Bellman Error reformulated with an auxiliary variable

$$y^2 = \max_{h \in \mathbb{R}} 2yh - h^2$$

$$\begin{aligned}
\overline{\mathrm{BE}}(\boldsymbol{w}) &= \sum_{s \in \mathcal{S}} d(s) \mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]^2 \\
&= \sum_{s \in \mathcal{S}} d(s) \max_{h \in \mathbb{R}} \left(2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]\, h - h^2\right) \\
&= \max_{h \in \mathcal{F}_{\text{all}}} \sum_{s \in \mathcal{S}} d(s) \left(2\mathbb{E}_\pi[\delta(\boldsymbol{w}) \mid S = s]\, h(s) - h(s)^2\right)
\end{aligned}$$

where $\mathscr{F}_{\text{all}}$ is the space of all functions

Why is this useful?

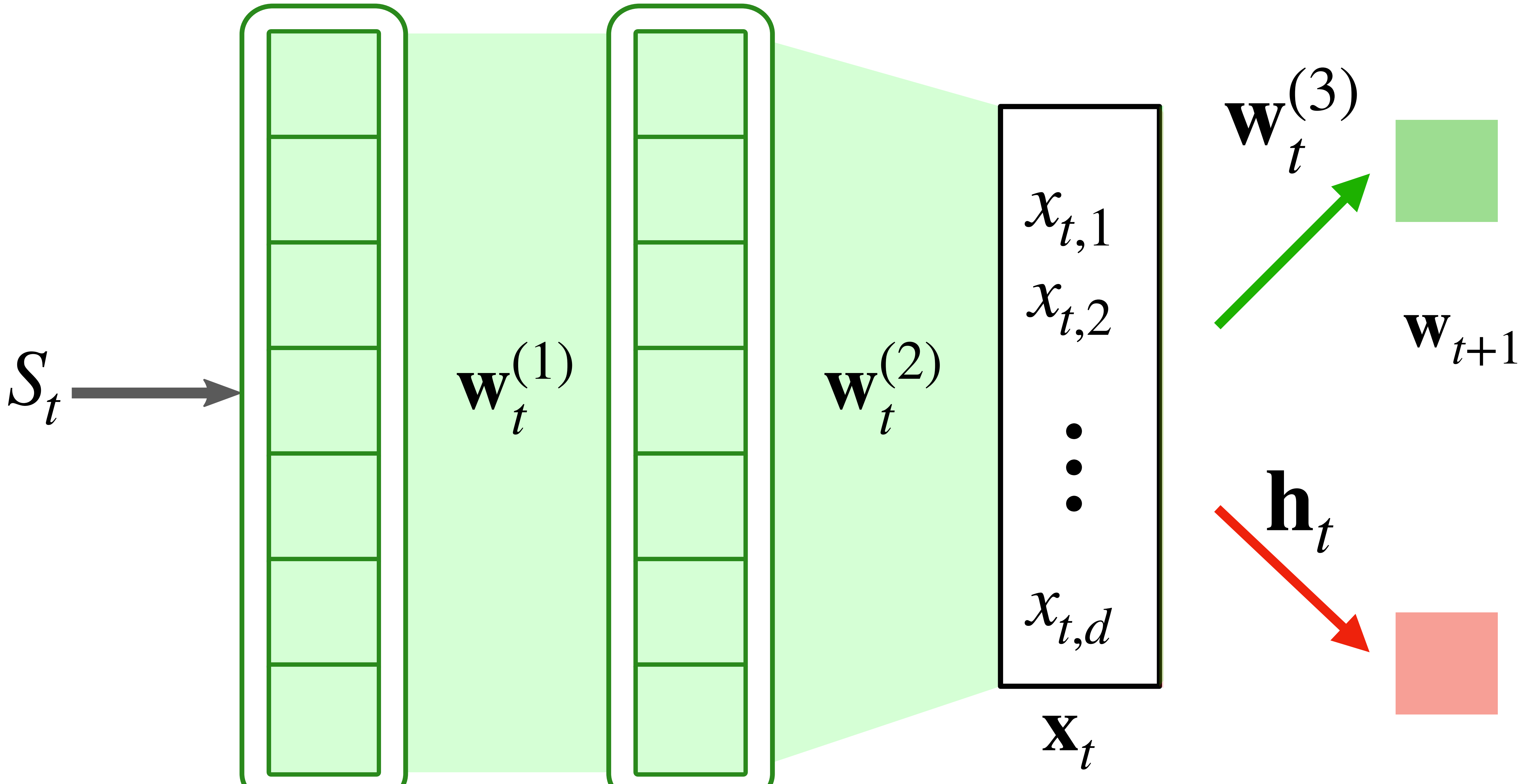
- Given h, computing a **gradient update** for the weights is **straightforward**
- Let $c_s(\mathbf{w}, h) = 2\mathbb{E}_\pi[\delta(\mathbf{w}) \mid S = s]h(s) - h(s)^2$
- $\overline{\text{BE}}(\mathbf{w}) = \max_{h \in \mathcal{F}_{\text{all}}} \sum_s d(s) c_s(\mathbf{w}, h)$
- $$\begin{aligned} \nabla_w c_s(\mathbf{w}, h) &= 2\mathbb{E}_\pi[\nabla \delta(\mathbf{w}) \mid S = s]h(s) \\ &= 2\mathbb{E}_\pi[\gamma \nabla \hat{v}(S', \mathbf{w}) - \nabla \hat{v}(S, \mathbf{w}) \mid S = s]h(s) \end{aligned}$$
- Stochastic gradient update for w: $h(s)\big(\gamma \nabla \hat{v}(S', \mathbf{w}) - \nabla \hat{v}(S, \mathbf{w})\big)$

Learning h is also straightforward

- The optimal solution for h is $h^*(s) = \mathbb{E}_\pi[\delta(\mathbf{w}) \,|\, S = s]$
- Update for h is a simple regression update with δ as a target

The architecture and updates

- **Green** part standard TD or Q-learning. **Red** is the added auxiliary variable



$\hat{v}(S_t, \mathbf{w}_t)$

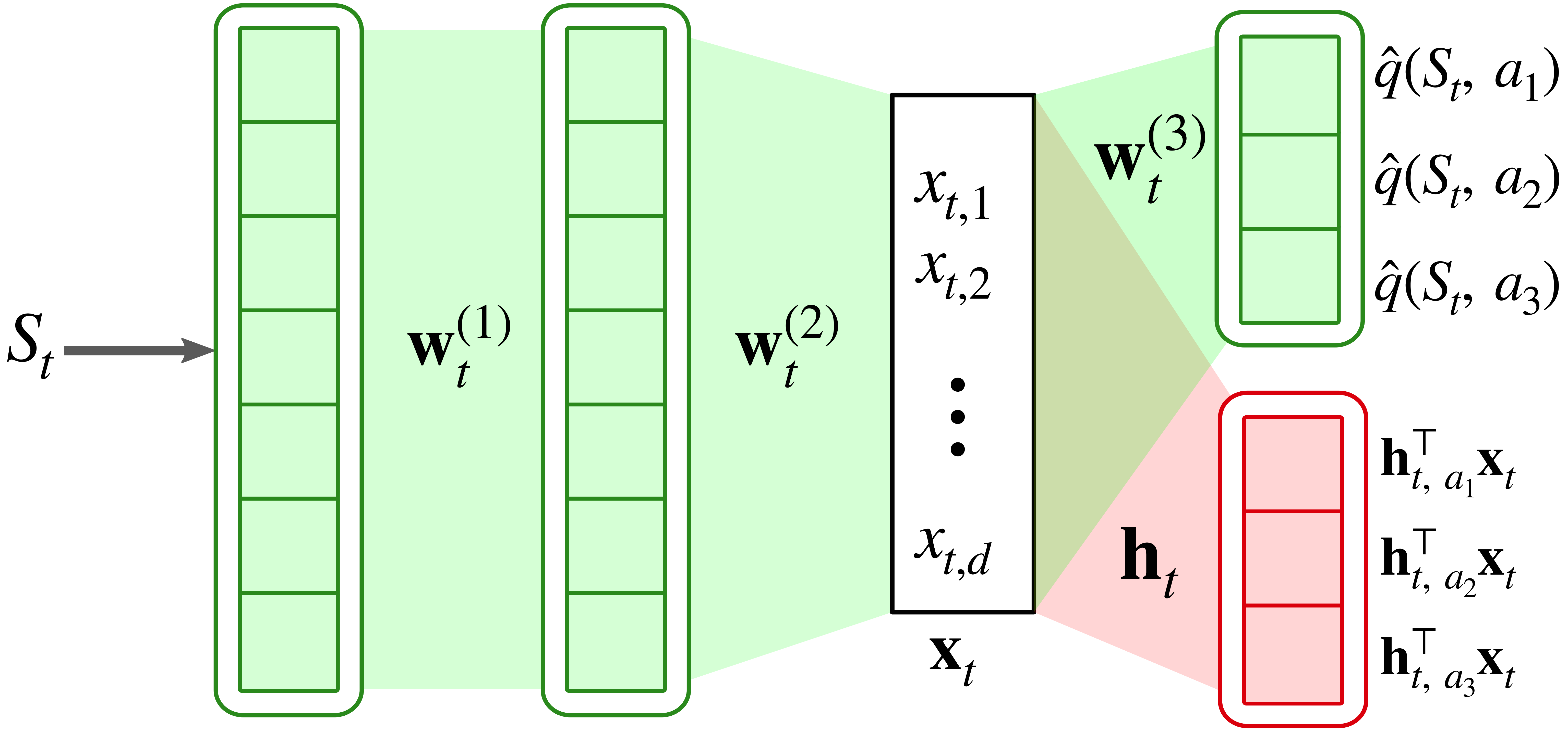
$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta_t \hat{h}(S_t, \mathbf{h}_t)(\nabla \hat{v}(S, \mathbf{w}_t) - \gamma \nabla \hat{v}(S_{t+1}, \mathbf{w}_t))$$

$\hat{h}(S_t, \mathbf{h}_t)$

$$\mathbf{h}_{t+1} \leftarrow \mathbf{h}_t + \eta_t(\delta_t - \hat{h}(S_t, \mathbf{h}_t)) \nabla \hat{h}(S_t, \mathbf{h}_t)$$

The architecture and updates for actions-values

- **Green** part standard TD or Q-learning. **Red** is the added auxiliary variable



Very similar updates

For control we use a TD error with a maximum or soft-max

Once we **approximate** h, no longer minimizing the $\overline{\mathrm{BE}}$.
What are the **ramifications** of **approximating** h?

(And what are we actually minimizing?)

Restricting the Function Space for h Corresponds to a Projection on the Bellman Error

$$\max_{h \in \mathcal{H}} \sum_{s \in \mathcal{S}} d(s) \left(2\mathbb{E}_\pi[\delta \mid S = s] \, h(s) - h(s)^2 \right)$$

...

$$= \|\Pi_{\mathcal{H},d}(\mathcal{T}\hat{v}(\cdot, \boldsymbol{w}) - \hat{v}(\cdot, \boldsymbol{w}))\|_d^2$$

where $\Pi_{\mathcal{H},d} u = \arg\min_{h \in \mathcal{H}} \|u - h\|_d$

$$\|v\|_d^2 = \sum_s d(s) v(s)^2$$

*Assuming $\mathscr{H}$ is a convex space

The Generalized $\overline{\text{PBE}}$

$$\overline{\text{PBE}}(\boldsymbol{w}) \stackrel{\text{def}}{=} \max_{h \in \mathcal{H}} \sum_{s \in \mathcal{S}} d(s) \left(2\mathbb{E}_{\pi}[\delta \mid S = s]\, h(s) - h(s)^2\right)$$

- For $\mathscr{H} = \mathscr{F} =$ a linear function space, this equals the linear $\overline{\text{PBE}}$
- For $\mathscr{H} = \mathscr{F} =$ a nonlinear function space, we get a natural extension of the linear $\overline{\text{PBE}}$ to the nonlinear setting
- For $\mathscr{H} = \mathscr{F}_{\text{all}}$, this equals the Identifiable $\overline{\text{BE}}$
- For $\mathscr{F} \subset \mathscr{H} \subset \mathscr{F}_{\text{all}}$, a Projected Bellman Error between typical $\overline{\text{PBE}}$ and $\overline{\text{BE}}$

Once we **approximate** h, no longer minimizing the $\overline{\text{BE}}$.
What are the **ramifications** of **approximating** h?

(And what are we actually minimizing?)

Approximating h means we are **optimizing the generalized PBE**
(and all is well, things are sound)

But how well does it work?

- Sadly, not that well when using the straightforward gradient update

$$\Delta \boldsymbol{w} \leftarrow h(s)(\nabla_{\boldsymbol{w}} \hat{v}(s, \boldsymbol{w}) - \gamma \nabla_{\boldsymbol{w}} \hat{v}(S', \boldsymbol{w}))$$

- The update relies heavily on having an accurate estimate of h(s)
    - e.g., if the estimate h(s) = 0, the update is zero

A **practical algorithm** using the generalized $\overline{\text{PBE}}$:
Reducing reliance on our estimate h

Sampling the Gradient

- The saddlepoint update

$$\Delta \boldsymbol{w} \leftarrow \ h(s)(\nabla_{\boldsymbol{w}}\hat{v}(s, \boldsymbol{w}) - \gamma \nabla_{\boldsymbol{w}}\hat{v}(S', \boldsymbol{w}))$$

- The gradient-correction update

$$\Delta \boldsymbol{w} \leftarrow \delta(\boldsymbol{w})\nabla_{\boldsymbol{w}} v(s, \boldsymbol{w}) - h(s)\gamma \nabla_{\boldsymbol{w}} v(S', \boldsymbol{w})$$

- Gradient-correction **much more** effective than saddlepoint update

- Notice:

$$\begin{aligned}
&-\mathbb{E}_\pi[\delta(\mathbf{w}) \,|\, S = s]\mathbb{E}_\pi[\nabla \delta(\mathbf{w}) \,|\, S = s] \\
&= \mathbb{E}_\pi[\delta(\mathbf{w}) \,|\, S = s]\mathbb{E}_\pi[\nabla \hat{v}(S, \mathbf{w}) - \gamma \nabla \hat{v}(S', \mathbf{w}) \,|\, S = s] \\
&= \mathbb{E}_\pi[\delta(\mathbf{w}) \,|\, S = s] \nabla \hat{v}(s, \mathbf{w}) - \mathbb{E}_\pi[\delta(\mathbf{w}) \,|\, S = s]\mathbb{E}_\pi[\gamma \nabla \hat{v}(S', \mathbf{w}) \,|\, S = s]
\end{aligned}$$

Sampling the Gradient

- The saddlepoint update

$$\Delta \boldsymbol{w} \leftarrow\ h(s)(\nabla_{\boldsymbol{w}} \hat{v}(s, \boldsymbol{w}) - \gamma \nabla_{\boldsymbol{w}} \hat{v}(S', \boldsymbol{w}))$$

- The gradient-correction update

$$\Delta \boldsymbol{w} \leftarrow \delta(\boldsymbol{w}) \nabla_{\boldsymbol{w}} v(s, \boldsymbol{w}) - h(s) \gamma \nabla_{\boldsymbol{w}} v(S', \boldsymbol{w})$$

- **Point 1**: Gradient-correction **much more** effective than saddlepoint update
- **Point 2**: Regularizing or restricting h significantly improves performance
  - We called the algorithm TD with Regularized Corrections (TDRC) or Q-learning with Regularized Corrections (QRC)
  - Potential reason: corresponds to using a Huber loss

General Strategy for Other Losses

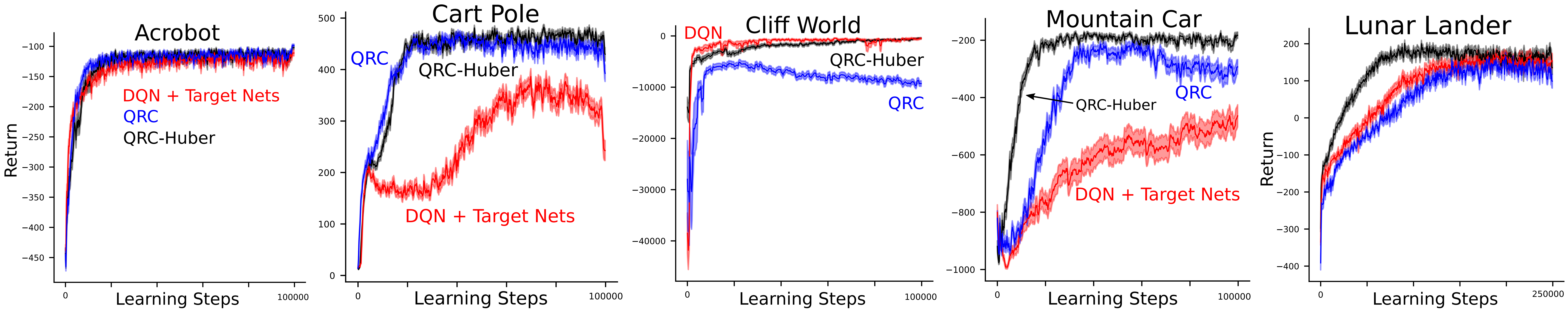
- Example with the Huber loss

$$p_\tau(a) \overset{\text{def}}{=} \begin{cases} a^2 & \text{if } |a| \le \tau \\ 2\tau|a| - \tau^2 & \text{otherwise} \end{cases}$$

$$\mathrm{MHBE}(\theta) \overset{\text{def}}{=} \max_{\boxed{h \in \mathcal{F}_{\text{clip}_\tau}}} \sum_{s \in \mathcal{S}} d(s)\left(2h(s)\mathbb{E}\left[\delta(\theta) \,|\, S = s\right] - h(s)^2\right)$$

$\mathcal{F}_{\text{clip}_\tau}$ the set of all functions $h_{\text{clip}_\tau} : \mathcal{S} \to [-\tau, \tau]$.

Control Experiments



- QRC optimizes squared PBE, without target nets, using gradient corrections
- QRC-Huber is consistently the most effective
- QRC methods generally more stable than DQN, even without target networks

* paper on arXiv: “Robust Losses for Learning Value Functions”

The Key Takeaway: Gradient-based approaches improve on our standard algorithms

- If we use the gradient-corrections form of the update
- If we constrain the auxiliary variable h

Summary of the Talk

- **Point 1:** We can improve on TD and Q-learning
- **Point 2:** Generalized $\overline{\text{PBE}}$ extends the linear $\overline{\text{PBE}}$ to the nonlinear setting and provides a better alternative to the $\overline{\text{BE}}$
- **Point 3:** The resulting gradient algorithms work! We can leverage the literature on linear $\overline{\text{PBE}}$ and $\overline{\text{BE}}$ to get new algorithms (and theory)

Thank you! Questions?