

## Chapter 11

# Off-policy methods with approximation

# Recall off-policy learning involves two policies

- One policy  $\pi$  whose value function we are learning
  - the *target policy*
- Another policy  $\mu$  that is used to select actions
  - the *behavior policy*

# Off-policy is much harder with Function Approximation

- Even linear FA
- Even for prediction (two fixed policies  $\pi$  and  $\mu$ )
- Even for Dynamic Programming
- The deadly triad: FA, TD, off-policy
  - Any two are OK, but not all three
  - With all three, we may get instability (elements of  $\theta$  may increase to  $\pm\infty$ )



# There are really 2 off-policy problems

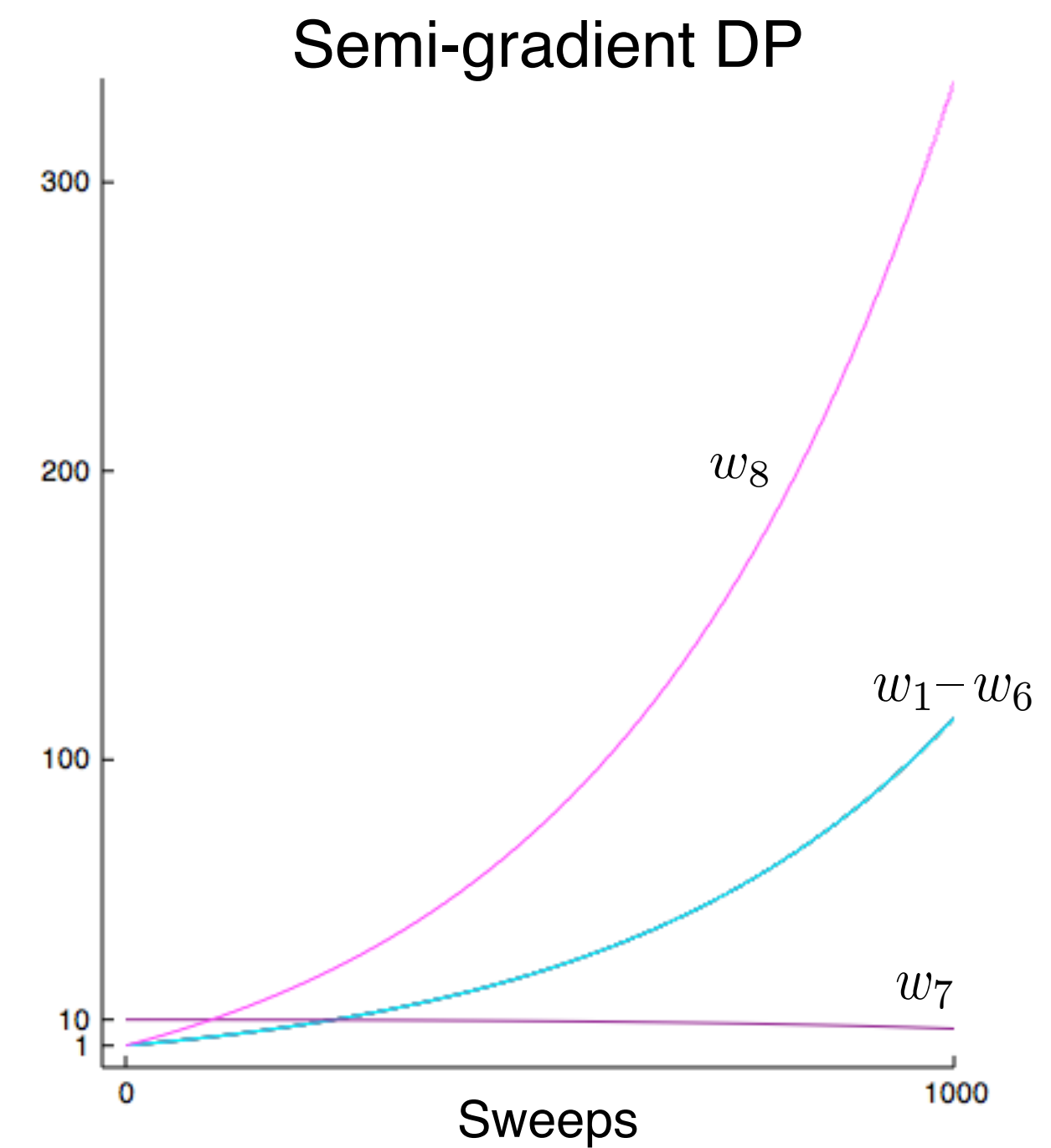
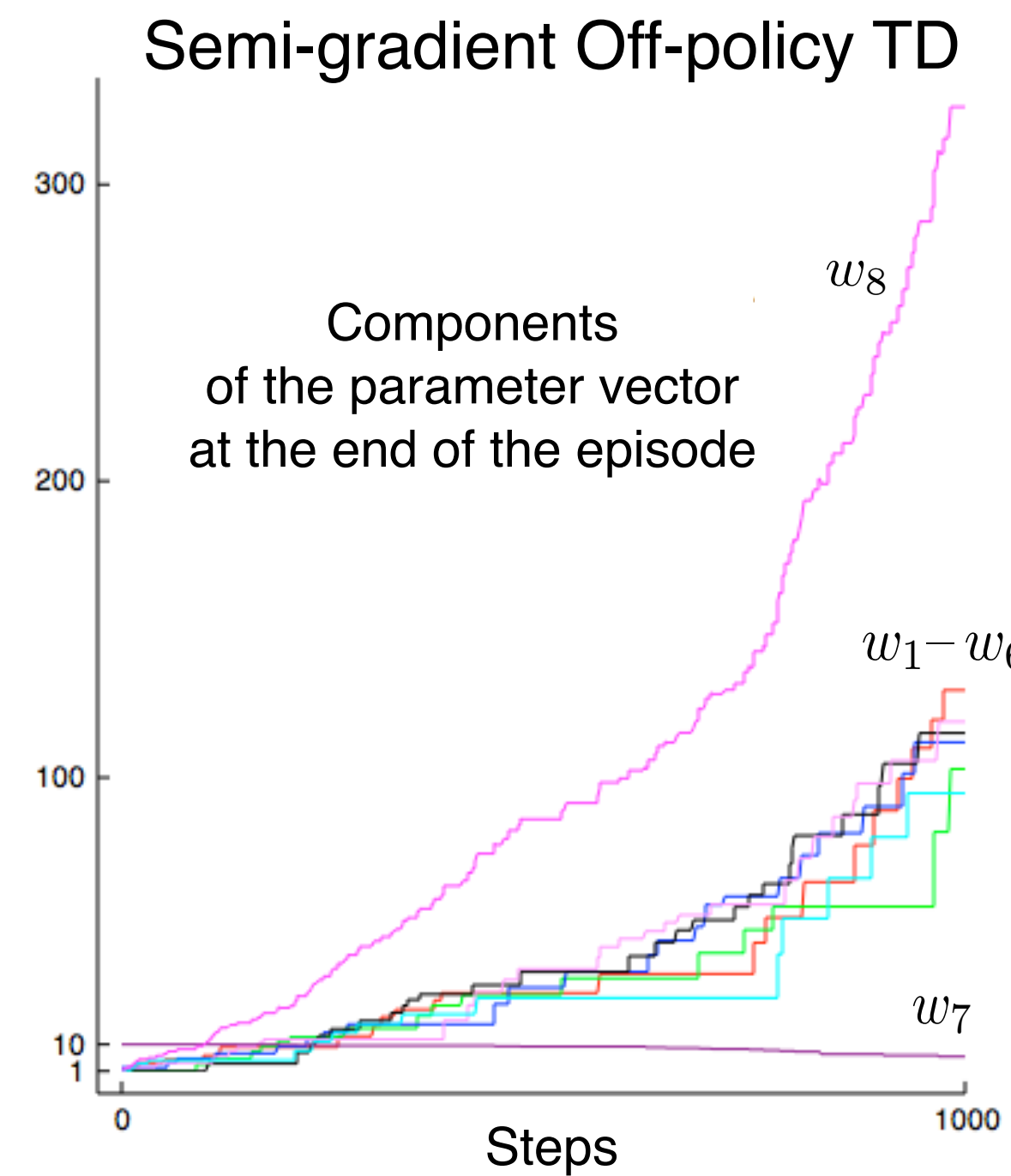
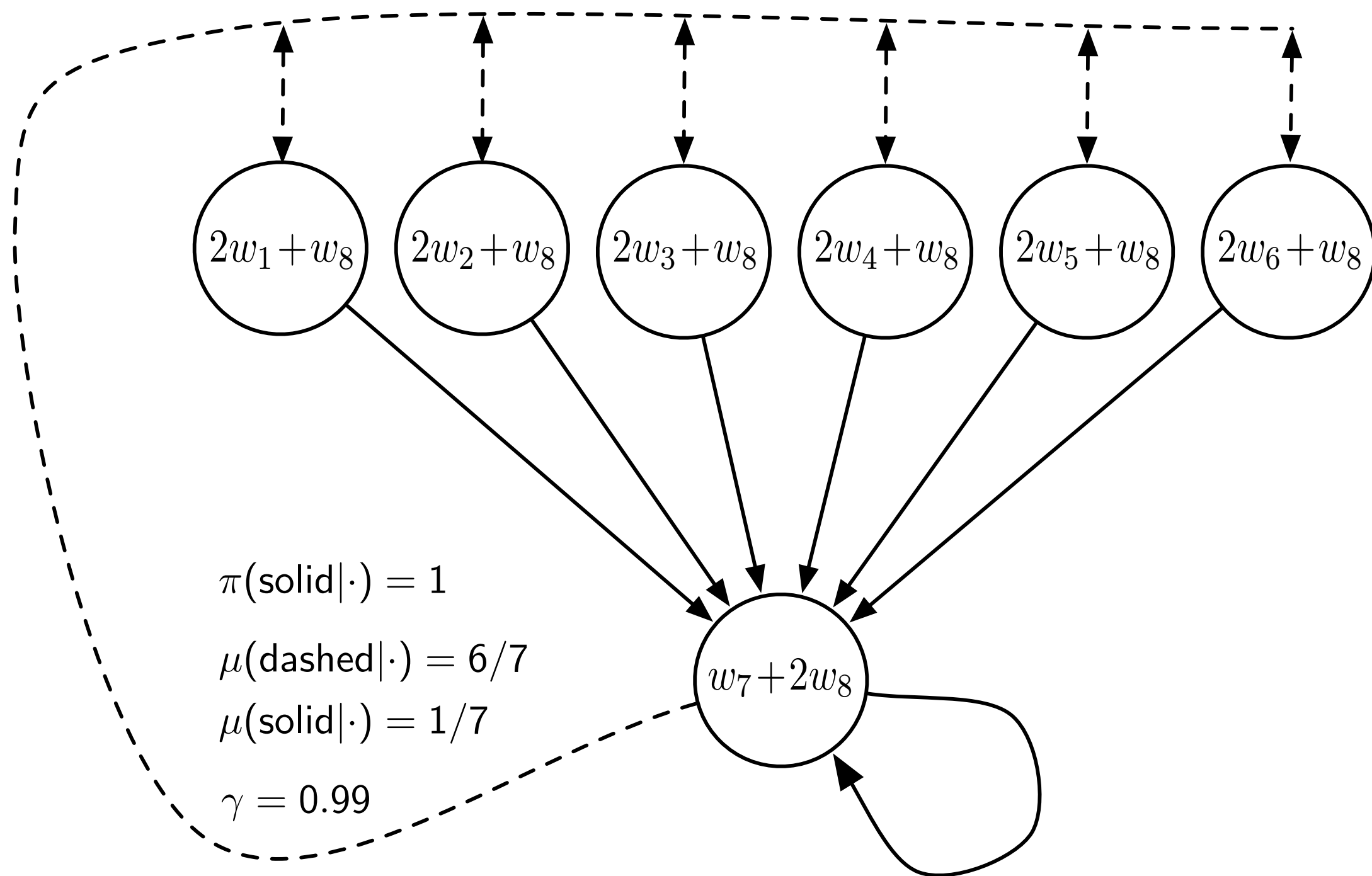
## One we know how to solve, one we are not sure

## One about the future, one about the present

- The easy problem is that of off-policy targets (future)
  - We have been correcting for that since Chapters 5 and 6
  - Using importance sampling in the target
- The hard problem is that of the distribution of states to update (present); we are no longer updating according to the on-policy distribution



# Baird's counterexample illustrates the instability



The parameters don't just fail to converge, they positively diverge!

# What causes the instability?

- It has nothing to do with learning or sampling
  - Even dynamic programming suffers from divergence with FA
- It has nothing to do with exploration, greedification, or control
  - Even prediction alone can diverge
- It has nothing to do with local minima or complex non-linear approximators
  - Even simple linear approximators can produce instability

# The deadly triad

- The risk of divergence arises whenever we combine three things:

## 1. Function approximation

- significantly generalizing from large numbers of examples

## 2. Bootstrapping

- learning value estimates from other value estimates,  
as in dynamic programming and temporal-difference learning

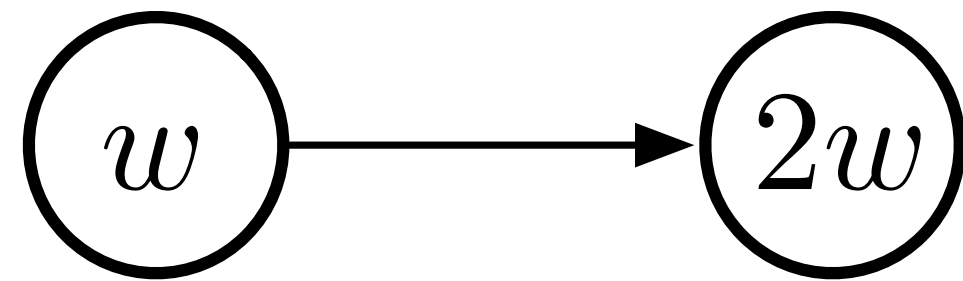
## 3. Off-policy learning (Why is dynamic programming off-policy?)

- learning about a policy from data not due to that policy,  
as in Q-learning, where we learn about the greedy policy from  
data with a necessarily more exploratory policy

Any 2 Ok



# TD(0) can diverge: A simple example



**TD error:**  $\delta_t = R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t) = 0 + \gamma 2w_t - w_t = (2\gamma - 1)w_t$

**TD update:**  $w_{t+1} = w_t + \alpha \rho_t \delta_t \nabla \hat{v}(S_t, w_t) = w_t + \alpha \cdot 1 \cdot (2\gamma - 1)w_t \cdot 1 = \underline{(1 + \alpha(2\gamma - 1))}w_t$

**TD fixpoint:**  $w_{\text{TD}} = 0$

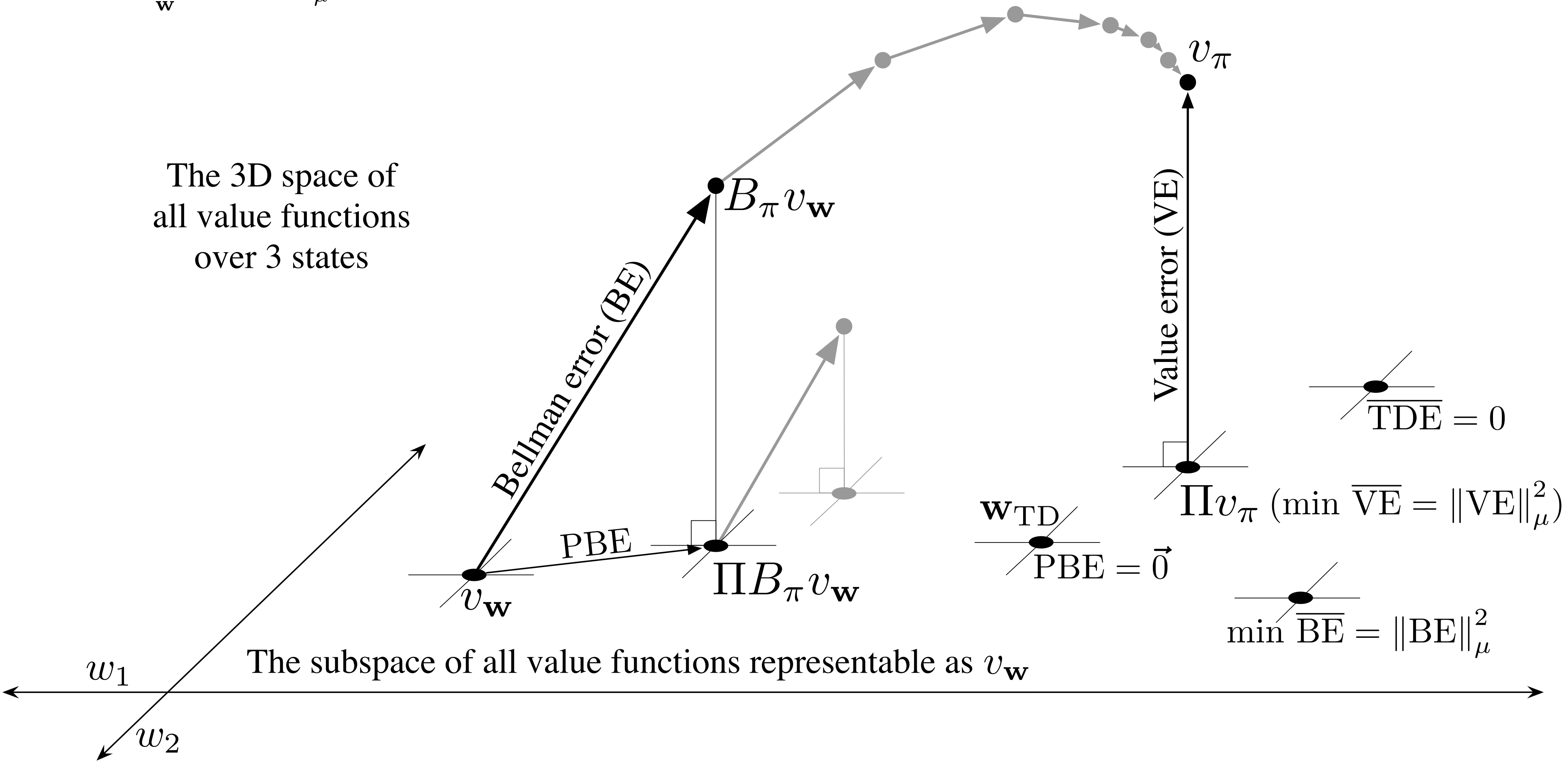
Diverges if  
this is  $> 1$

# Geometric intuition

$v_{\mathbf{w}} \doteq \hat{v}(\cdot, \mathbf{w})$  as a giant vector  $\in \mathbb{R}^{|S|}$

$$(B_{\pi}v)(s) \doteq \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) [r + \gamma v(s')]$$

$$\Pi v \doteq v_{\mathbf{w}} \text{ where } \mathbf{w} = \arg \min_{\mathbf{w}} \|v - v_{\mathbf{w}}\|_{\mu}^2$$



# Can we do without bootstrapping?

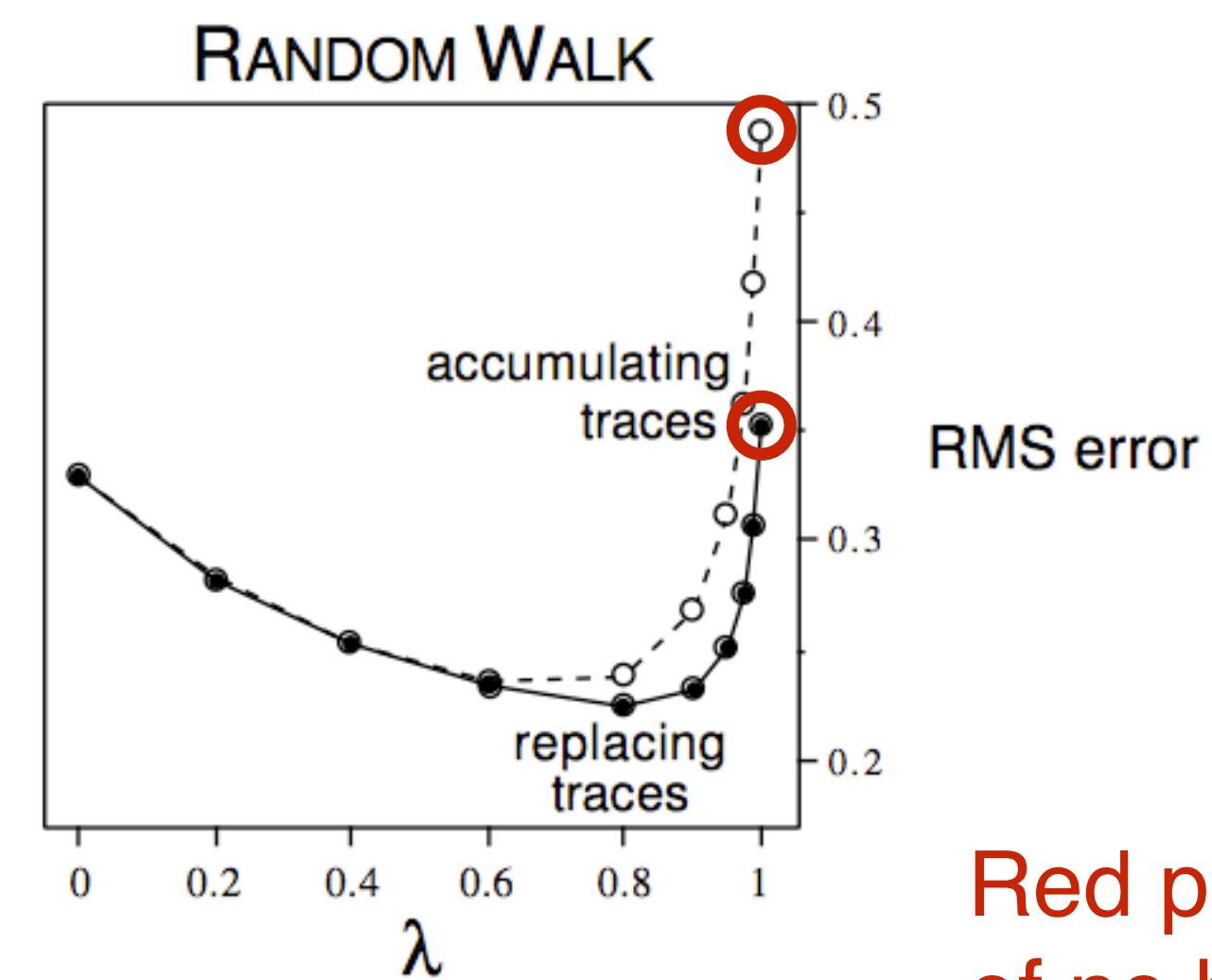
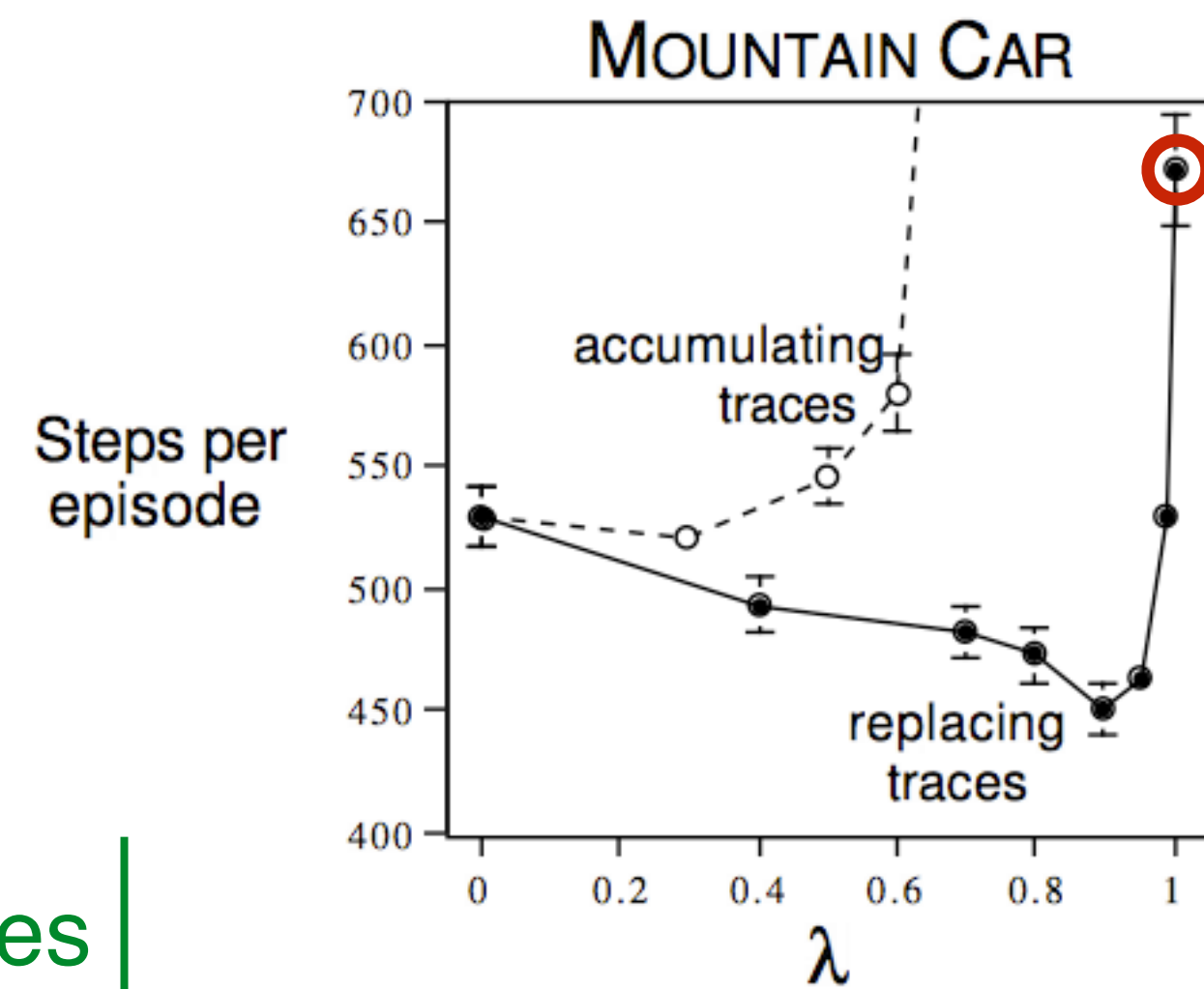
- Bootstrapping is critical to the computational efficiency of DP
- Bootstrapping is critical to the data efficiency of TD methods
- On the other hand, bootstrapping introduces bias, which harms the asymptotic performance of approximate methods
- The degree of bootstrapping can be finely controlled via the  $\lambda$  parameter, from  $\lambda=0$  (full bootstrapping) to  $\lambda=1$  (no bootstrapping)



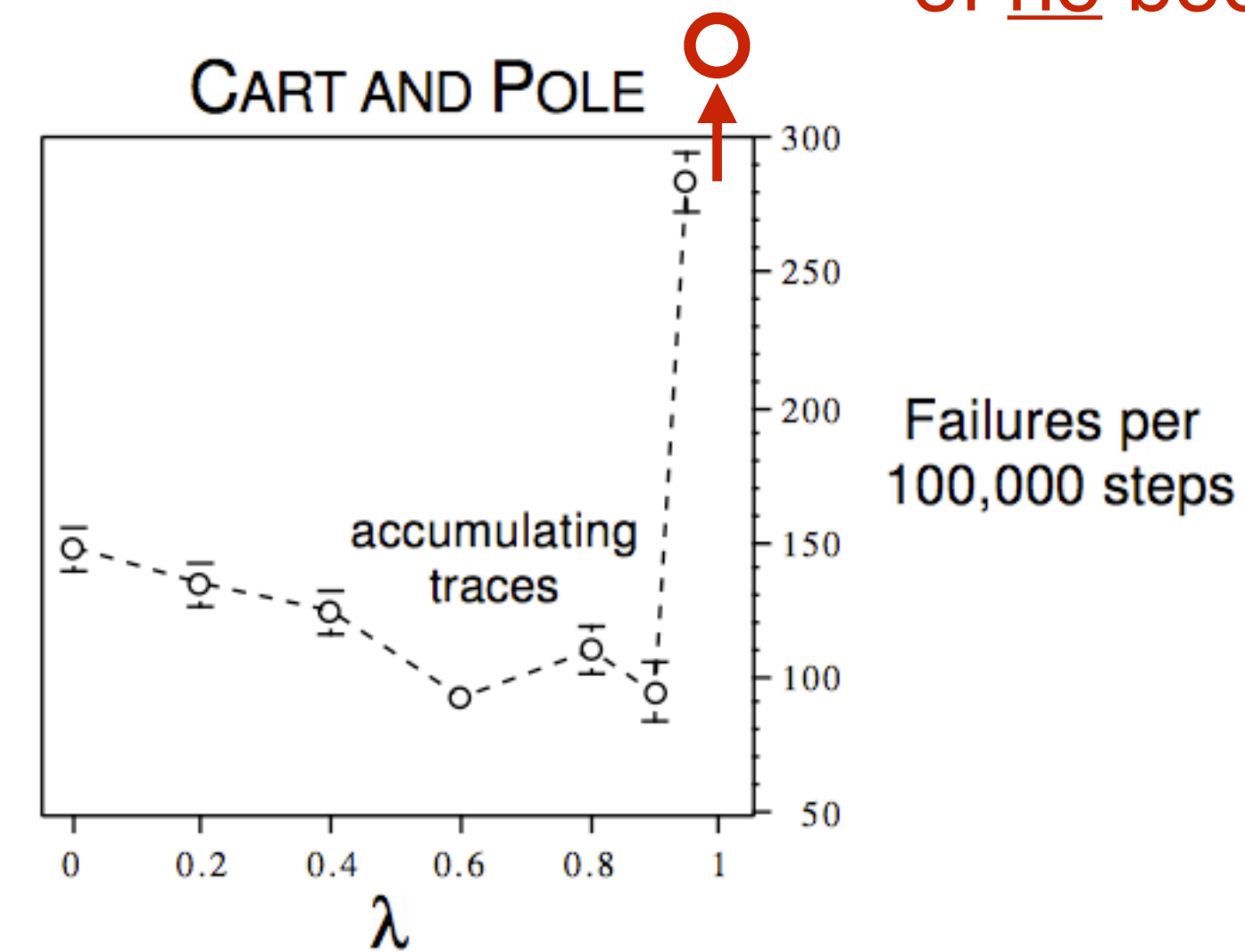
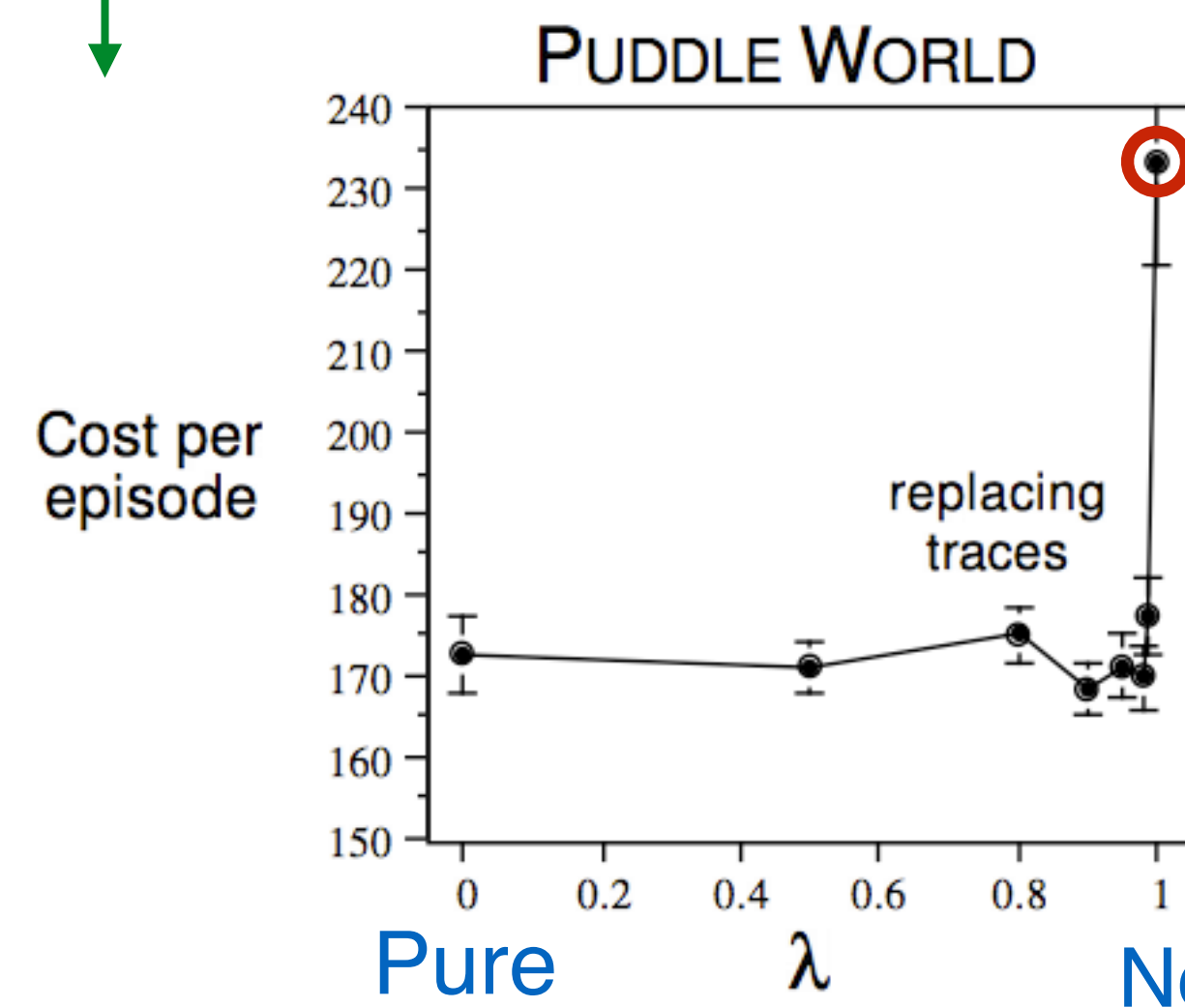
# 4 examples of the effect of bootstrapping

suggest that  $\lambda=1$  (no bootstrapping) is a very poor choice

In all cases  
lower is better



Red points are the cases  
of no bootstrapping



Pure bootstrapping      No bootstrapping

We need bootstrapping!

# Desiderata: We want a TD algorithm that

- Bootstraps (genuine TD)
- Works with linear function approximation (stable, reliably convergent)
- Is simple, like linear TD —  $O(n)$
- Learns fast, like linear TD
- Can learn off-policy
- Learns from online causal trajectories (no repeat sampling from the same state)



# 4 easy steps to stochastic gradient descent

1. Pick an objective function  $J(\mathbf{w})$ ,  
a parameterized function to be minimized
2. Use calculus to analytically compute the gradient  $\nabla_{\mathbf{w}} J(\mathbf{w})$
3. Find a “sample gradient”  $\nabla_{\mathbf{w}} J_t(\mathbf{w})$  that you can sample on  
*every time step* and whose *expected value equals the gradient*
4. Take small steps in  $\mathbf{w}$  proportional to the sample gradient:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla_{\mathbf{w}} J_t(\mathbf{w}_t)$$

# Could TD be the gradient of some J?

TD(0) algorithm:  $\Delta \mathbf{w} = \alpha (R + \gamma \mathbf{w}^\top \mathbf{x}' - \mathbf{w}^\top \mathbf{x}) \mathbf{x}$

Assume there is a J such that:  $\frac{\partial J}{\partial w_i} = \alpha (R + \gamma \mathbf{w}^\top \mathbf{x}' - \mathbf{w}^\top \mathbf{x}) x_i$

Then look at the second derivative:

$$\frac{\partial^2 J}{\partial w_j \partial w_i} \stackrel{?}{=} \frac{\partial^2 J}{\partial w_i \partial w_j}$$

2<sup>nd</sup> derivative matrices must be symmetric

# Could TD be the gradient of some J?

TD(0) algorithm:  $\Delta \mathbf{w} = \alpha (R + \gamma \mathbf{w}^\top \mathbf{x}' - \mathbf{w}^\top \mathbf{x}) \mathbf{x}$

Assume there is a J such that:  $\frac{\partial J}{\partial w_i} = \alpha (R + \gamma \mathbf{w}^\top \mathbf{x}' - \mathbf{w}^\top \mathbf{x}) x_i$

Then look at the second derivative:

$$\frac{\partial^2 J}{\partial w_j \partial w_i} \neq \frac{\partial^2 J}{\partial w_i \partial w_j}$$

Contradiction!

2<sup>nd</sup> derivative matrices must be symmetric



# Could TD be the gradient of some J?

**TD(0) algorithm:**  $\Delta \mathbf{w} = \alpha (R + \gamma \mathbf{w}^\top \mathbf{x}' - \mathbf{w}^\top \mathbf{x}) \mathbf{x}$

**Assume there is a J such that:**  $\frac{\partial J}{\partial w_i} = \alpha (R + \gamma \mathbf{w}^\top \mathbf{x}' - \mathbf{w}^\top \mathbf{x}) x_i$

**Then look at the second derivative:**

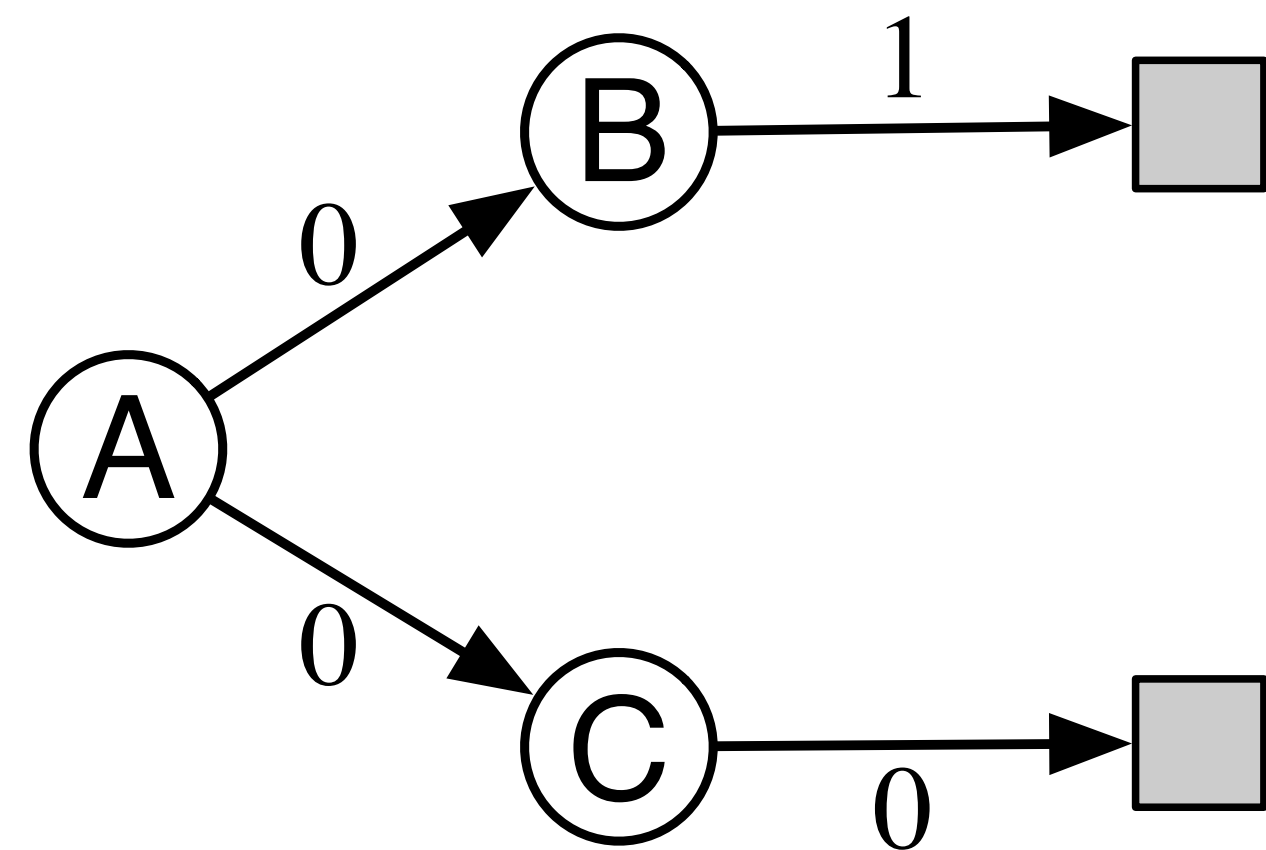
$$\left. \begin{aligned} \frac{\partial^2 J}{\partial w_j \partial w_i} &= \frac{\partial(\alpha(R + \gamma \mathbf{w}^\top \mathbf{x}' - \mathbf{w}^\top \mathbf{x})x_i)}{\partial w_j} = \alpha(\gamma x'_j - x_j)x_i \\ \frac{\partial^2 J}{\partial w_i \partial w_j} &= \frac{\partial(\alpha(R + \gamma \mathbf{w}^\top \mathbf{x}' - \mathbf{w}^\top \mathbf{x})x_i)}{\partial w_i} = \alpha(\gamma x'_i - x_i)x_j \end{aligned} \right\} \frac{\partial^2 J}{\partial w_j \partial w_i} \neq \frac{\partial^2 J}{\partial w_i \partial w_j}$$

**Contradiction!**

**2<sup>nd</sup> derivative matrices must be symmetric**

**Conventional TD is not the gradient of anything**

# A-split example (Dayan 1992)



$$v_{\pi}(A) = \frac{1}{2}$$

- What are the true values of **B** and **C**?

$$v_{\pi}(B) = 1$$

$$v_{\pi}(C) = 0$$

TD will find these values

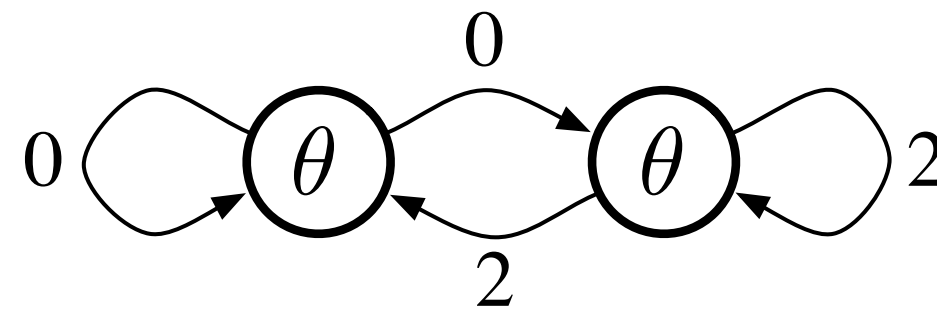
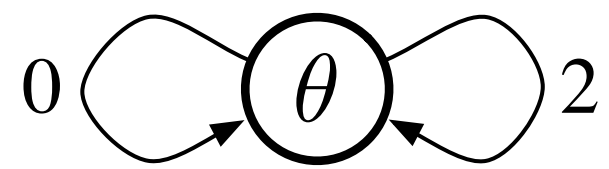
- What values minimize the squared TD error?

$$v(B) = \frac{3}{4}$$

$$v(C) = \frac{1}{4}$$

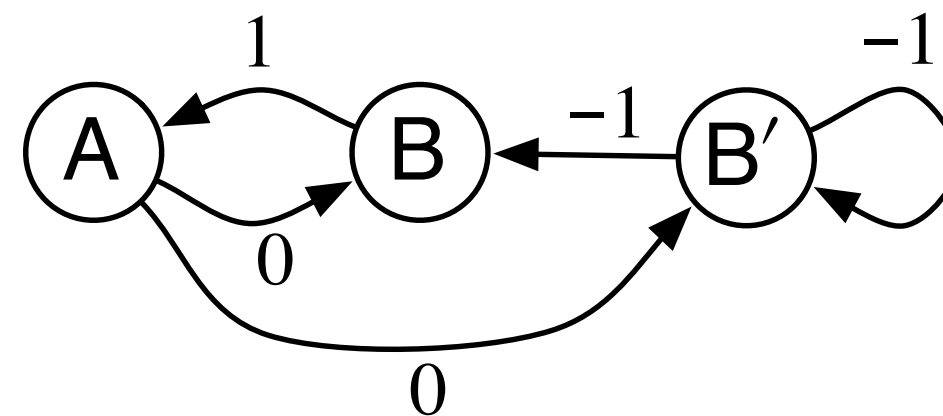
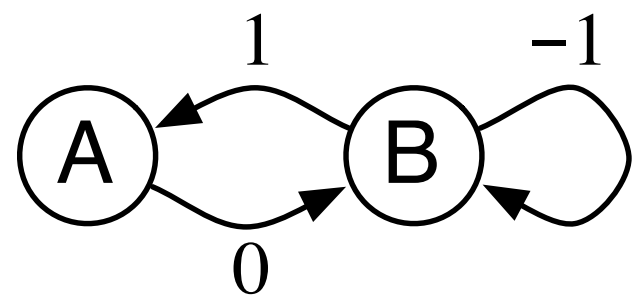
RG will find these values

# Indistinguishable pairs of MDPs



These two have different Value Errors, but the same Return Errors (both errors have the same minima)

$$\overline{\text{RE}}(\mathbf{w}) = \mathbb{E}[(G_t - \hat{v}(S_t, \mathbf{w}))^2] = \overline{\text{VE}}(\mathbf{w}) + \mathbb{E}[(G_t - v_\pi(S_t))^2]$$

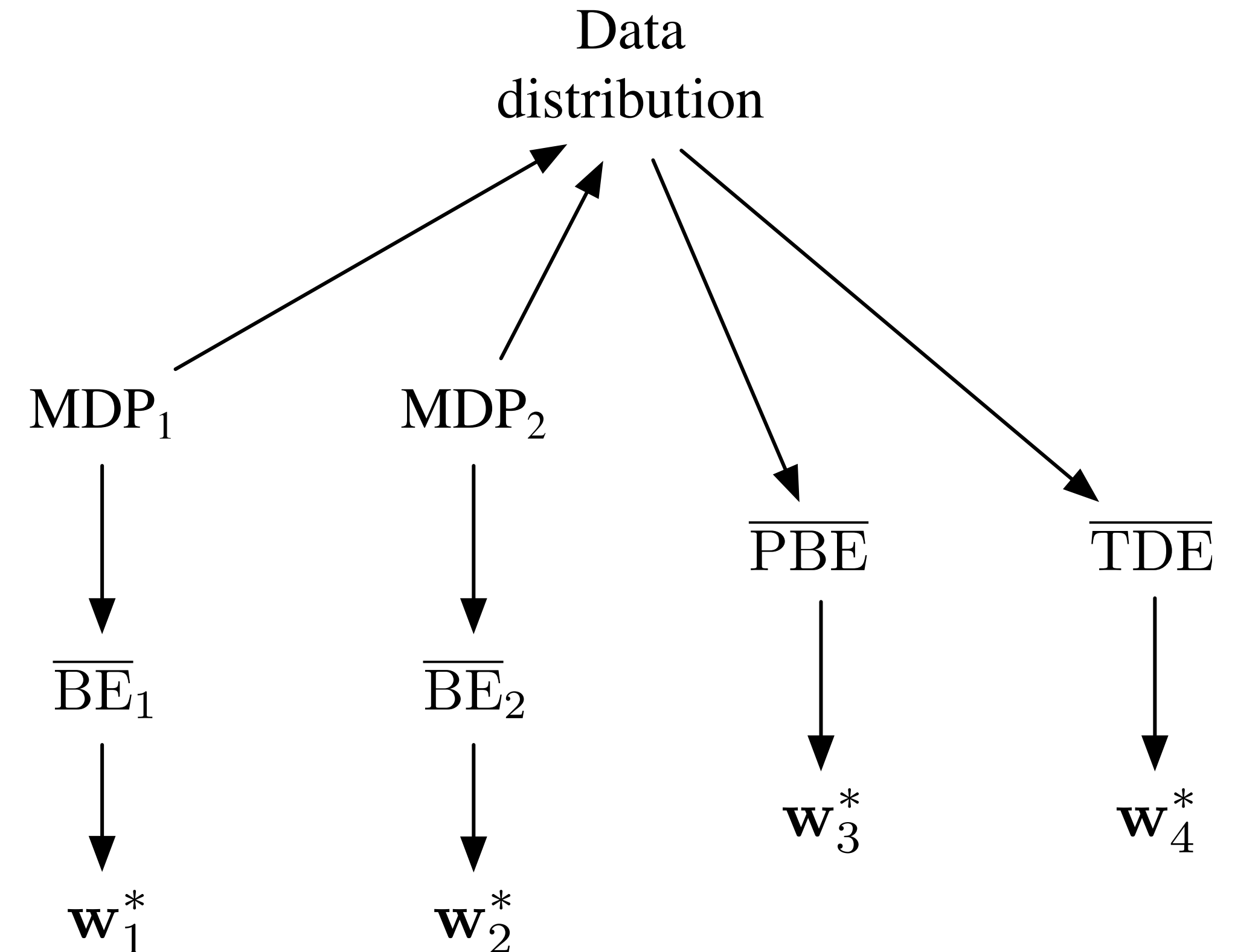
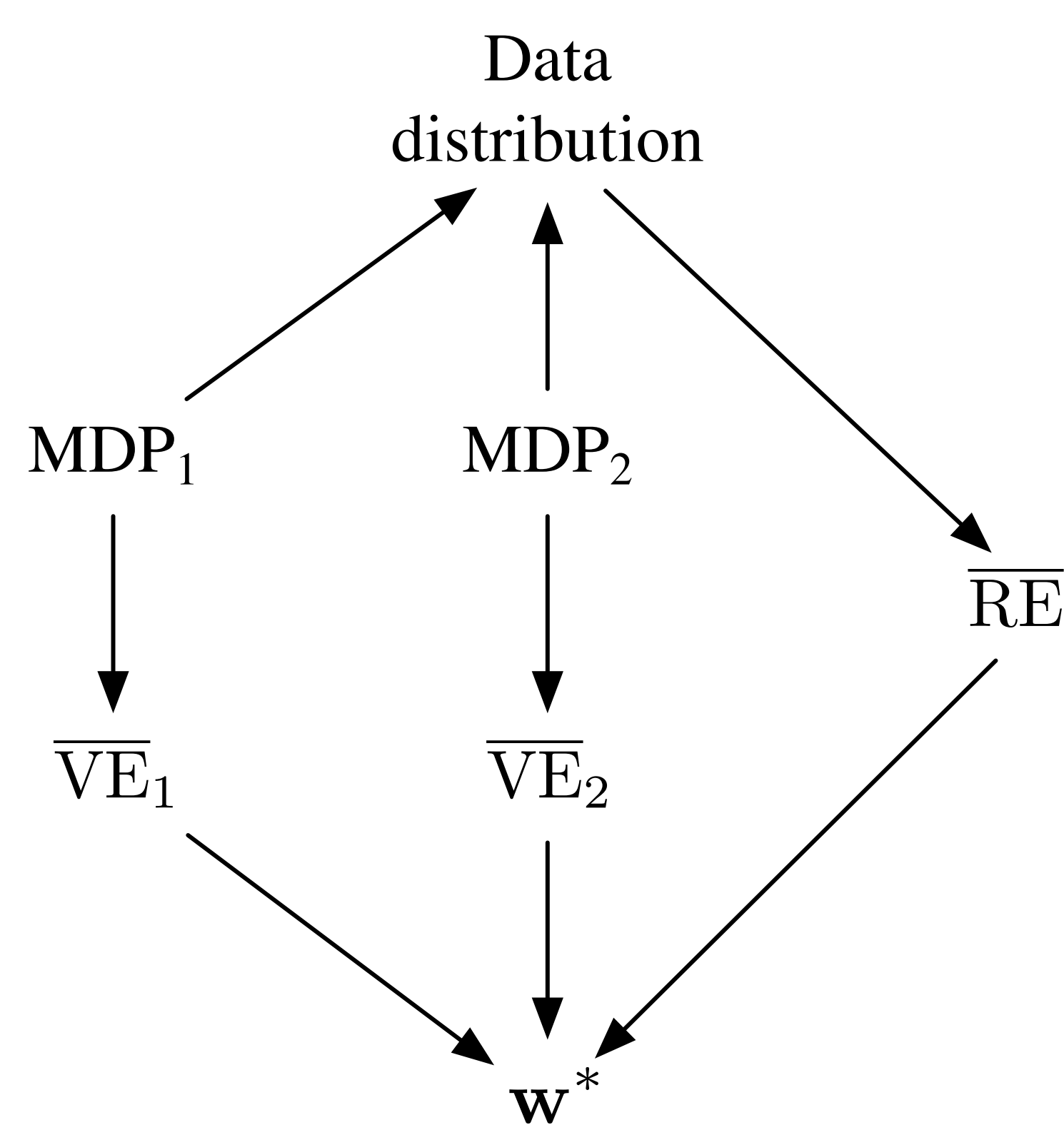


These two have different Bellman Errors, but the same Projected Bellman Errors (the errors have different minima)



# Not all objectives can be estimated from data

# Not all minima can be found by learning



No learning algorithm can find the minimum of the Bellman Error



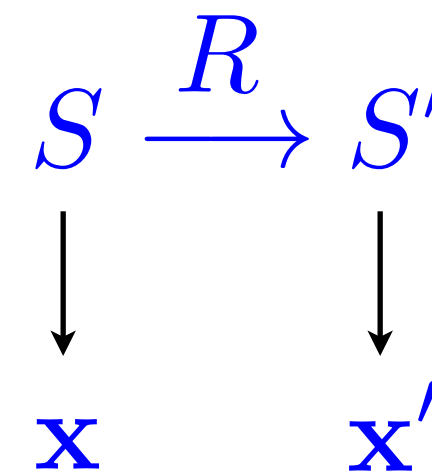
# The Gradient-TD Family of Algorithms

- True gradient-descent algorithms in the Projected Bellman Error
- GTD( $\lambda$ ) and GQ( $\lambda$ ), for learning state values and action values
- Solve two open problems:
  - convergent linear-complexity off-policy TD learning
  - convergent non-linear TD
- Extended to control variate, proximal forms by Mahadevan et al.

# *TD with gradient correction (TDC) algorithm*

aka GTD(0)

- on each transition



- update two parameters

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \delta \mathbf{x} - \alpha \gamma \mathbf{x} (\mathbf{x}^\top \mathbf{v})$$

$$\mathbf{v} \leftarrow \mathbf{v} + \beta (\delta - \mathbf{x}^\top \mathbf{v}) \mathbf{x}$$

- where, as usual

$$\delta \doteq R + \gamma \mathbf{w}^\top \mathbf{x}' - \mathbf{w}^\top \mathbf{x}$$

with gradient  
correction

estimate of the  
TD error ( $\delta$ ) for  
the current state  $\mathbf{x}$

# Convergence theorems

- All algorithms converge w.p.1 to the TD fix-point:

$$\mathbb{E}[\delta \mathbf{x}] \longrightarrow 0$$

- GTD, GTD-2 converges at one time scale

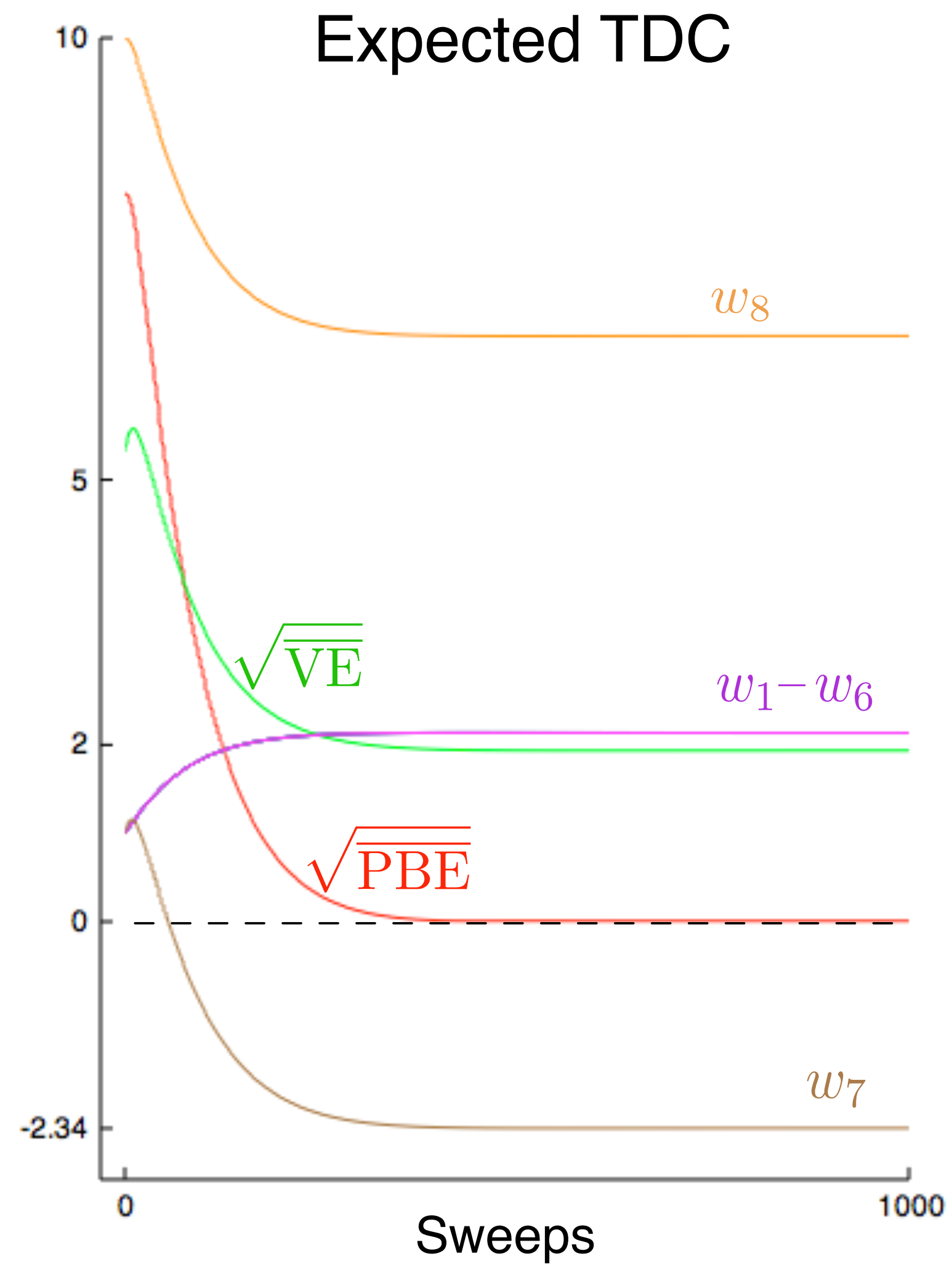
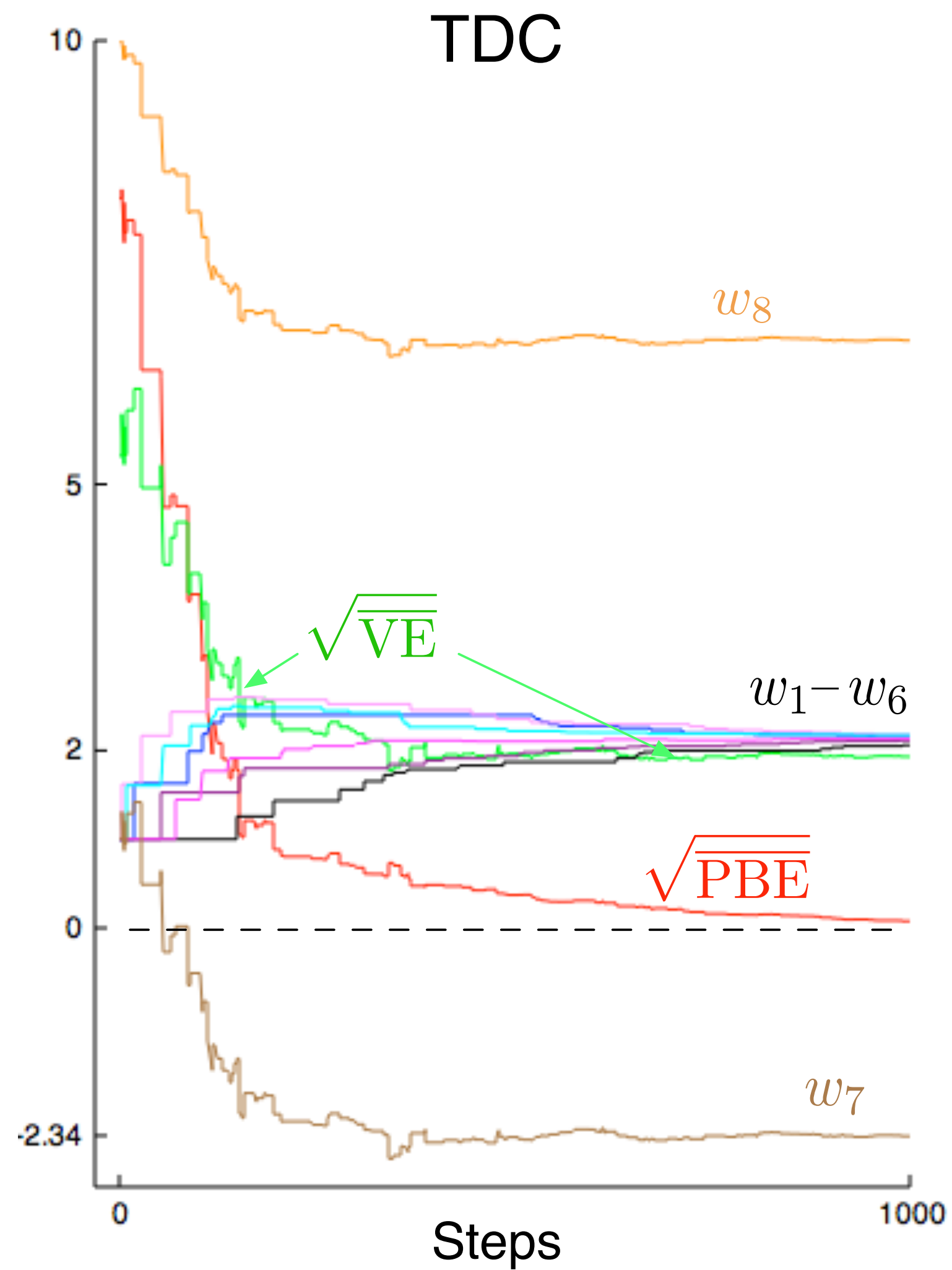
$$\alpha = \beta \longrightarrow 0$$

- TD-C converges in a two-time-scale sense

$$\alpha, \beta \longrightarrow 0 \quad \frac{\alpha}{\beta} \longrightarrow 0$$



# Gradient-TD algorithms converge on Baird's counter-example

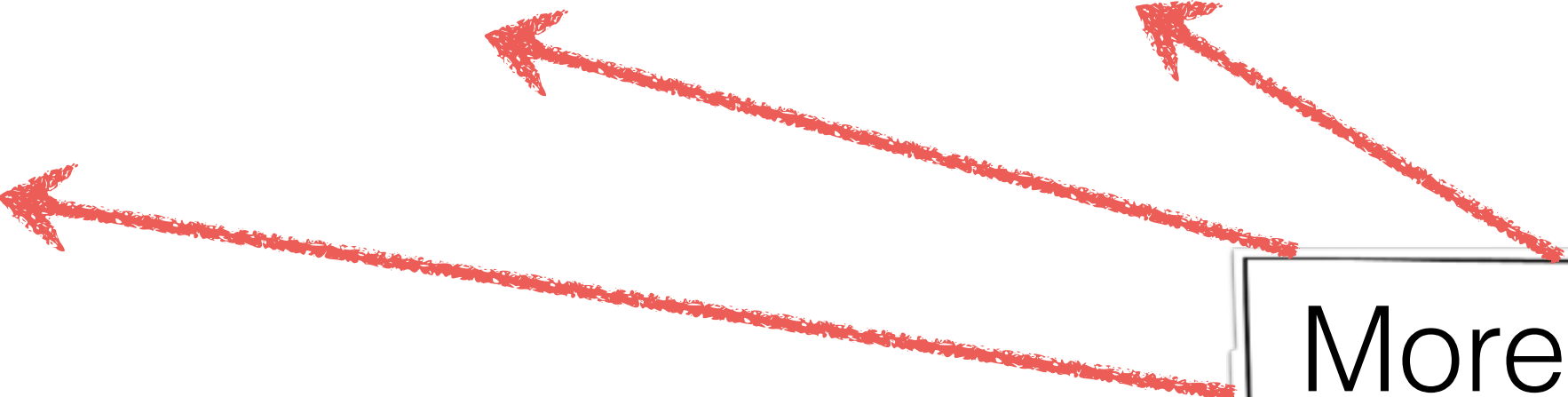


|       |                          | ALGORITHM                              |               |                                         |          |                      |     |                                      |
|-------|--------------------------|----------------------------------------|---------------|-----------------------------------------|----------|----------------------|-----|--------------------------------------|
|       |                          | TD( $\lambda$ ),<br>Sarsa( $\lambda$ ) | Approx.<br>DP | LSTD( $\lambda$ ),<br>LSPE( $\lambda$ ) | Fitted-Q | Residual<br>gradient | GDP | GTD( $\lambda$ ),<br>GQ( $\lambda$ ) |
| ISSUE | Linear<br>computation    | ✓                                      | ✓             | ✗                                       | ✗        | ✓                    | ✓   | ✓                                    |
|       | Nonlinear<br>convergent  | ✗                                      | ✗             | ✗                                       | ✓        | ✓                    | ✓   | ✓                                    |
|       | Off-policy<br>convergent | ✗                                      | ✗             | ✓                                       | ✗        | ✓                    | ✓   | ✓                                    |
|       | Model-free,<br>online    | ✓                                      | ✗             | ✓                                       | ✗        | ✓                    | ✗   | ✓                                    |
|       | Converges to<br>PBE = 0  | ✓                                      | ✓             | ✓                                       | ✓        | ✗                    | ✓   | ✓                                    |

In conclusion

# Off-policy RL with FA and TD remains challenging; there are multiple ideas, plus combinations

- Gradient TD, proximal gradient TD, and hybrids
- Emphatic TD
- Higher  $\lambda$  (less TD)
- Better state rep'ns (less FA)
- Recognizers (less off-policy)
- LSTD ( $O(n^2)$  methods)



More work needed  
on these novel algs!





# Emphatic temporal-difference learning

Rupam Mahmood, Huizhen (Janey) Yu, Martha White, Rich Sutton

Reinforcement Learning and Artificial Intelligence Laboratory  
Department of Computing Science  
University of Alberta  
Canada



State weightings are important,  
powerful, even magical,  
when using “genuine function approximation”  
(i.e., when the optimal solution can’t be approached)

- They are the difference between convergence and divergence in on-policy and off-policy TD learning
- They are needed to make the problem well-defined
- We can change the weighting by *emphasizing* some steps more than others in learning



# Often some time steps are more important

- Early time steps of an *episode* may be more important
- Because of *discounting*
- Because the control objective is to maximize the value of the *starting state*
- In general, function approximation resources are limited
  - Not all states can be accurately valued
  - The accuracy of different state must be traded off!
  - You may want to control the tradeoff

# Bootstrapping interacts with state importance

- In the Monte Carlo case ( $\lambda=1$ ) the values of different states (or time steps) are estimated independently, and their importances can be assigned independently
- But with bootstrapping ( $\lambda<1$ ) each state's value is estimated based on the estimated values of later states; if the state is important, then it becomes important to accurately value the later states even if they are not important on their own

# Two kinds of importance

- Intrinsic and derived, primary and secondary
  - The one you specify, and the one that follows from it because of bootstrapping
- Our terms: *Interest* and *Emphasis*
  - Your intrinsic *interest* in valuing accurately on a time step
  - The total resultant *emphasis* that you place on each time step



## Problem

- Data

$$\cdots \phi(S_t) A_t R_{t+1} \overset{\substack{\phi : \mathcal{S} \rightarrow \mathbb{R}^n \\ \text{feature function}}}{\phi(S_{t+1})} A_{t+1} R_{t+2} \cdots$$

- State distribution

$$d_\mu(s) = \lim_{t \rightarrow \infty} \Pr[S_t = s \mid A_{0:t-1} \sim \overset{\substack{\text{behavior policy} \\ \mu}}{\mu}]$$

- Objective to minimize

$$\text{MSE}(\overset{\substack{\text{parameter vector} \\ \boldsymbol{\theta}}}{\boldsymbol{\theta}}) = \sum_{s \in \mathcal{S}} d_\mu(s) \overset{\substack{\text{interest function} \\ i : \mathcal{S} \rightarrow \mathbb{R}^+}}{i(s)} \left( \overset{\substack{\text{true value function} \\ v_\pi(s)}}{v_\pi(s)} - \boldsymbol{\theta}^\top \overset{\substack{\text{target policy} \\ \phi(s)}}{\phi(s)} \right)^2$$

$\text{transpose (inner product)}$

- Emphatic TD(0)

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \overset{\substack{\text{emphasis} \\ M_t > 0}}{M_t} \overset{\substack{\text{importance sampling ratio} \\ \rho_t = \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} \quad \mathbb{E}[\rho_t] = 1}}{\rho_t} (R_{t+1} + \gamma \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_{t+1} - \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_t) \boldsymbol{\phi}_t$$

## Solution

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}(S_t)$$

## Problem

$$\cdots \phi(S_t) A_t R_{t+1} \phi(S_{t+1}) A_{t+1} R_{t+2} \cdots$$

- State distribution

$$d_\mu(s) = \lim_{t \rightarrow \infty} \Pr[S_t = s \mid A_{0:t-1} \sim \mu]$$

behavior policy  
↓

- Objective to minimize

$$\text{MSE}(\boldsymbol{\theta}) = \sum_{s \in \mathcal{S}} d_\mu(s) \underset{\substack{\text{interest function} \\ i: \mathcal{S} \rightarrow \mathbb{R}^+}}{i(s)} \left( \underset{\substack{\text{true value function} \\ \text{target policy}}}{v_\pi(s)} - \underset{\substack{\text{parameter vector}}}{\boldsymbol{\theta}}^\top \underset{\substack{\text{transpose} \\ \text{(inner product)}}}{\phi(s)} \right)^2$$

- Emphatic TD(0)

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \underset{\substack{\text{emphasis} \\ M_t > 0}}{M_t} \underset{\substack{\text{importance sampling ratio}}}{\rho_t} (R_{t+1} + \gamma \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_{t+1} - \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_t) \boldsymbol{\phi}_t$$

$$\rho_t = \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} \quad \mathbb{E}[\rho_t] = 1$$

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}(S_t)$$

## Solution

- Emphatic LSTD(0)

$$\mathbf{A}_t = \sum_{k=0}^t \boldsymbol{\phi}_k (\boldsymbol{\phi}_k - \gamma \boldsymbol{\phi}_{k+1})^\top \quad \mathbf{b}_t = \sum_{k=1}^t M_k \rho_k R_k \boldsymbol{\phi}_k$$

$$\boldsymbol{\theta}_{t+1} = \mathbf{A}_t^{-1} \mathbf{b}_t$$

# Emphasis algorithm

(Sutton, Mahmood & White 2015)

- Derived from analysis of general bootstrapping relationships (Sutton, Mahmood, Precup & van Hasselt 2014)

- Emphasis is a scalar signal  $M_t \geq 0$

$$M_t = \lambda_t i(S_t) + (1 - \lambda_t) F_t$$

- Defined from a new scalar *followon trace*  $F_t \geq 0$

$$F_t = \rho_{t-1} \gamma_t F_{t-1} + i(S_t)$$



# Off-policy implications

- The emphasis weighting is *stable under off-policy TD( $\lambda$ )* (like the on-policy weighting) (Sutton, Mahmood & White 2015)
- It is the *followon* weighting, from the interest weighted behavior distribution ( $d_\mu(s)i(s)$ ), under the target policy
- Learning is *convergent* (though not necessarily of finite variance) under the emphasis weighting for arbitrary target and behavior policies (with coverage) (Yu 2015)
- There are error bounds analogous to those for on-policy TD( $\lambda$ ) (Munos)
- Emphatic TD is the simplest convergent off-policy TD algorithm (one parameter, one learning rate)