Chapter 11

Off-policy methods with approximation

Recall off-policy learning involves two policies

- One policy π whose value function we are learning
  - the *target policy*
- Another policy μ that is used to select actions
  - the *behavior policy*

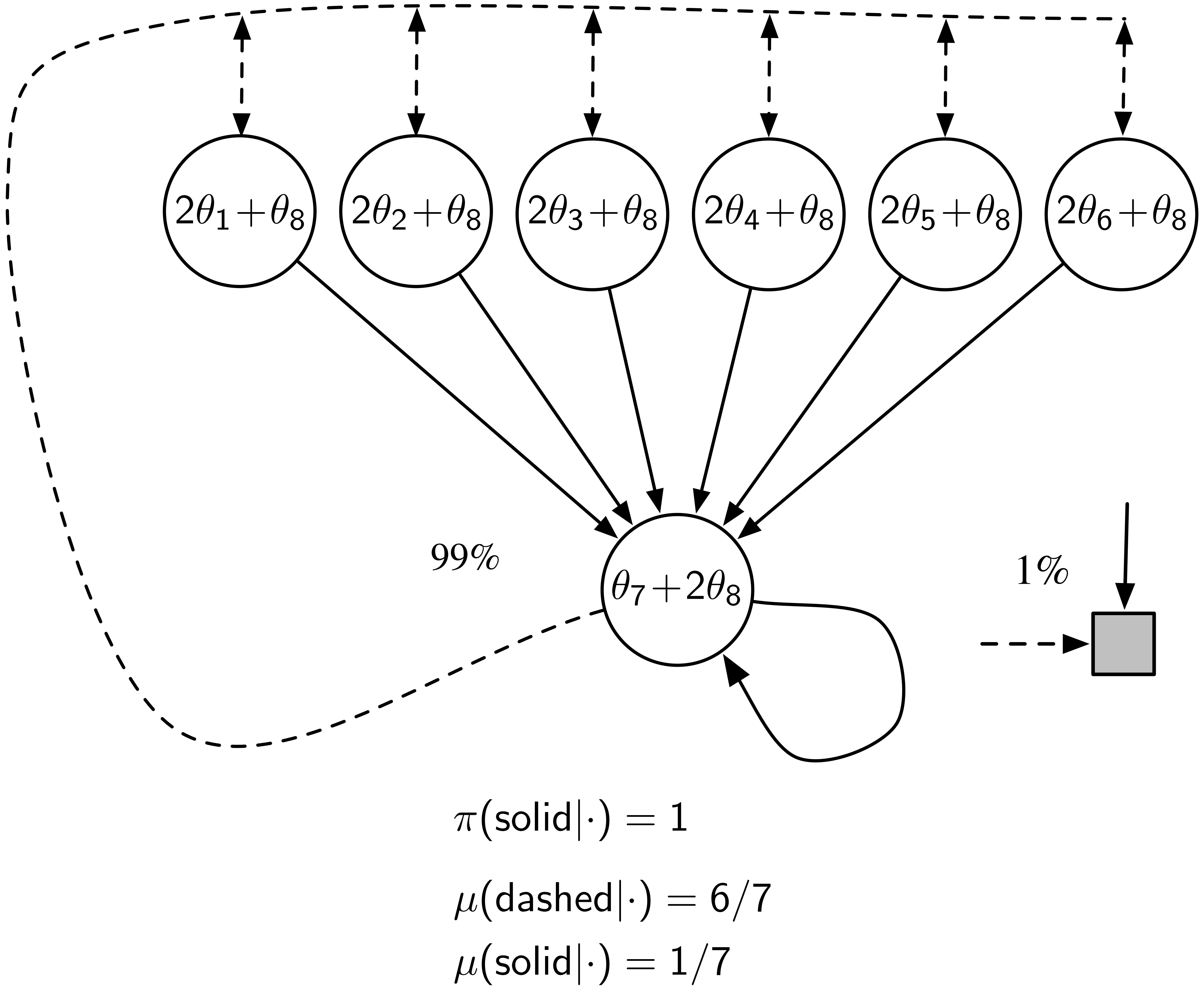
Off-policy is much harder with Function Approximation

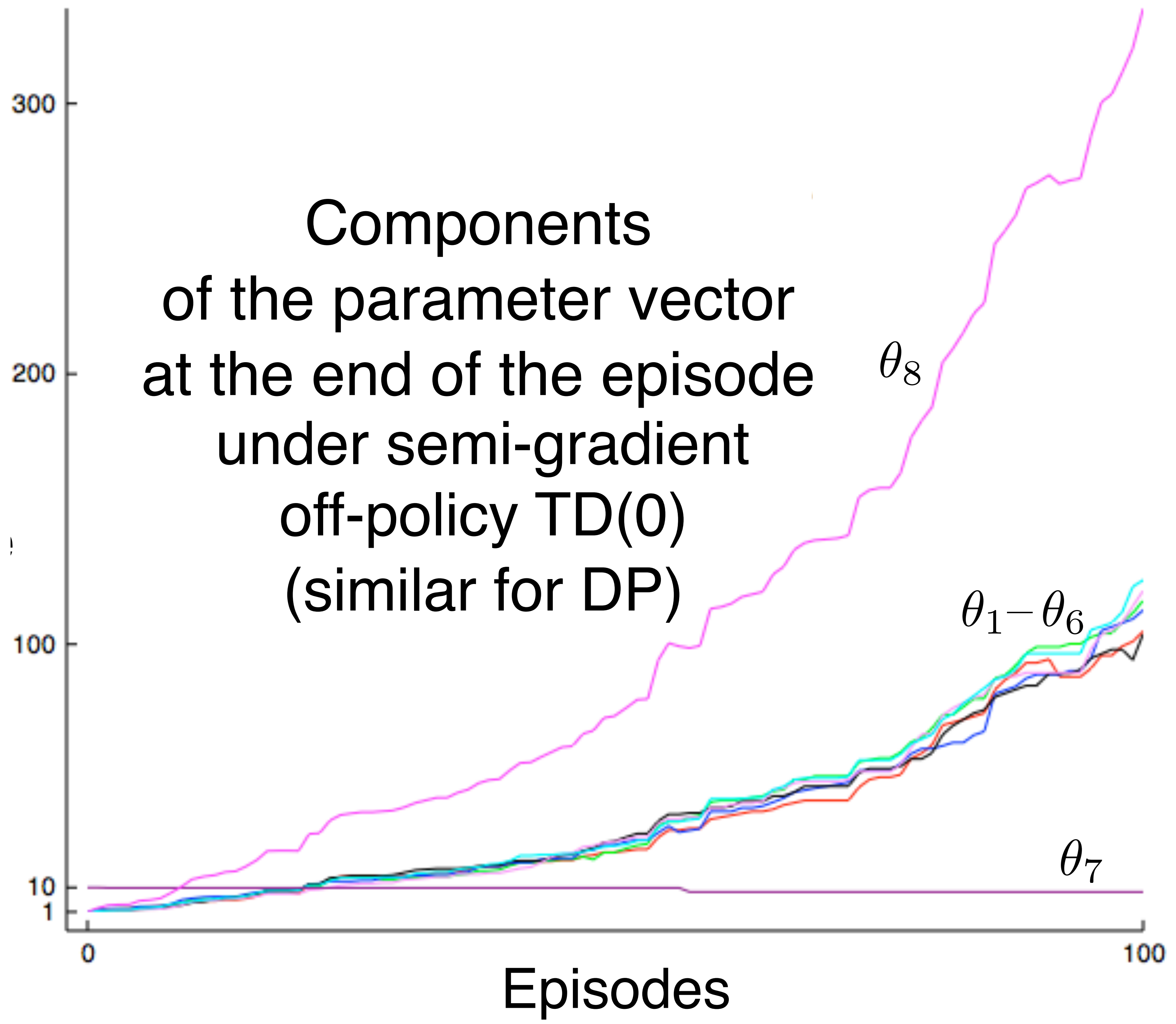
- Even linear FA
- Even for prediction (two fixed policies π and μ)
- Even for Dynamic Programming
- The deadly triad: FA, TD, off-policy
  - Any two are OK, but not all three
  - With all three, we may get instability (elements of $\boldsymbol{\theta}$ may increase to $\pm\infty$)

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





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

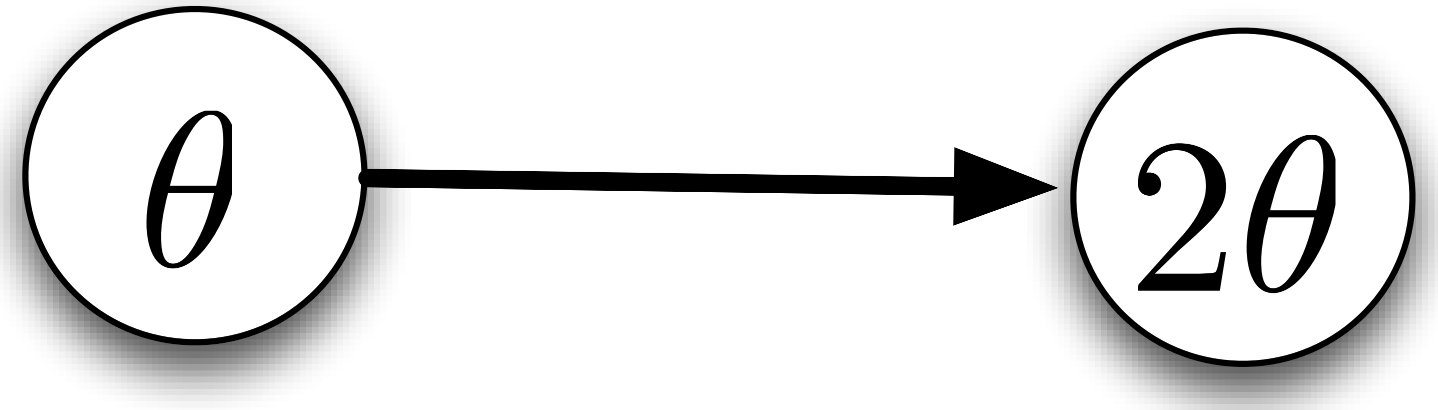
        - learning value estimates from other value estimates, as in dynamic programming and temporal-difference learning

    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



$$\begin{aligned}\delta &= r + \gamma\theta^\top\phi' - \theta^\top\phi \\ &= 0 + 2\theta - \theta \\ &= \theta\end{aligned}$$

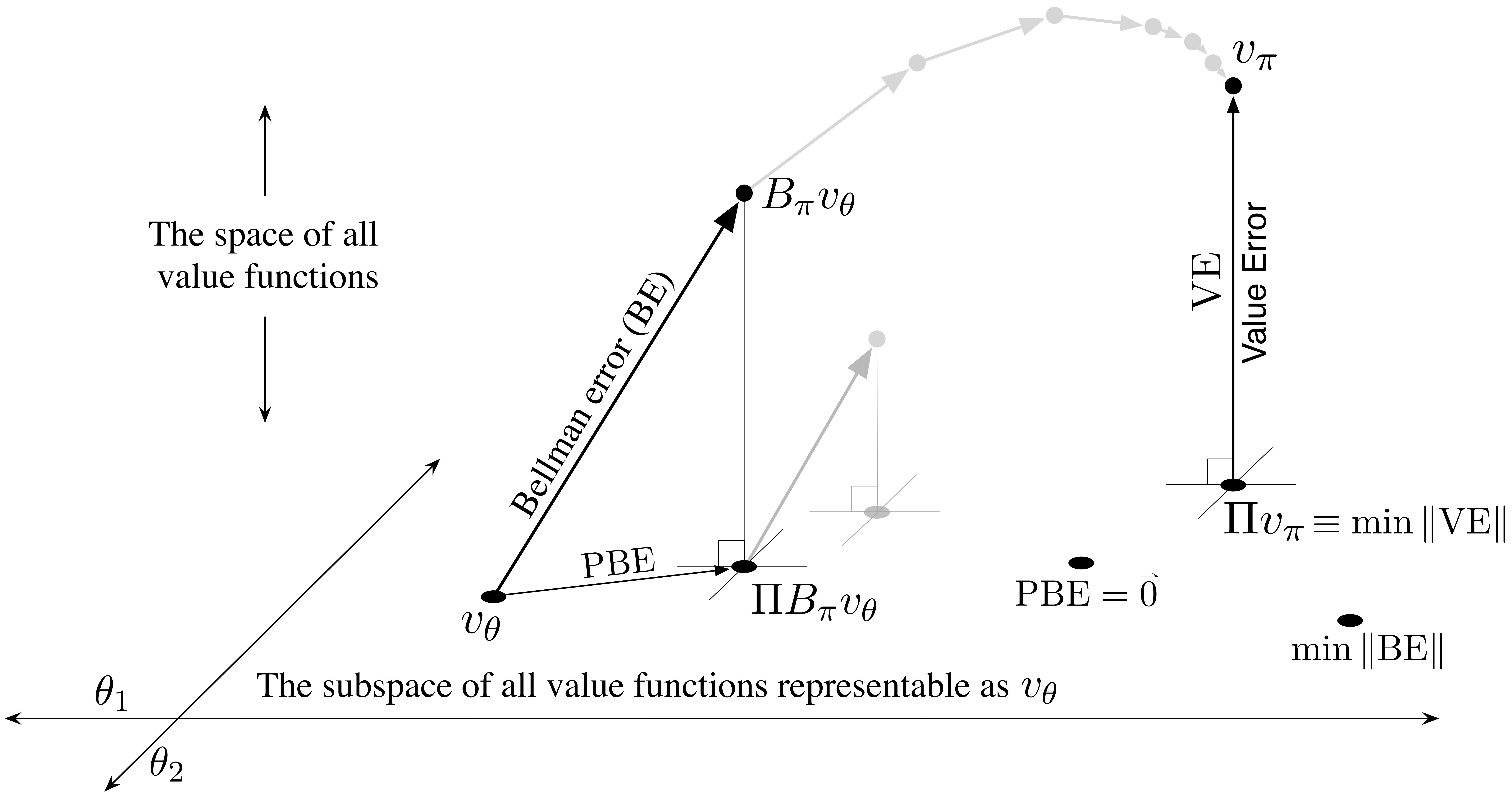
TD update: $$\begin{aligned}\Delta\theta &= \alpha\delta\phi \\ &= \alpha\theta\end{aligned}$$ Diverges!

TD fixpoint: $$\theta^* = 0$$

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

Geometric intuition

$$(B_\pi v)(s) \doteq \sum_{a \in \mathcal{A}} \pi(s,a) \left[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v(s') \right]$$

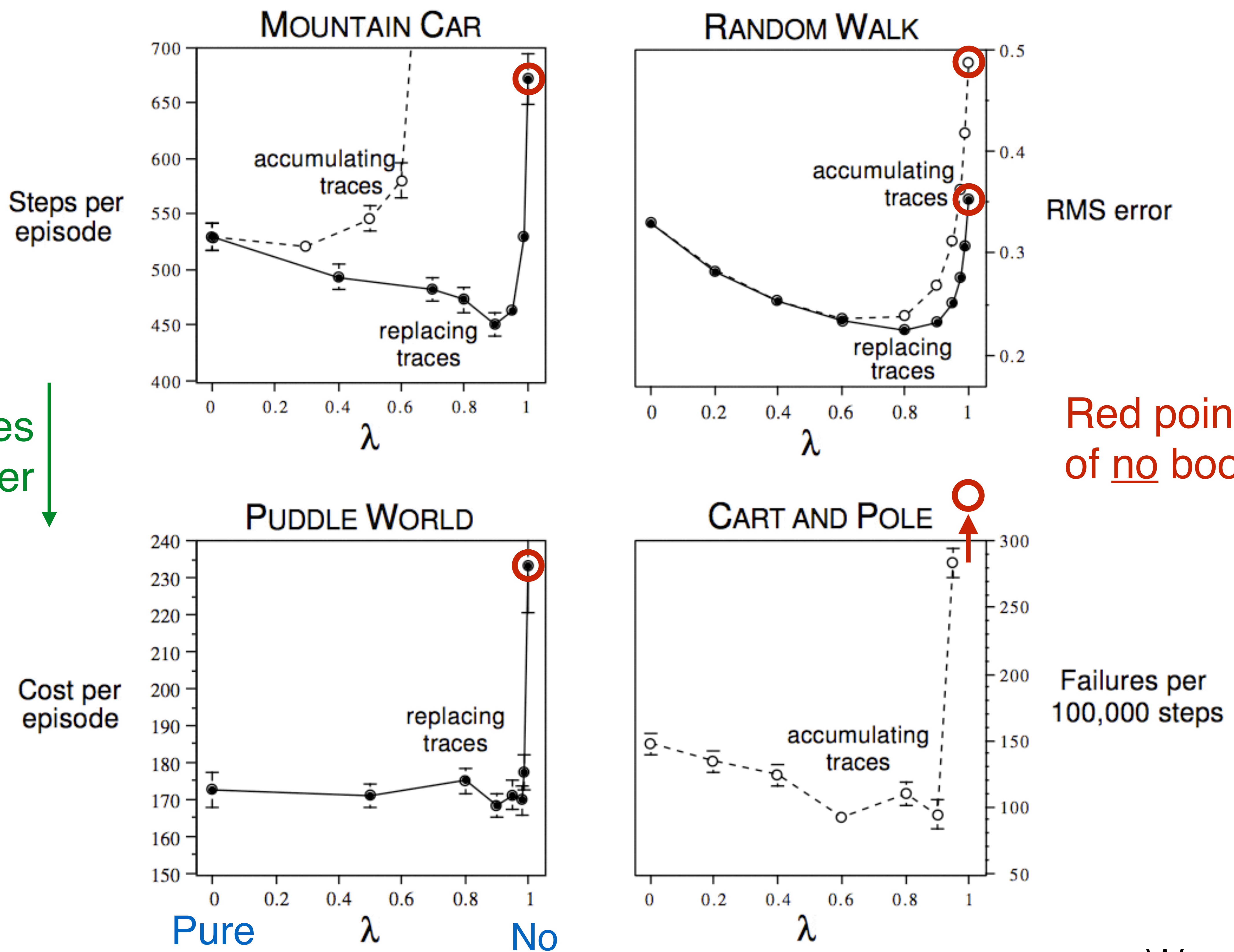


Can we do without bootstrapping?

- 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 λ parameter, from $\lambda=0$ (full bootstrapping) to $\lambda=1$ (no bootstrapping)

4 examples of the effect of bootstrapping

suggest that λ=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 $J(\theta)$,
   a parameterized function to be minimized

2. Use calculus to analytically compute the gradient $\nabla_\theta J(\theta)$

3. Find a "sample gradient" $\nabla_\theta J_t(\theta)$ that you can sample on every time step and whose expected value equals the gradient

4. Take small steps in θ proportional to the sample gradient:

$$\theta \leftarrow \theta - \alpha \nabla_\theta J_t(\theta)$$

Conventional TD is not the gradient of anything

TD(0) algorithm:

$$\Delta\theta = \alpha\delta\phi$$

$$\delta = r + \gamma\theta^\top\phi' - \theta^\top\phi$$

Assume there is a J such that: $\frac{\partial J}{\partial \theta_i} = \delta\phi_i$

Then look at the second derivative:

$$\frac{\partial^2 J}{\partial\theta_j \partial\theta_i} = \frac{\partial(\delta\phi_i)}{\partial\theta_j} = (\gamma\phi'_j - \phi_j)\phi_i$$

$$\frac{\partial^2 J}{\partial\theta_i \partial\theta_j} = \frac{\partial(\delta\phi_j)}{\partial\theta_i} = (\gamma\phi'_i - \phi_i)\phi_j$$

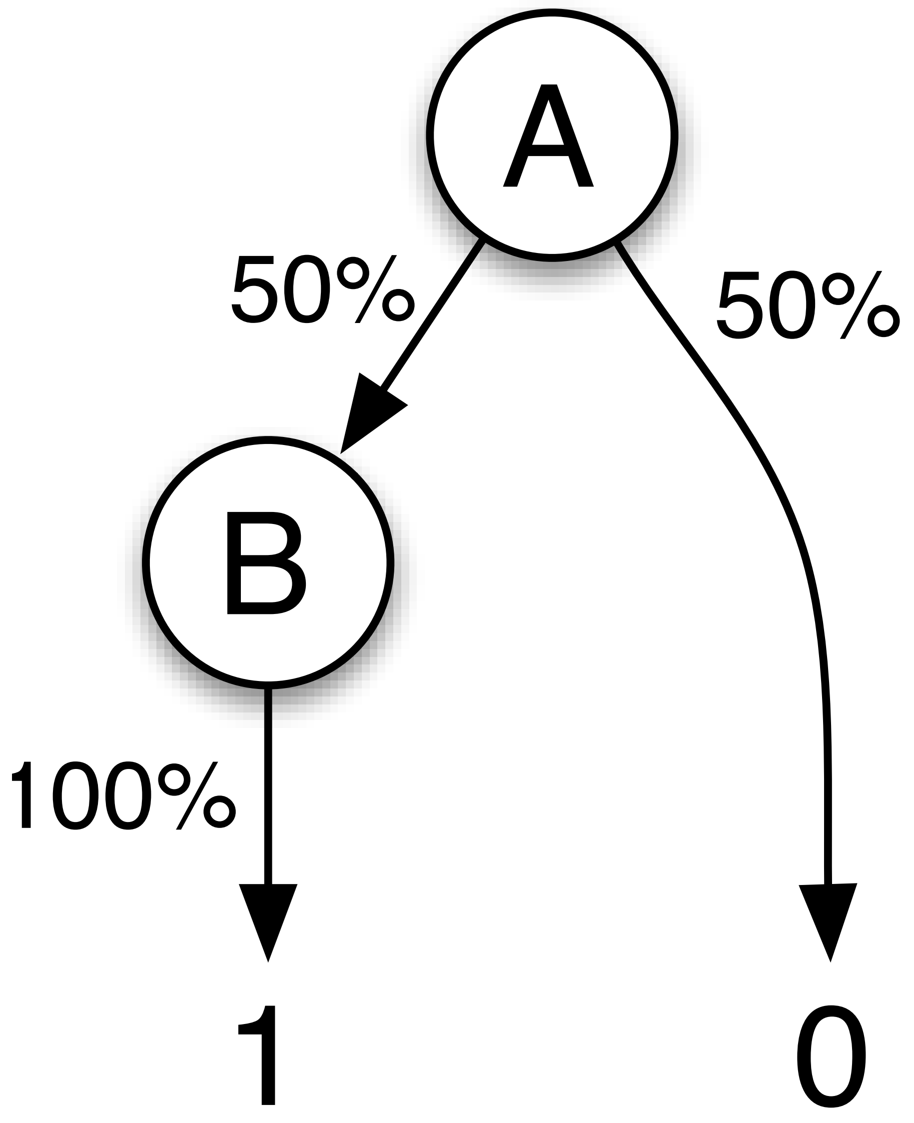
$$\frac{\partial^2 J}{\partial\theta_j \partial\theta_i} \neq \frac{\partial^2 J}{\partial\theta_i \partial\theta_j}$$

Contradiction!

Real 2nd derivatives must be symmetric

Etienne Barnard 1993

A-split example (Dayan 1992)



Clearly, the true values are

$$V(A) = 0.5$$

$$V(B) = 1$$

But if you minimize the naive objective fn,

$$J(\theta) = \mathbb{E}[\delta^2],$$

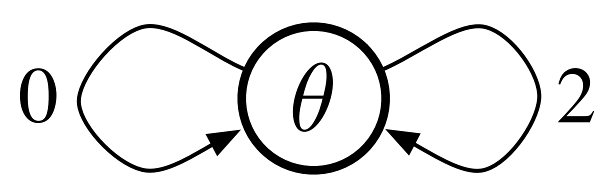
then you get the solution

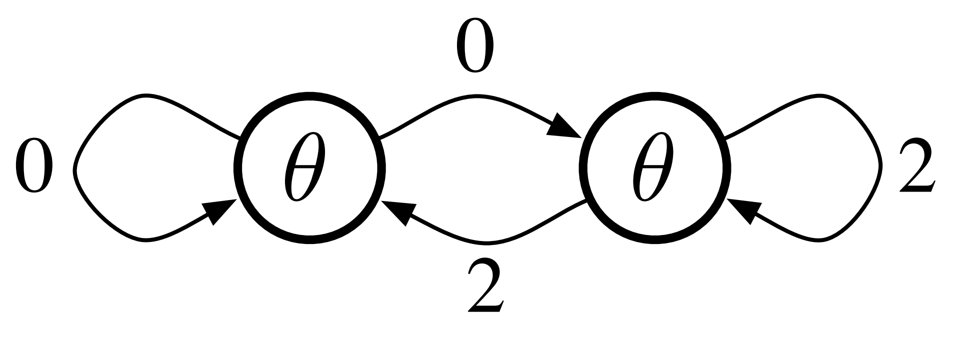
$$V(A) = 1/3$$

$$V(B) = 2/3$$

Even in the tabular case (no FA)

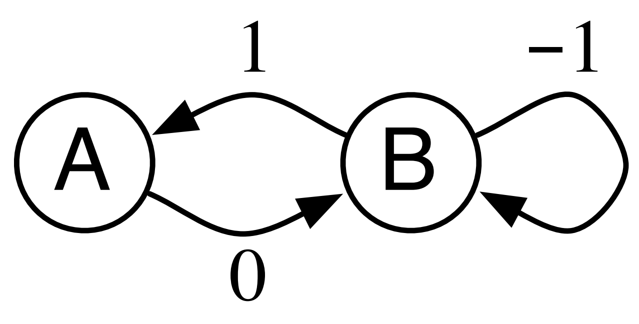
Indistinguishable pairs of MDPs

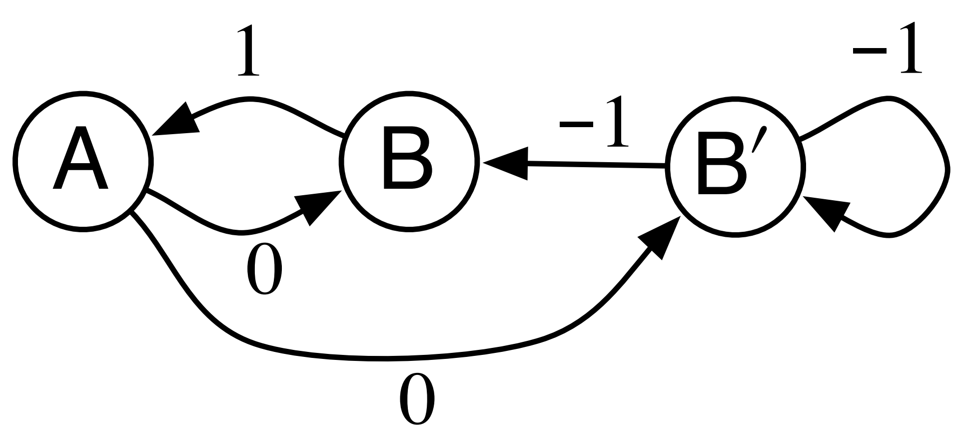




These two have different Value Errors,
but the same Return Errors
(both errors have the same minima)

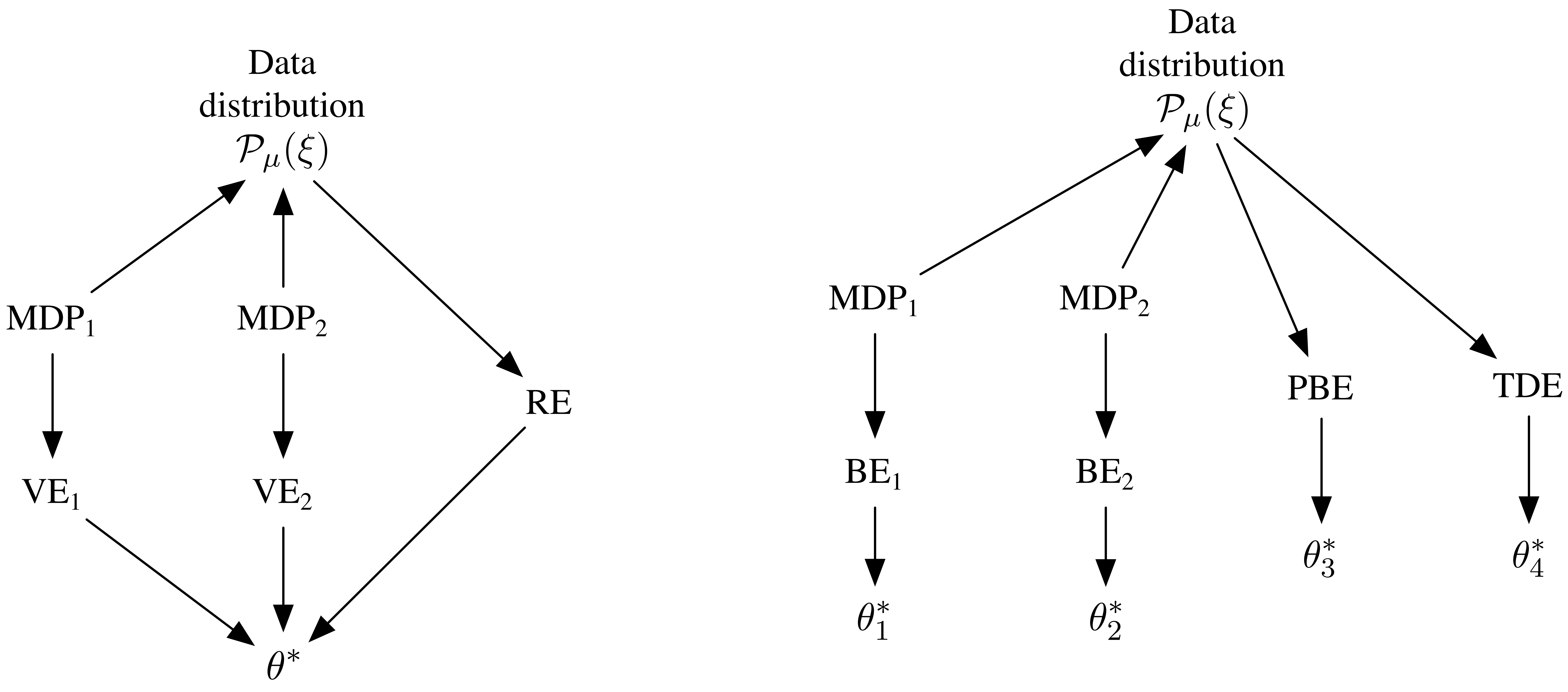
$$J_{\mathrm{RE}}(\theta)^2 = J_{\mathrm{VE}}(\theta)^2 + \mathbb{E}\left[\left(v_\pi(S_t) - G_t\right)^2 \,\middle|\, A_{t:\infty} \sim \pi\right]$$





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

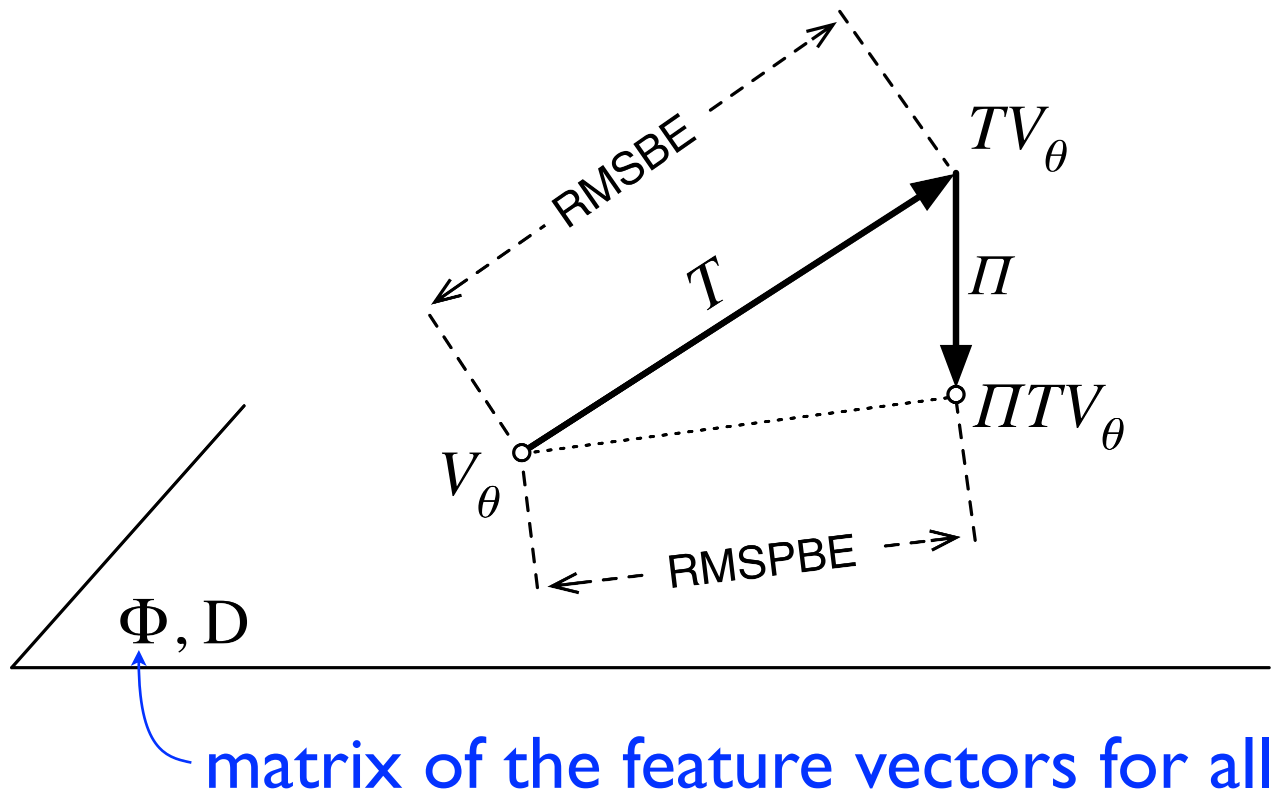


No learning algorithm can find the minimum of the Bellman Error

The Gradient-TD Family of Algorithms

- True gradient-descent algorithms in the Projected Bellman Error
- GTD(λ) and GQ(λ), for learning V and Q
- 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.

First relate the geometry to the iid statistics



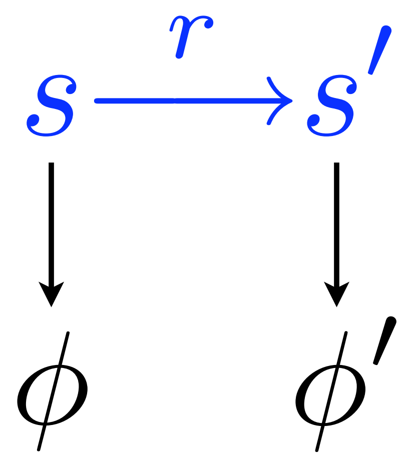
$$\Pi = \Phi(\Phi^\top D\Phi)^{-1}\Phi^\top D$$

$$\Phi^T D(TV_\theta - V_\theta) = \mathbb{E}[\delta\phi]$$

$$\Phi^T D\Phi = \mathbb{E}[\phi\phi^T]$$

$$\begin{aligned}
\text{MSPBE}(\theta) \\
&= \| V_\theta - \Pi T V_\theta \|_D^2 \\
&= \| \Pi(V_\theta - TV_\theta) \|_D^2 \\
&= (\Pi(V_\theta - TV_\theta))^\top D(\Pi(V_\theta - TV_\theta)) \\
&= (V_\theta - TV_\theta)^\top \Pi^\top D\Pi(V_\theta - TV_\theta) \\
&= (V_\theta - TV_\theta)^\top D^\top \Phi(\Phi^\top D\Phi)^{-1}\Phi^\top D(V_\theta - TV_\theta) \\
&= (\Phi^\top D(TV_\theta - V_\theta))^\top (\Phi^\top D\Phi)^{-1}\Phi^\top D(TV_\theta - V_\theta) \\
&= \mathbb{E}[\delta\phi]^\top \, \mathbb{E}\left[\phi\phi^\top\right]^{-1} \mathbb{E}[\delta\phi] \, .
\end{aligned}$$

Derivation of the TDC algorithm



$$
\begin{aligned}
\Delta\theta = -\frac{1}{2}\alpha\nabla_\theta J(\theta) \quad &= \quad -\frac{1}{2}\alpha\nabla_\theta \parallel V_\theta - \Pi T V_\theta \parallel_D^2 \\
&= \quad -\frac{1}{2}\alpha\nabla_\theta \left(\mathbb{E}\left[\delta\phi\right]\mathbb{E}\left[\phi\phi^\top\right]^{-1}\mathbb{E}\left[\delta\phi\right]\right) \\
&= \quad -\alpha\left(\nabla_\theta\mathbb{E}\left[\delta\phi\right]\right)\mathbb{E}\left[\phi\phi^\top\right]^{-1}\mathbb{E}\left[\delta\phi\right] \\
&= \quad -\alpha\mathbb{E}\left[\nabla_\theta[\phi\left(r+\gamma\phi'^\top\theta-\phi^\top\theta\right)]\right]\mathbb{E}\left[\phi\phi^\top\right]^{-1}\mathbb{E}\left[\delta\phi\right] \\
&= \quad -\alpha\mathbb{E}\left[\phi\left(\gamma\phi'-\phi\right)^\top\right]^\top\mathbb{E}\left[\phi\phi^\top\right]^{-1}\mathbb{E}\left[\delta\phi\right] \\
&= \quad -\alpha\left(\gamma\mathbb{E}\left[\phi'\phi^\top\right]-\mathbb{E}\left[\phi\phi^\top\right]\right)\mathbb{E}\left[\phi\phi^\top\right]^{-1}\mathbb{E}\left[\delta\phi\right] \\
&= \quad \alpha\mathbb{E}\left[\delta\phi\right]-\alpha\gamma\mathbb{E}\left[\phi'\phi^\top\right]\mathbb{E}\left[\phi\phi^\top\right]^{-1}\mathbb{E}\left[\delta\phi\right] \\
&\approx \quad \alpha\mathbb{E}\left[\delta\phi\right]-\alpha\gamma\mathbb{E}\left[\phi'\phi^\top\right]w \\
\text{(sampling)} \quad &\approx \quad \alpha\delta\phi-\alpha\gamma\phi'\phi^\top w
\end{aligned}
$$

This is the trick!
$w \in \Re^n$ is a second set of weights

TD with gradient correction (TDC) algorithm

aka GTD(0)

- on each transition

$$\begin{array}{ccc} s & \xrightarrow{r} & s' \\ \downarrow & & \downarrow \\ \phi & & \phi' \end{array}$$

- update two parameters

TD(0)

with gradient correction

$$\theta \leftarrow \theta + \alpha\delta\phi - \alpha\gamma\phi' \left(\phi^\top w\right)$$

$$w \leftarrow w + \beta(\delta - \phi^\top w)\phi$$

estimate of the TD error (δ) for the current state ϕ

- where, as usual

$$\delta = r + \gamma\theta^\top\phi' - \theta^\top\phi$$

Convergence theorems

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

$$\mathbb{E}[\delta\phi] \longrightarrow 0$$

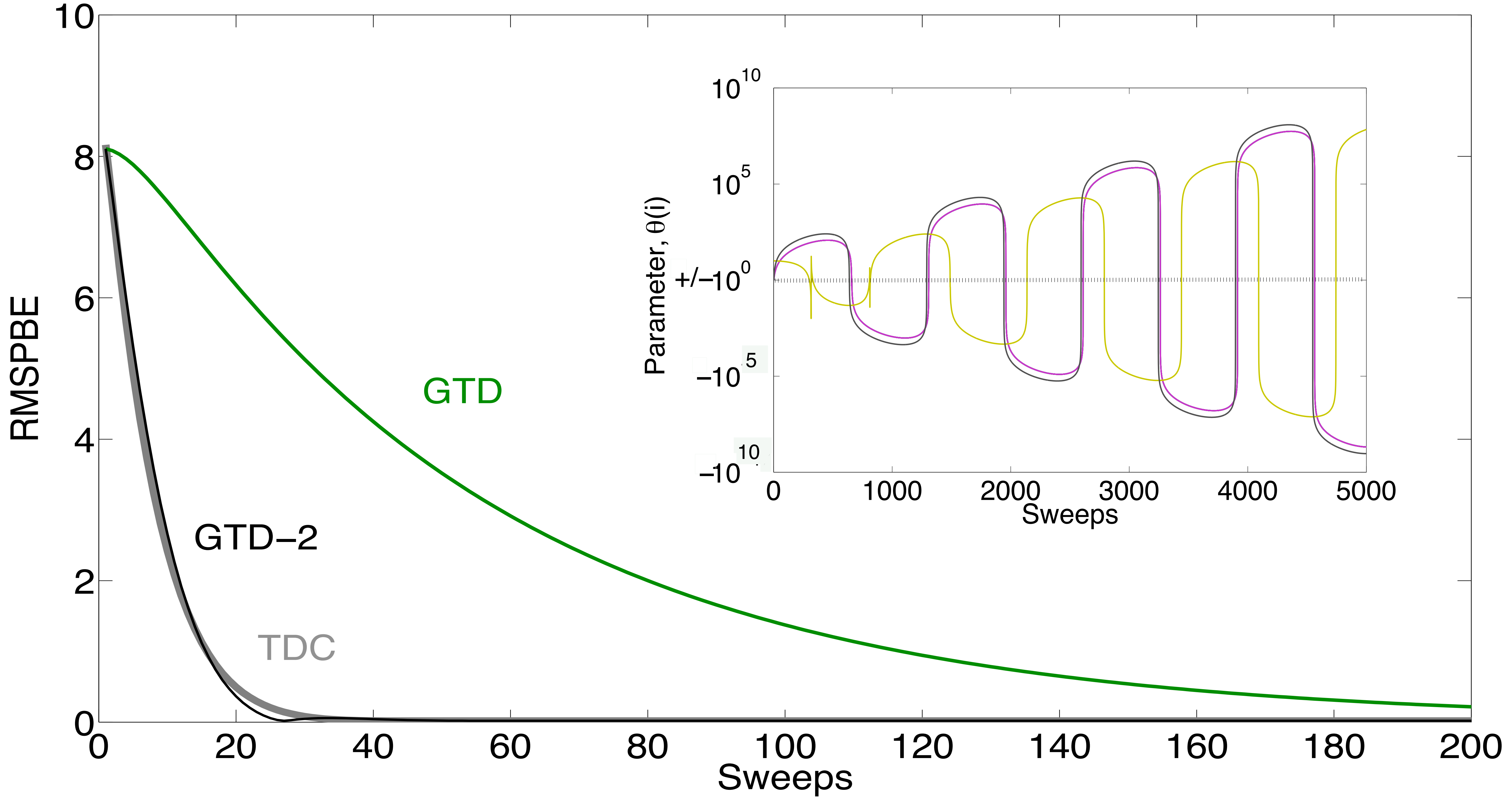
- GTD, GTD-2 converges at one time scale

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

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

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

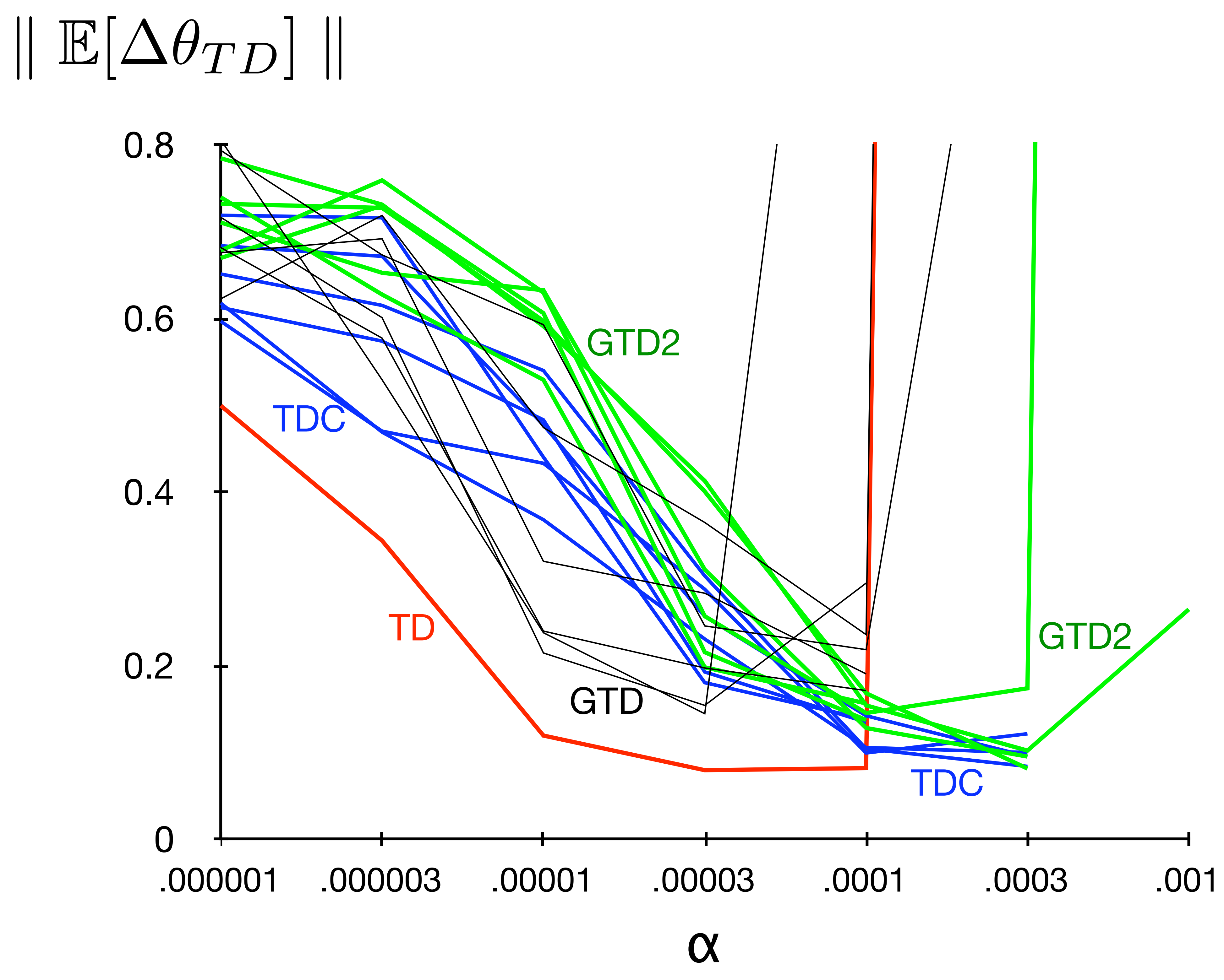
Off-policy result: Baird's counter-example



Gradient algorithms converge. TD diverges.

Computer Go experiment

- Learn a linear value function (probability of winning) for 9x9 Go from self play
- One million features, each corresponding to a template on a part of the Go board
- An established experimental testbed



ISSUE \ ALGORITHM	TD(λ), Sarsa(λ)	Approx. DP	LSTD(λ), LSPE(λ)	Fitted-Q	Residual gradient	GDP	GTD(λ), GQ(λ)
Linear computation	✓	✓	✖	✖	✓	✓	✓
Nonlinear convergent	✖	✖	✖	✓	✓	✓	✓
Off-policy convergent	✖	✖	✓	✖	✓	✓	✓
Model-free, online	✓	✖	✓	✖	✓	✖	✓
Converges to 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 λ (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

$\phi : \mathcal{S} \rightarrow \Re^n$

feature function

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

- State distribution

behavior policy

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

- Objective to minimize

parameter vector · true value function · transpose (inner product)

$$\text{MSE}(\boldsymbol{\theta}) = \sum_{s\in\mathcal{S}} d_\mu(s) i(s) \left(v_\pi(s) - \boldsymbol{\theta}^\top \phi(s)\right)^2$$

interest function $i : \mathcal{S} \rightarrow \Re^+$ · target policy

Solution

- Emphatic TD(0)

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha M_t \rho_t \left(R_{t+1} + \gamma \boldsymbol{\theta}_t^\top \phi_{t+1} - \boldsymbol{\theta}_t^\top \phi_t\right) \phi_t$$

emphasis $M_t > 0$

importance sampling ratio

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

$$\phi_t = \phi(S_t)$$

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

Problem

- State distribution

behavior policy

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

- Objective to minimize

parameter vector; true value function; transpose (inner product)

$$\text{MSE}(\boldsymbol{\theta}) = \sum_{s\in\mathcal{S}} d_\mu(s) i(s) \left(v_\pi(s) - \boldsymbol{\theta}^\top \boldsymbol{\phi}(s)\right)^2$$

interest function $i : \mathcal{S} \to \Re^+$; target policy

Solution

- Emphatic TD(0)

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha M_t \rho_t \left(R_{t+1} + \gamma \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_{t+1} - \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_t\right) \boldsymbol{\phi}_t$$

emphasis $M_t > 0$

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

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

- Emphatic LSTD(0)

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

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

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(λ)* (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(λ) (Munos)
- Emphatic TD is the simplest convergent off-policy TD algorithm (one parameter, one learning rate)