

STD(λ): learning state temporal differences with TD(λ)

Lex Weaver

Department of Computer Science
Australian National University

Jonathan Baxter
WhizBang! Labs
Pittsburgh PA USA

Reinforcement Learning

- Not supervised learning; no oracle providing the correct responses
- Learning by trial and error
- Feedback signal
 - a *reward* (or cost) being a measure of goodness (or badness)
 - which may be significantly delayed

Value Functions

- Associate a value, representing *expected future rewards*, with each system state.
 - Useful for choosing between actions leading to subsequent states.
- The (state \rightarrow value) mapping is the *true value function*.
- For most interesting problems we cannot determine the true value function.

Value Function Approximation

- Approximate the true value function with one from a parameterised class of functions.
- Use some learning method to find the parameterisation giving the “closest” approximation.
 - back propagation (supervised learning)
 - genetic algorithms (unsupervised learning)
 - TD(λ) (unsupervised learning)

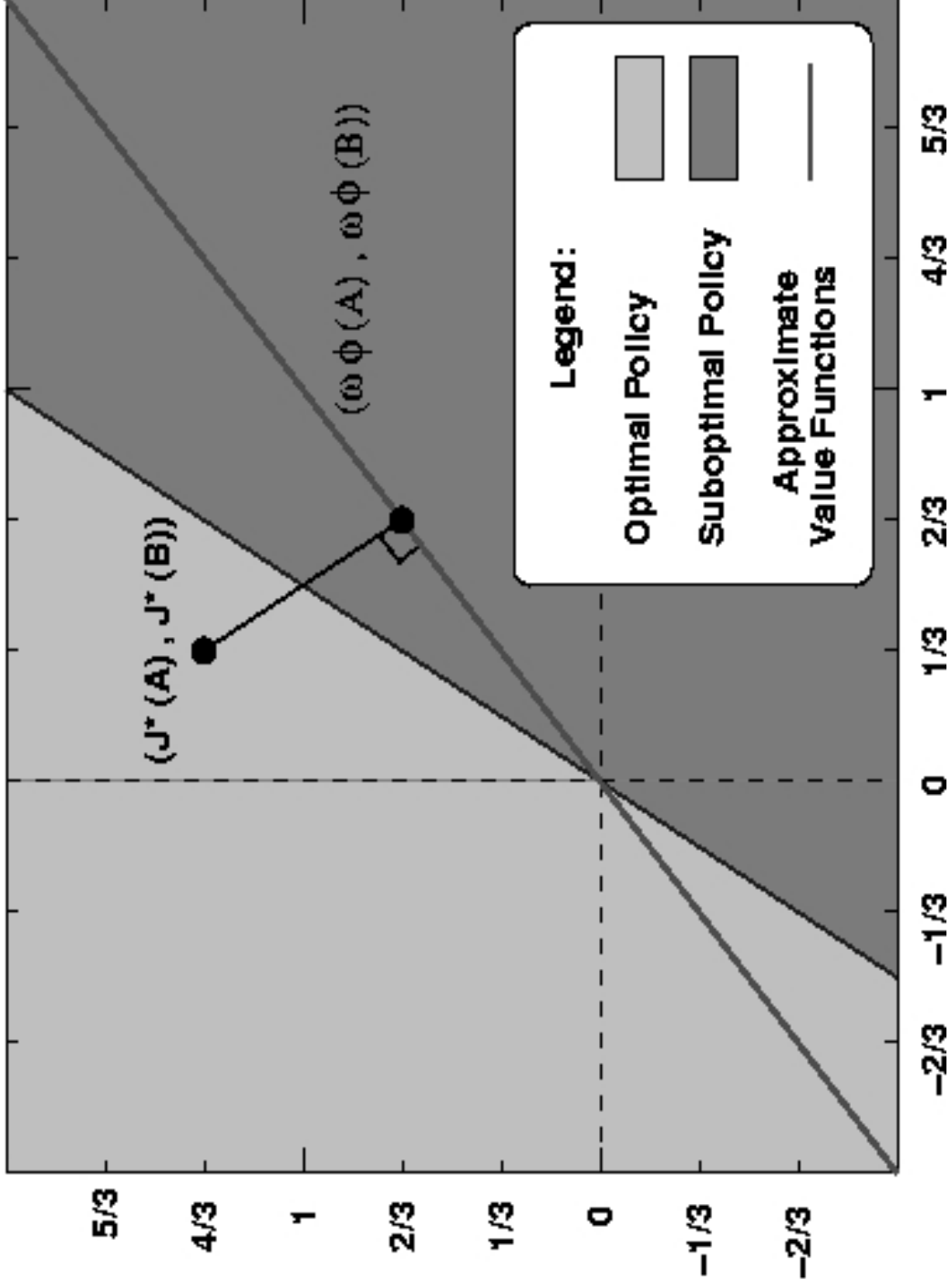
TD(λ)

- *temporal difference*: the difference between one approximation of expected future rewards, and a subsequent approximation
- TD(λ) is a successful algorithm for reducing temporal differences (Sutton 1988)
 - TD-Gammon (Tesauro 1992, 1994)
 - KnightCap, TDLeaf(λ) (Baxter *et. al.* 1998, 2000)
 - tunes parameters to most closely approximate (weighted least squares) the true value function for the system being observed

Relative State Values

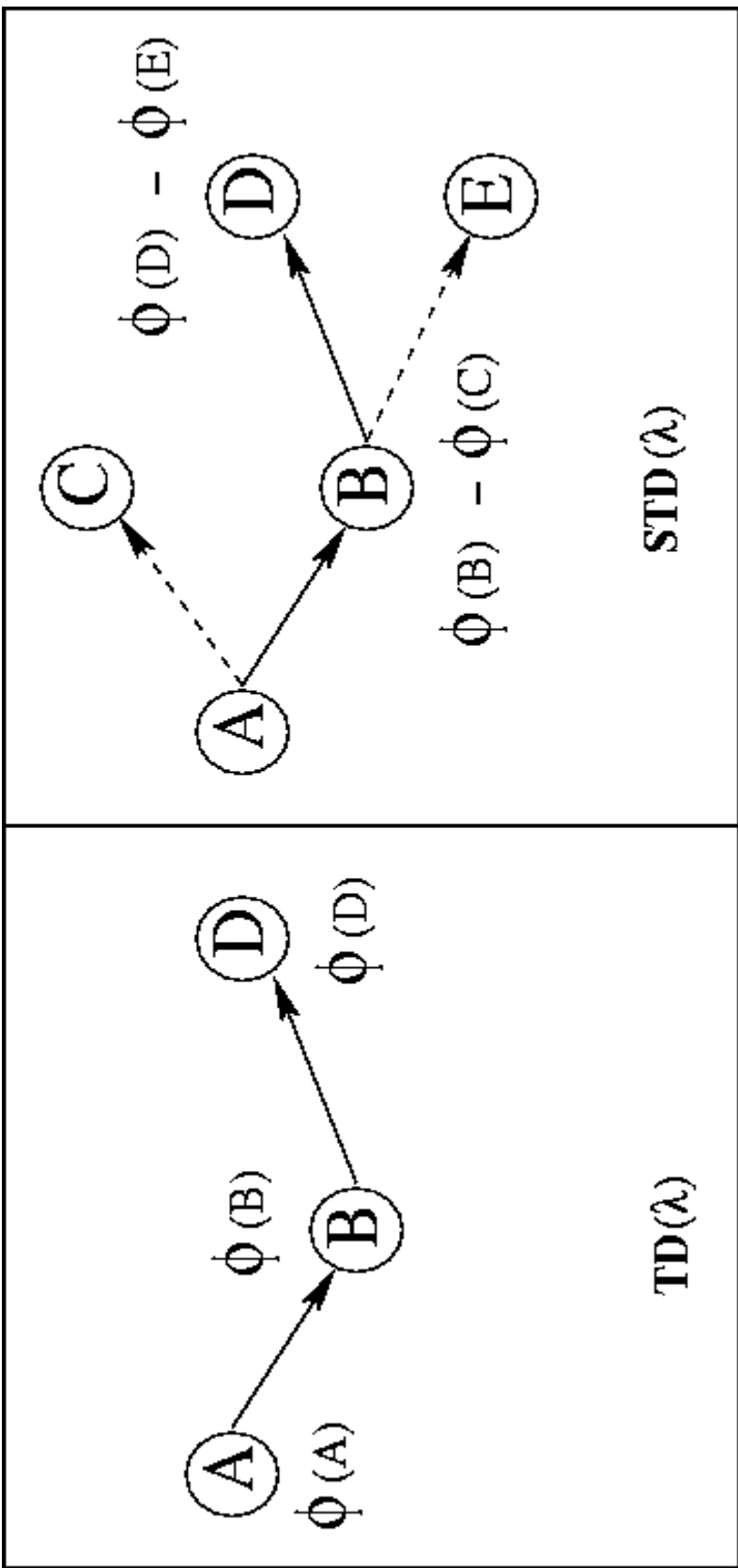
- When using [approximate] state values for making decisions, it is the relative state values that matter, not the absolute values.
- TD(λ) minimises absolute error without regard to the relativities between states.
- This can lead to problems, ...

The Two-State System



STD(λ)

- uses a TD(λ) approach to minimise temporal differences in relative state values.
 - restricted to binary markov decision processes
- weights parameter updates wrt probabilities of state transitions
 - facilitates learning optimal policies even when observing sub-optimal policies

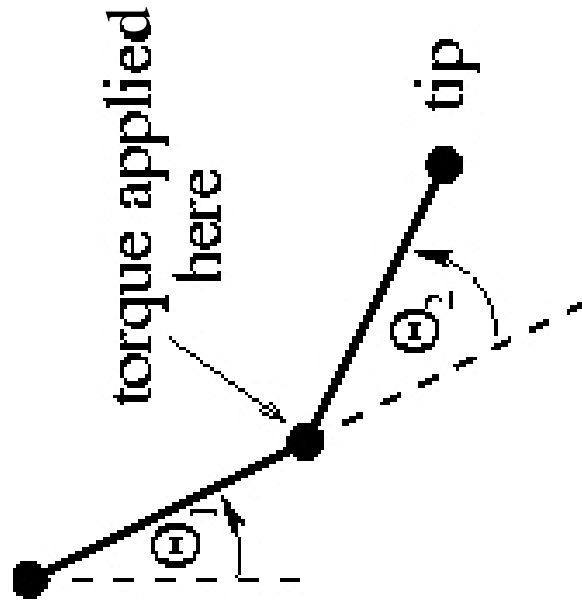


Previous Work: $DT(\lambda)$

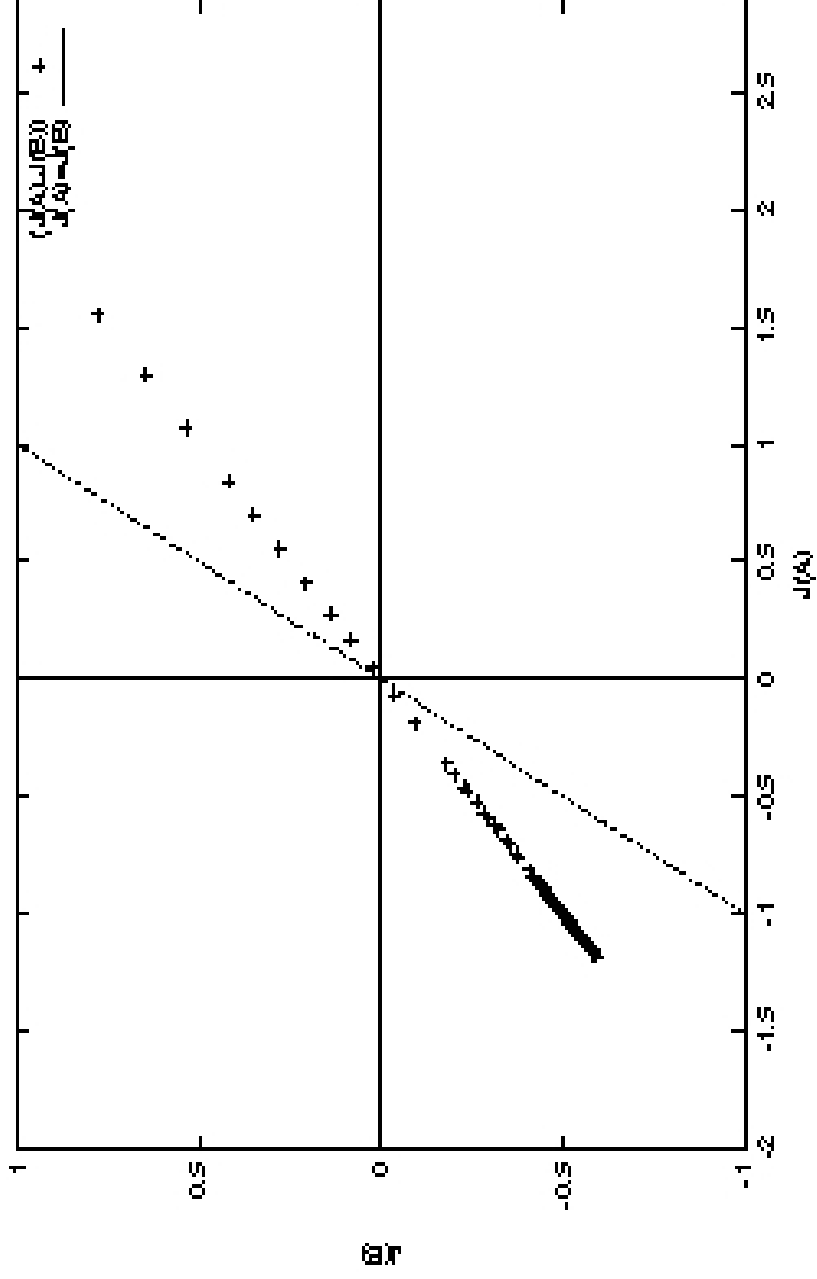
- Differential Training; Bertsekas 1997
 - during training, two instances of the system evolve independently but in lock-step
 - use a $TD(\lambda)$ approach to minimise the temporal differences in relative (between the two instances) state values
 - many of the state-pairs seen in training may never need to be compared in reality, yet accommodating them can bias the function approximator and result policy degradation

TD(λ) v STD(λ) Experiments

- Two-State System
 - observing optimal, random, and bad policies
 - STD converges to optimal, TD converges to bad policy
- Acrobot
 - single element feature vector & four element feature vector
 - observing both good and random policy
 - STD yields good policies
 - TD yields bad policies



Evolution of the Function Approximator in the Two-State System with STD(λ)



Contributions & Future Work

- Shown theoretically & empirically that learning with $\text{TD}(\lambda)$ *can* result in policy degradation.
- Proposed & successfully demonstrated an algorithm, $\text{STD}(\lambda)$, for BMDPs that does not suffer from the policy degradation identified in $\text{TD}(\lambda)$.
- Analysed $\text{DT}(\lambda)$, and shown that it includes irrelevant state pairings in parameter updates, which can result in policy degradation.
- Extend $\text{STD}(\lambda)$ to general Markov Decisions Processes.
- Investigate the significance of the reweighting (wrt state transition probabilities) of the parameter updates.