Least squares temporal difference learning

TD(λ)

- Good properties of TD
  - Easy to implement, traces achieve the forward view
  - Linear complexity = fast
  - Combines easily with linear function approximation
  - Outperforms MC methods in practice

Can we do better than TD(λ)

- Bad properties of TD
- Diverges with off-policy sampling
- It's less data efficient than it could be:
  - makes a small incremental update from each sample, then throws it away
  - it doesn't build a model of the environment
    - or save data and make multiple passes over the data
  - stochastic semi-gradient descent

A least squares approach to TD(0), on-policy

- The **TD fixed-point** solution is the weight vector found by TD in the limit of infinite sampling
- The weight vector corresponding to zero MSPBE
- It also corresponds to: $E[\boldsymbol{\delta}_t(\theta)\boldsymbol{\phi}_t \mid \pi] = 0$
  - the expected (TD-error * features), with samples drawn according to π is zero.

A least squares approach to TD(0), on-policy

- $E[\delta_t(\theta)\phi_t \mid \pi] = 0$
  - we can approximate this by making it zero with respect to the observed data

- $$\frac{1}{T}\sum_{t=1}^{T}\delta_t(\theta)\phi_t = 0$$

  $$= \frac{1}{T}\sum_{t=1}^{T}\left(R_{t+1} + \gamma\theta^\top\phi(S_{t+1}) - \theta^\top\phi(S_t)\right)\phi(S_t)$$

LSTD(0)

$$= \frac{1}{T}\sum_{t=1}^{T}\big(R_{t+1} + \gamma\theta^\top\phi(S_{t+1}) - \theta^\top\phi(S_t)\big)\,\phi(S_t)$$

$$= \frac{1}{T}\sum_{t=1}^{T} R_{t+1}\phi(S_t) + (\gamma\theta^\top\phi(S_{t+1}) - \theta^\top\phi(S_t))\phi(S_t)$$

$$= \frac{1}{T}\sum_{t=1}^{T} R_{t+1}\phi(S_t) + \frac{1}{T}\sum_{t=1}^{T}(\gamma\theta^\top\phi(S_{t+1}) - \theta^\top\phi(S_t))\phi(S_t)$$

$$= \mathbf{b} + \frac{1}{T}\sum_{t=1}^{T}(\gamma\theta^\top\phi(S_{t+1}) - \theta^\top\phi(S_t))\phi(S_t)$$

$$= \mathbf{b} + \frac{1}{T}\sum_{t=1}^{T}(\gamma\phi(S_{t+1})^\top\theta - \phi(S_t)^\top\theta)\phi(S_t)$$

$$= \mathbf{b} + \frac{1}{T}\sum_{t=1}^{T}(\gamma\phi(S_t)\phi(S_{t+1})^\top\theta - \phi(S_t)\phi(S_t)^\top\theta)$$

$$= \mathbf{b} + \frac{1}{T}\sum_{t=1}^{T}\phi(S_t)(\gamma\phi(S_{t+1}) - \phi(S_t))^\top\theta$$

$$= \boldsymbol{b} - \frac{1}{T}\sum_{t=1}^{T}\phi(S_t)(\phi(S_t) - \gamma\phi(S_{t+1}))^T\theta$$

$$0 = \boldsymbol{b} - A_\pi\theta$$

LSTD(0)

$$0 = \boldsymbol{b} - A_\pi \theta$$

thus,

$$\theta = A_\pi^{-1} \boldsymbol{b}$$

$$A_\pi = \frac{1}{T} \sum_{t=1}^{T} \phi(S_t)(\phi(S_t) - \gamma\phi(S_{t+1}))^T$$

$$\boldsymbol{b} = \frac{1}{T} \sum_{t=1}^{T} R_{t+1} \phi(S_t)$$

Batch policy evaluation with LSTD(0)

1. Collect a batch of data by selecting actions according to π
2. Compute A_π and $\mathbf{b}$
3. Compute inverse of A_π
    - typically with a numerically stable method designed for large matrices
    - e.g., Singular value decomposition
4. Directly compute $\theta = (A_\pi)^{-1}\mathbf{b}$

Incremental policy evaluation with LSTD(0)

1. Initialize A_π and **b** to zero
2. On each step, take action A_t according to **π** and observe S_{t+1} and R_{t+1}
    1. $A_{\pi,t+1} = A_{\pi,t} + \phi_t(\phi_t - \gamma\phi_{t+1})^\top$
    2. $\mathbf{b}_{t+1} = \mathbf{b}_t + R_{t+1}\phi_t$

- Invert A_π and compute $\theta = (A_\pi)^{-1}b$ **when required**

LSTD(0) v TD(0)

- Advantages of LSTD(0) algorithm:
- No learning rate parameter
- No bias due to initial value function
- Performs much better than TD(0) in practice…more latter

Multiple ways to arrive at LSTD(0) algorithm

- Start with the MSPBE
- Take the gradient
- Set it equal to zero
- And directly solve for θ

MSPBE

$$\mathbf{MSPBE}(\theta) \quad = \quad \| V_\theta - \Pi T V_\theta \|_D^2$$

- We can rewrite the MSPBE as:
  - $\text{MSPBE}(\theta) = (A_\pi\theta - \mathbf{b})^T C^{-1}(A_\pi\theta - \mathbf{b})$
  - where A_π is the LSTD(0) matrix
  - $\mathbf{b}$ is the same as in LSTD(0)
  - C^{-1} is the inverse feature covariance matrix $C = E[\boldsymbol{\phi}_t\boldsymbol{\phi}_t^T]$
- Take gradient of MSPBE(θ) with respect to θ
  - $\boldsymbol{\nabla}_\theta \text{MSPBE}(\theta) = A_\pi C^{-1}(A_\pi\theta - \mathbf{b})$

MSPBE

$$\text{MSPBE}(\theta) \quad = \quad \| V_\theta - \Pi T V_\theta \|_D^2$$

- $\nabla_\theta \text{MSPBE}(\theta) = A_\pi C^{-1}(A_\pi\theta - \mathbf{b})$
- $A_\pi C^{-1}(A_\pi\theta - \mathbf{b}) = 0$
  - A_π and C^{-1} inverses exist
  - $(A_\pi\theta - \mathbf{b}) = 0$
  - $\theta = A_\pi^{-1}\mathbf{b}$
- Same as LSTD(0)
- The LSTD solution is the direct or closed form solution of the MSPBE

LSTD(λ)

- To derive the algorithm, again we start with the TD-fixed point
- In the case of λ>0, the weight vector corresponding to the TD-fixed point satisfies:
- $E[\boldsymbol{\delta}_t(\theta)\mathbf{e}_t \mid \pi] = 0$
- Can you follow the same derivation to get the update equations for LSTD(λ)…or template match as guess the algorithm

LSTD(λ)

$$\mathbf{e}_t \leftarrow \gamma\lambda\mathbf{e}_{t-1} + \phi_t$$

$$A_{\pi,t+1} = A_{\pi,t} + \mathbf{e}_t(\phi_t - \gamma\phi_{t+1})^\top$$

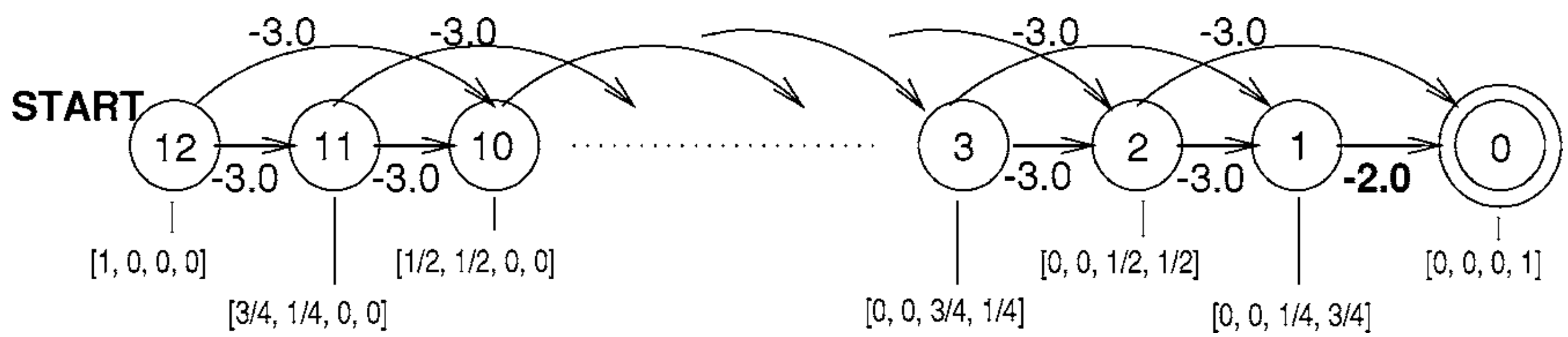
$$\mathbf{b}_{t+1} = \mathbf{b}_t + R_{t+1}\mathbf{e}_t$$

LSTD(λ)

- Finds the same weight vector as found by TD(λ) in the limit of infinite sampling
- Corresponds to the closed form solution to the λ version of the MSPBE

LSTD(λ) experiments

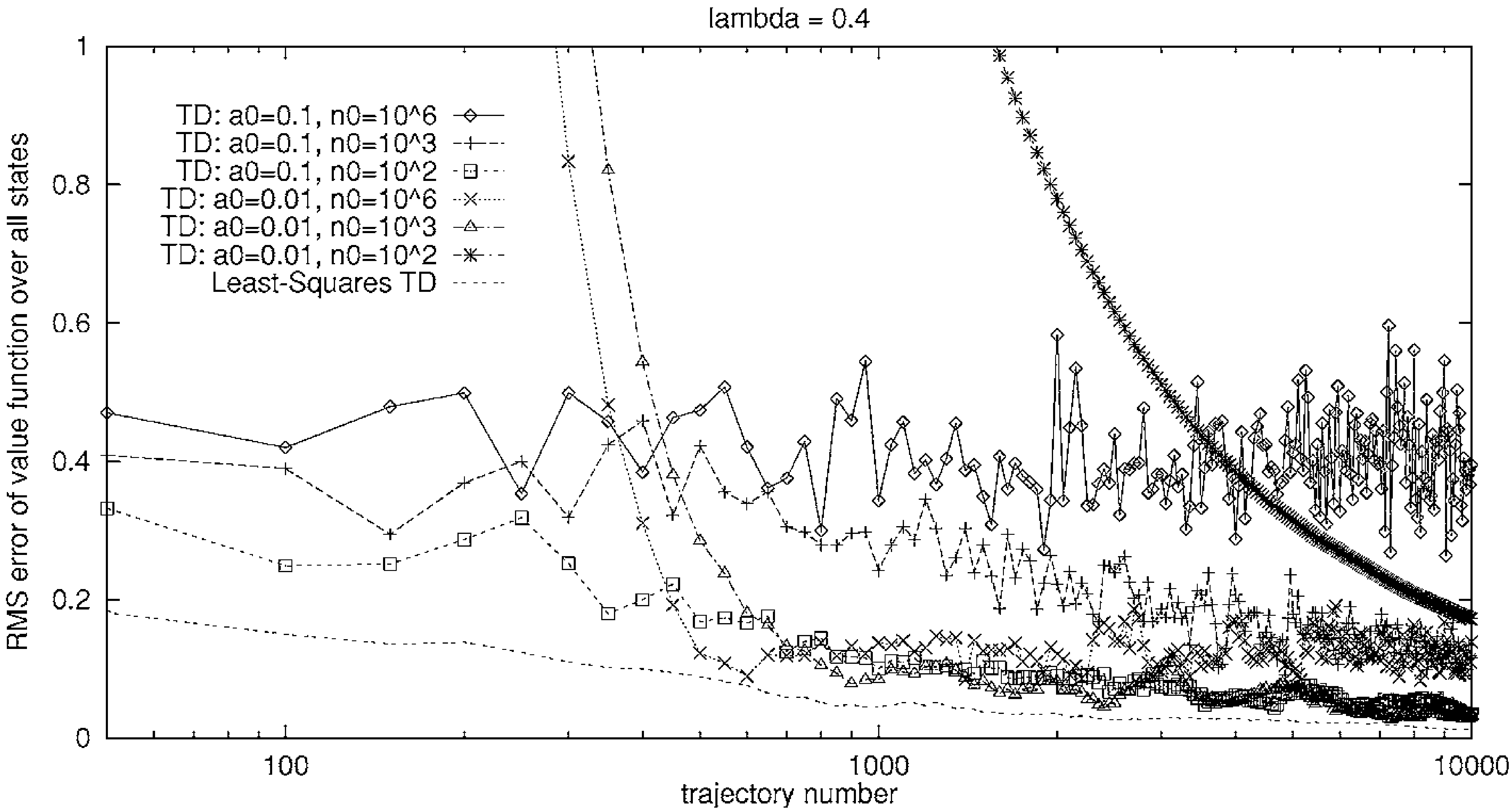
- Boyan Chain
- Episodic, 13-state chain
- 4D feature vector, but zero error solution possible—we can represent the true value function=[-24,-16,-8,0]



Experimental setup

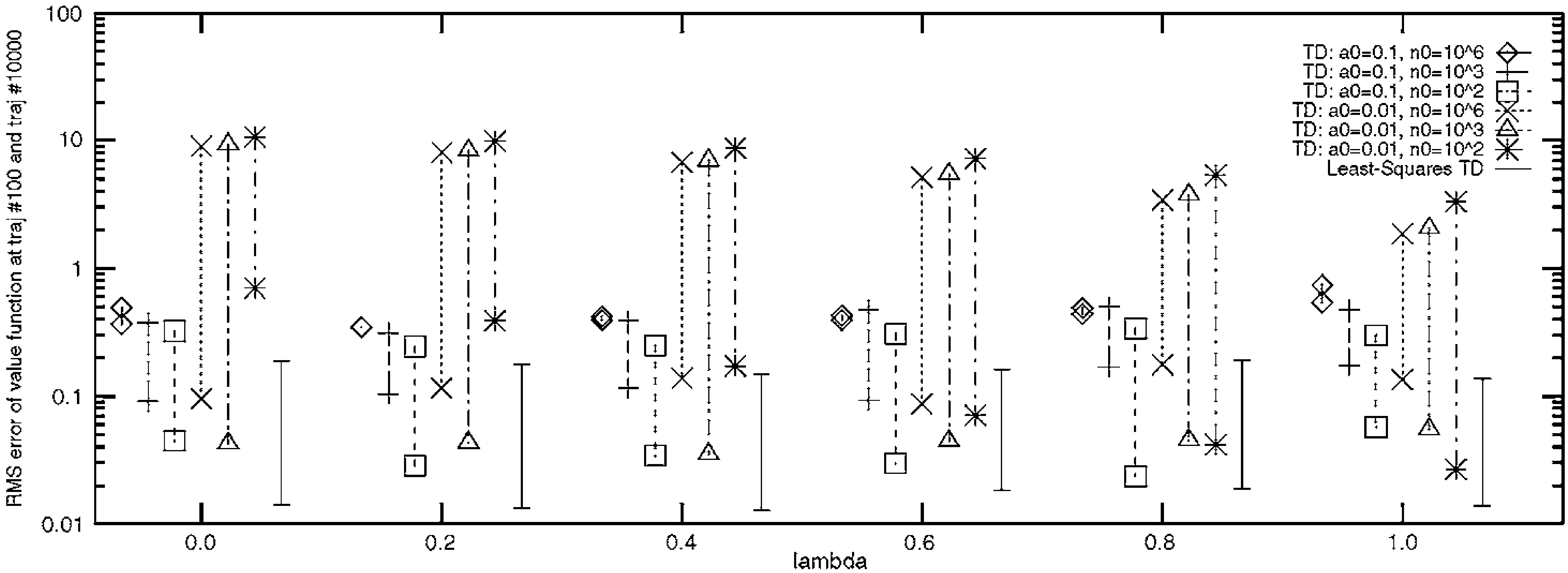
- Want to compare TD(λ) and LSTD(λ) under early learning and asymptote (close to convergence with a computer program)
- Convergence of TD requires
  - $\boldsymbol{\alpha}_t \geq 0$; $\sum \boldsymbol{\alpha}_t = \infty$; $\sum (\boldsymbol{\alpha}_t)^2 < \infty$
  - e.g., $\boldsymbol{\alpha}_t = 1/t$ (harmonic series)
- Boyan used $\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_0 (t_0 + 1)/(t_0 + t)$
  - bigger t_0, slower $\boldsymbol{\alpha}$ decreases
  - $\boldsymbol{\alpha}_0$={0.1,0.01}, t_0={10^2,10^3,10^6}
- Initial weights = zero vector
- Results averaged over 10 independent runs

Results



lambda = 0.4
TD: a0=0.1, n0=10^6
TD: a0=0.1, n0=10^3
TD: a0=0.1, n0=10^2
TD: a0=0.01, n0=10^6
TD: a0=0.01, n0=10^3
TD: a0=0.01, n0=10^2
Least-Squares TD
RMS error of value function over all states
trajectory number
1
0.8
0.6
0.4
0.2
0
100
1000
10000

λ sensitivity



Experiment summary

- LSTD(λ) learns a good approximation of the value function with less data than TD(λ)
- LSTD(λ) gets better long-run error
- TD(λ)'s performance is dramatically effected by choice of step-size parameter
- LSTD(λ) does not seem sensitive to λ

More incremental form of LSTD

- Inverting the matrix in LSTD costs $O(n^3)$ computation
- Bradtke and Barto noticed that the Sherman-Morrison formula can be used to incrementally update the inverse of A_π:
  - $A_\pi^{-1} = A_\pi^{-1}$ + outer product

$$C_{t+1} = C_t - \frac{C_t \mathbf{e_t} (\phi_t - \gamma \phi_{t+1})^\top C_t}{1 + (\phi_t - \gamma \phi_{t+1})^\top C_t \mathbf{e_t}}$$

$$\theta_{t+1} = C_{t+1} \mathbf{b}_{t+1}$$

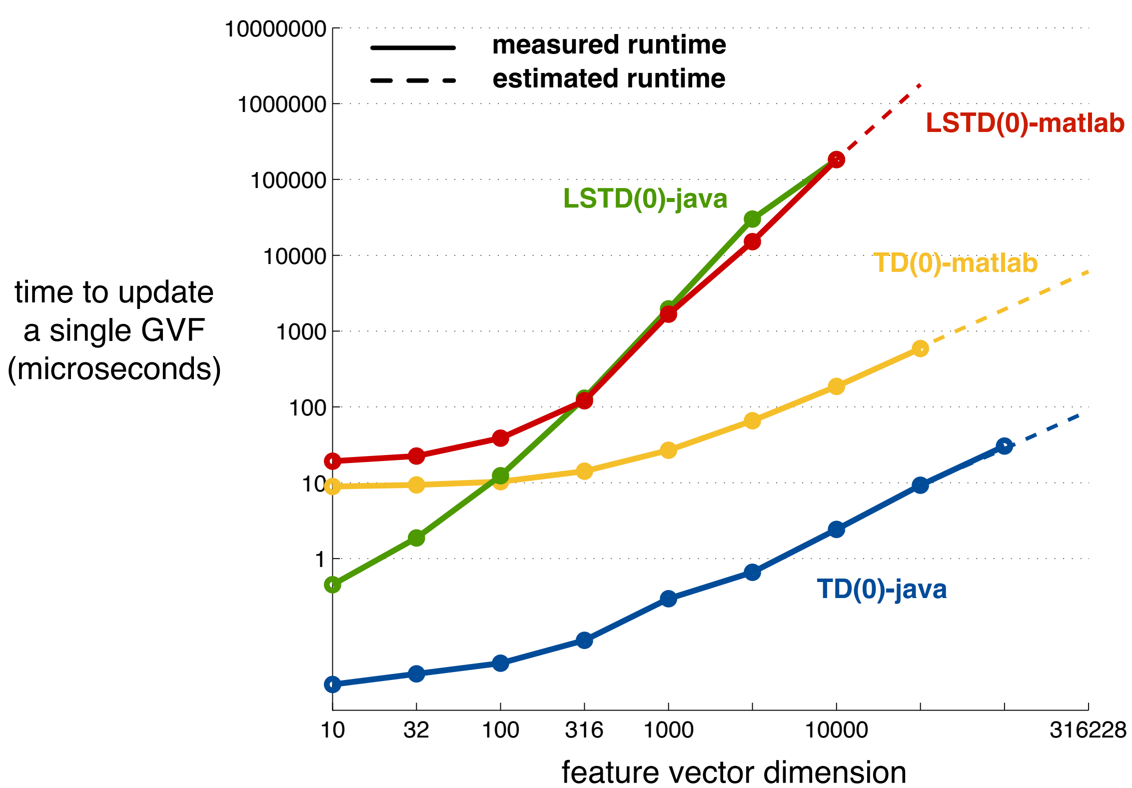
incremental LSTD

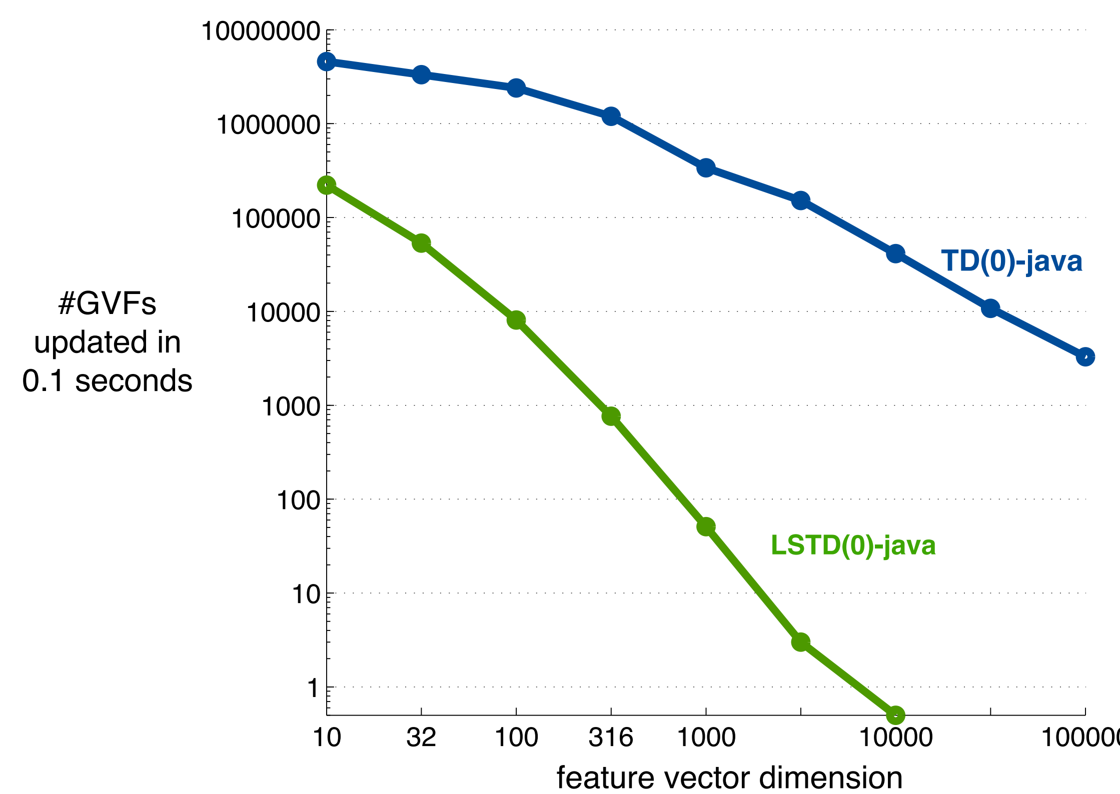
$$C_{t+1} = C_t - \frac{C_t \mathbf{e_t} (\phi_t - \gamma\phi_{t+1})^\top C_t}{1 + (\phi_t - \gamma\phi_{t+1})^\top C_t \mathbf{e_t}}$$

- When we use batch LSTD(λ), we typically wait until we have process many samples before we invert A_π
- With the incremental version, we are effectively inverting the C matrix on every step
  - in the beginning of learning this can lead to very poor behavior
  - most of C is zero
- We initialize $C_0 = I\beta$
  - β can be very small (e.g., 0.0001) or very large (e.g., 10000)
- Incremental LSTD(λ)'s performance is sensitive to this value

Sometimes we still want to do TD(λ)

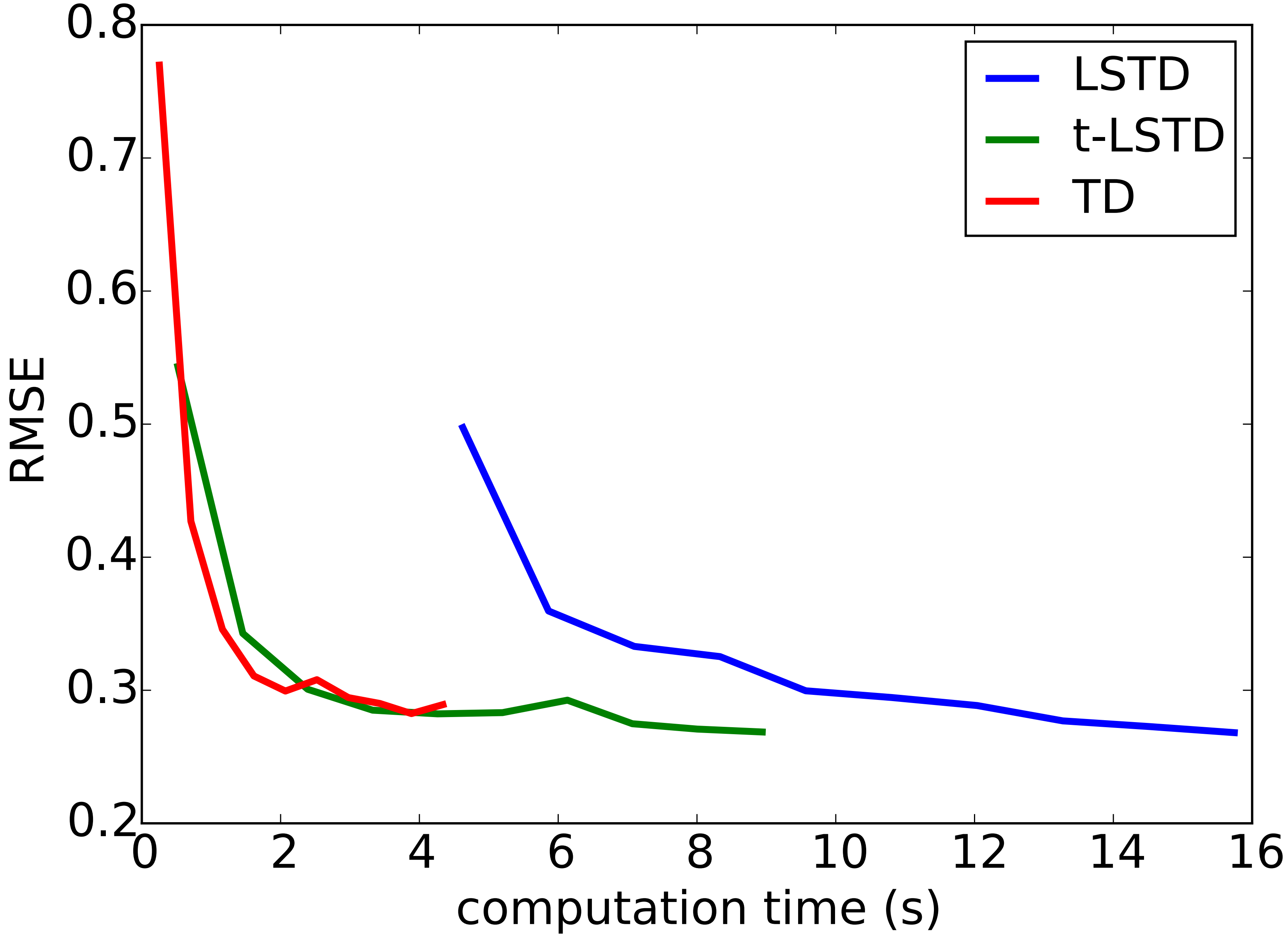
- If the number of features is large, and/or the number of value functions is large
- $O(n^2)$ computation and memory per time-step, per value function hurts





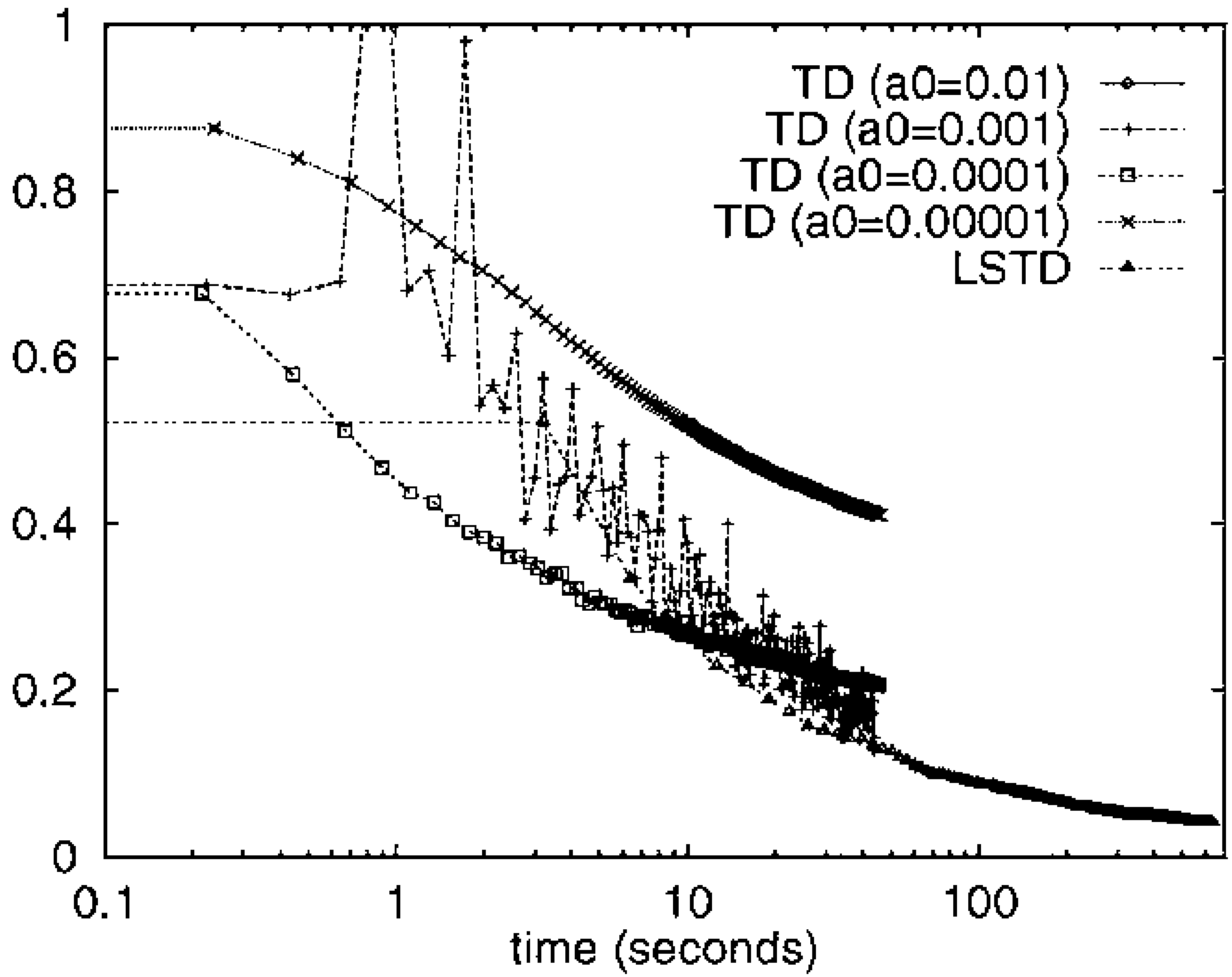
Sometimes we still want to do TD(λ)

- Local **IU** work (Gehring, Pan, & White) on LSTD on Mountain Car domain



Sometimes we still want to do TD(λ)

- Boyan's Backgammon experiment:



sometimes TD(λ), sometimes LSTD(λ)

- Boyan:
  - “If a domain has many features and simulation data is available cheaply, then incremental methods such as TD(λ) may have better real-time performance than least-squares methods. On the other hand, some reinforcement learning applications have been successful with small numbers of features … and in these situations LSTD(λ) should be superior.”
- Szepesvari:
  - if we take into account **frequency** with which samples are available, TD(λ) will achieve better accuracy than LSTD(λ) at high sampling rates—e.g., some robots

Policy iteration with LSTD(λ)

- Use state-action features
- Define a behavior policy μ—(e.g., epsilon-greedy)
- Define a target policy $\pi=\mu$
- Select actions according to μ
- Use incremental LSTD(λ) to update θ

- Like an LSTD variant of Sarsa(λ)

incremental LSTD for control

$$C_{t+1} = C_t - \frac{C_t \mathbf{e_t} (\phi_t - \gamma\phi_{t+1})^\top C_t}{1 + (\phi_t - \gamma\phi_{t+1})^\top C_t \mathbf{e_t}}$$

- There can be large variance in performance of incremental LSTD(λ) control algorithm
- Best practice:
  - use the algorithm in mini-batch style
  - only recompute θ every **k** iterations
    - updating θ changes the policy
  - smooths out problems due to initial non-accurate C matrix learning to strange initial policies
  - k is problem dependent

incremental LSTD for control

- In general it is widely held that incremental LSTD for control suffers from the **forgetting** problem
- Most of the C matrix summarizes prior policies that are no longer the current behavior policy
  - ...the policy changes continually during policy iteration
  - C is out of date and strongly skewing the solution for θ
  - form of non-stationarity
- Linear, incremental methods like Sarsa use a vector of weights that seems to more naturally handle this problem
- There are no conclusive imperial studies comparing Sarsa and incremental LSTD for control

Off-policy LSTD(λ)

- Simple extension
- Just add importance sampling corrections to the traces :

$$\mathbf{e}_t \leftarrow \rho_t(\gamma\lambda\mathbf{e}_{t-1} + \phi_t)$$

$$A_{\pi,t+1} = A_{\pi,t} + \mathbf{e}_t(\phi_t - \gamma\phi_{t+1})^\top$$

$$\mathbf{b}_{t+1} = \mathbf{b}_t + R_{t+1}\mathbf{e}_t$$

$$\rho_t \stackrel{\text{def}}{=} \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}$$

- Can implement incrementally with Sherman-Morrison formula
- Proven to converge under general conditions by Yu (2010)

Off-policy, policy iteration with LSTD(λ)

- Use state-action features
- Define a behavior policy μ—(e.g., epsilon greedy)
- Define a target policy π—(greedy)
- Select actions according to μ
- Use off-policy incremental LSTD(λ) to update θ

- Like a LSTD variant of Q(λ) learning

References

- LSTD

  - Bradtke and Barto (1996). Linear least-squares algorithms for temporal difference learning
  - Geramifard et al (2006). Incremental Least-Squares Temporal Difference Learning
  - Szepesv´ari (2009). Algorithms for Reinforcement Learning.

- LSTD(λ)

  - Boyan (2002). Technical Update: Least-Squares Temporal Difference Learning.
  - Gehring et al (2016). Incremental Truncated LSTD.

- Off-policy LSTD(λ)

  - Yu (2010). Convergence of Least Squares Temporal Difference Methods Under General Conditions.