Preconditioned Temporal Difference Learning

Hengshuai Yao
City Univ. of Hong Kong
hengshuai@gmail.com

TD Learning with LFA [Sutton 88]

- Well known for its sucess in Control and AI game [Barto, et al 83] [Tesauro 92]
  - Incremental
  - O(K)
- Inefficient use of data unless using experience replay TD variant [Lin, 92]
  - LSTD, [Bradtke & Barto 96] [Boyan 99,02]
  - LSPE, [Bertsekas, et al., 96, 03, 04]
  - iLSTD, [Geramifard 06a,06b]

Outline

Motivation---Using Preconditioning technique to:

1. Connect existing policy evaluation algorithms
2. Extend them to LSXY or iLSXY
3. Save us from Tuning the step-size, and
4. Implement them more efficiently and stably

Preconditioning

How to View and Extend existing policy evaluation alg's as Preconditioning

On-line Step-size for all PTD alg's

Reducing complexities: Incremental Preconditioning with Compressed Sparse Row Representation .

Experimental Results

What is preconditioning?

- A linear system of equations

$$Aw=-b,$$

can be sovled by ***unpreconditioned*** alg':

$$w_{t+1}=w_t+\alpha(Aw_t+b)$$

$$=(I+\alpha A)w_t+\alpha b$$

- ***Preconditioning*** the original system before solving it: $C^{-1}Aw=-C^{-1}b$, *C:* preconditioner.
- Standard preconditioned algorithms:

$$w_{t+1}=w_t+C^{-1}(Aw_t+b)$$

$$=(I+C^{-1}A)w_t+C^{-1}b$$

How to use preconditioning in RL?

- Convergence of TD [Dayan 92; Tsitsiklis&Van Roy 97]

$$w_{t+1}=w_t+\alpha\, E\{\delta_t(w_t) \mid w_t\},$$

$$=w_t+\alpha\,(Aw_t+b)$$

- Idea:
  - Estimate *A* and *b* on line by Robbins-Monro
  - Then apply preconditioning to the estimated linear equation

$$A_t w=-b_t$$

Gradient Descent and Preconditioning

$$A_t w = -b_t,$$

- Objective Error Function $||A_t w + b_t||^2$
- Gradient Descent algorithm

$$w_{t+1} = w_t - \alpha_t A_t' (A_t w_t + b)$$

- Apply Preconditiong to Gradient Descent

$$w_{t+1} = w_t + B_t^{-1} A_t' (A_t w_t + b)$$

$$\mathbf{A_t' A_t w = -A_t' b_t}$$

- Taking $B_t = A_t' C_t$, Preconditiond TD:

$$w_{t+1} = w_t + \alpha_t \boldsymbol{C_t^{-1}(A_t w_t + b_t)}$$

Preconditioned Residual

Connecting and Extending existing alg's

Preconditioner Varying

- $\boldsymbol{C_t=A_t}$ **LSTD**
- $\boldsymbol{C_t=D_t}$ **LSPE**
- $\boldsymbol{C_t=I}$ **iLSTD**
- $C_t=E_t$ Jacobi
- $C_t=F_t$ Gauss-Seidel
- $C_t=G_t$ SOR
- ...

Efficient and stable implementation $O(K^2)$

Incremental Preconditioning

- LSTD and LSPE Preconditioning:
  - Too Complex, requiring $O(K^3)$ to invert the preconditioner
  - Slow and Unstable learning in the face of limited data
- Solutions:
  - Sherman-Morison recursive inversion, $O(K^2)$, RLS-TD [Bradtke & Barto 96], [Xu, et al.,02]
  - incremental LSTD [Geramifard, et al., 06]
  - ***Incremental Preconditioning***, $O(K^2)$, $O(qK)$

Incremental Preconditioning

Iterative Preconditioned Residual

- Preconditioning:
  - Preconditioned residual

$$\delta_t = C_t^{-1} e_t, \qquad e_t = A_t w_t + b_t,$$

- Incremental Preconditioning
  - Iterative preconditioned residual

$$C_t \delta_t = e_t \quad \Longleftrightarrow \quad C_t \delta_t - e_t = f_t$$

$$\Delta_{t+1} = \Delta_t + \beta_t f_t$$

iPTD: iLSPE;
i-Jacobi, i-GS, i-SOR:
parts of iLSTD

Now complexity is reduced to $O(K^2)$

Incremental PTD

Simulation $s_t \rightarrow s_{t+1}, r_t$

Model Estimation

$$A_{t+1} = A_t + 1/T \left(z_t \left(\gamma \, \phi(s_{t+1}) - \phi(s_t) \right)' - A_t \right)$$

$$b_{t+1} = b_t + 1/T \left(z_t r_t - b_t \right)$$

Residual

$$e_t = A_t w_t + b_t$$

Iterative Residual

$$f_t = C_t \, \delta_t - e_t$$

$$\Delta_{t+1} = \Delta_t + \beta_t f_t$$

Weight

$$w_{t+1} = w_t + \alpha_t \Delta_{t+1}$$

CSR

Online α_t

Ease of Tuning

On-line Step-size for PTD alg's

- (Old) Preconditioned residual

$$\delta_t = C_t^{-1}(A_t w_t + b_t)$$

- New Preconditioned residual

$$\theta_{t+1} = C_t^{-1}(A_t w_{t+1} + b_t),$$

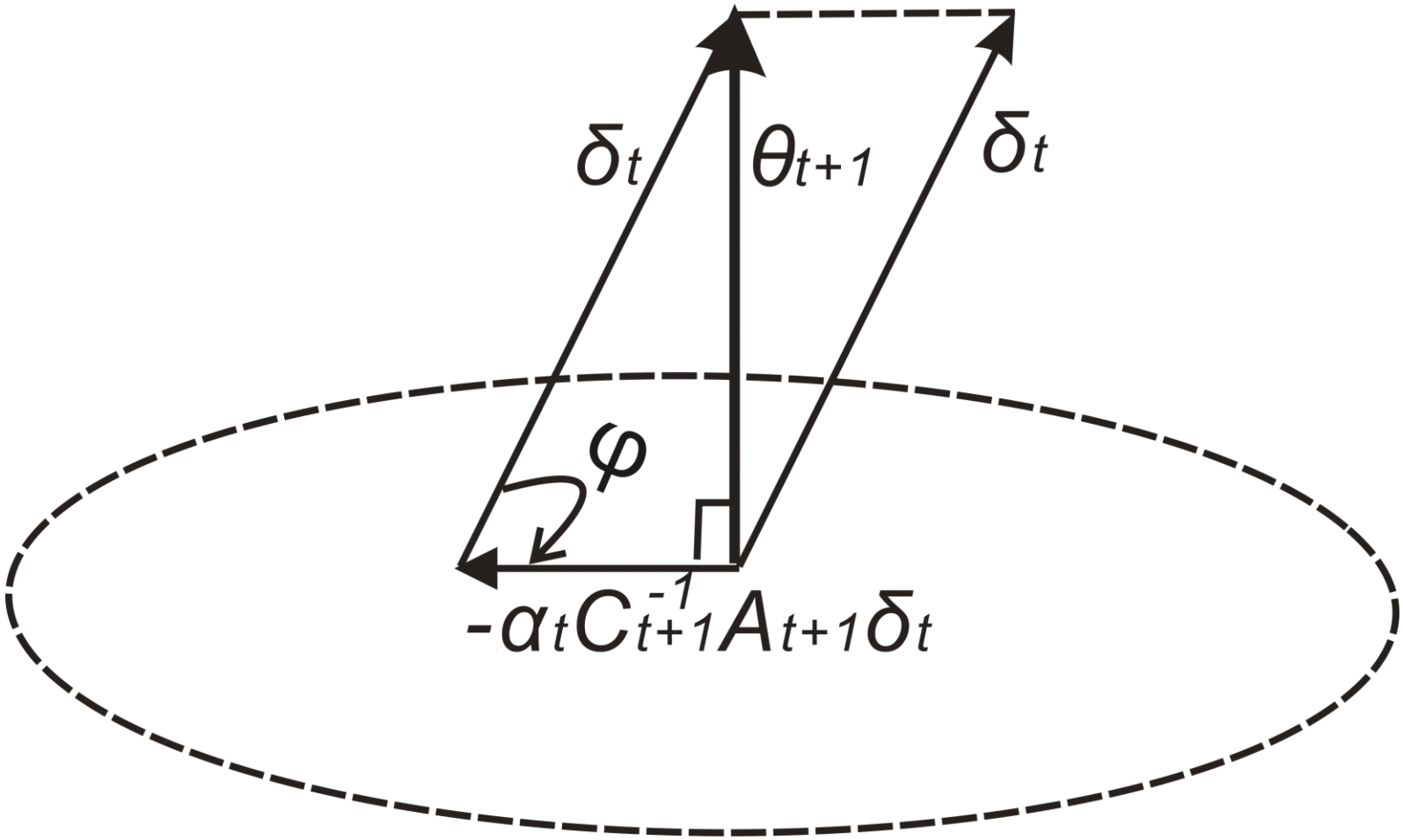
- Making

$$\| \theta_{t+1} \|^2 < \| \delta_{t+1} \|^2$$

gives us a step-size that can be computed on line:

$$\alpha_t = \frac{\delta_t' C_{t+1}^{-1} A_{t+1} \delta_t}{(C_{t+1}^{-1} A_{t+1} \delta_t)' (C_{t+1}^{-1} A_{t+1} \delta_t)}.$$

$$\alpha_t = \mathbf{Min} \, \| \theta_{t+1} \|^2, \quad \alpha \in \mathbf{R}$$



On-line Step-size for Incremental PTD alg's

$$\alpha_t = \frac{\delta_t' \Sigma_{t+1}^{-1} A_{t+1} \delta_t}{(\Sigma_{t+1}^{-1} A_{t+1} \delta_t)' (\Sigma_{t+1}^{-1} A_{t+1} \delta_t)}.$$

Efficient O(qK)

Sparse Nature of RL tasks

- RL tasks are sparse in two aspects:
  - Transitions are sparse P
    - *Nature of evolution events such as human relations, geological traits, wheather, sports, ...*
    - *[Moore & Atkeson 92]*
  - Linear function approximation is sparse Φ
    - *Lookup table representation*
    - *CMAC networks*
    - *Linear interpolation*

Matrix *A* is sparse!

- How sparse?
  - Boyan Chain, 801 states, A has dimension *40401*, but has only *1200* nonzero entries
  - Cart-pole balancing task. Using a CMAC M=4, C=7,A has dimension $(4^4 * 7)^2 = 3211264$, but has only about *15000* nonzero entries.
- ***Removing 0s!***
  - save memory
  - efficient computation

Compressed Sparse Row (CSR)

- CSR representation $Z_t(a_t, c_t, d_t)$, $\boldsymbol{Q_t}$: $\boldsymbol{A_t}$ **or** $\boldsymbol{C_t}$
  - a_t is a real array containing all the real values of the nonzero elements of Q_t,
  - c_t is an integer array containing the column indices of the elements stored in a_t,
  - d_t is an integer array containing the pointers to the beginning of each row in at and c_t.

Experimental Results

Effects of Preconditoning

Un-preconditioned (Gradient Descent) Vs. Preconditioned

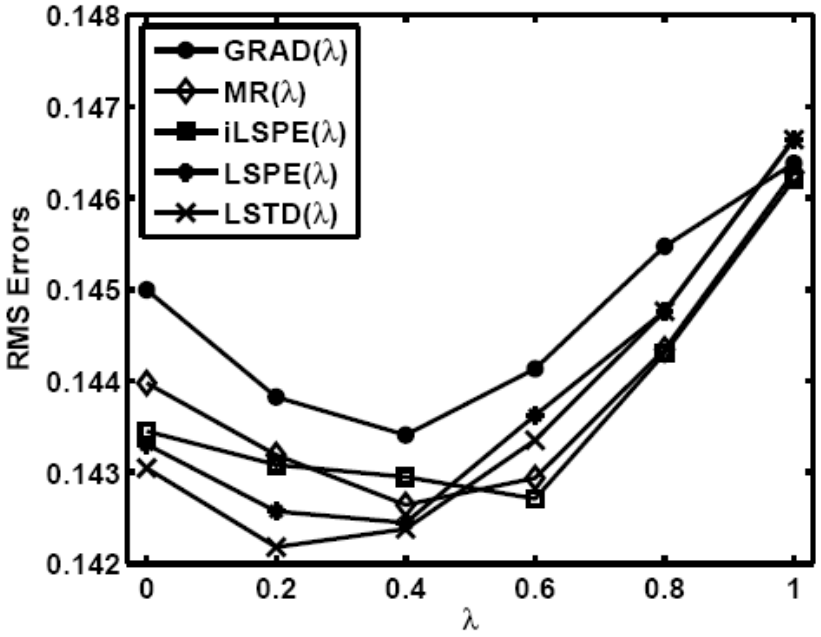


Figure 7. Comparison of convergence rate in terms of RMS errors at 900th state visit.

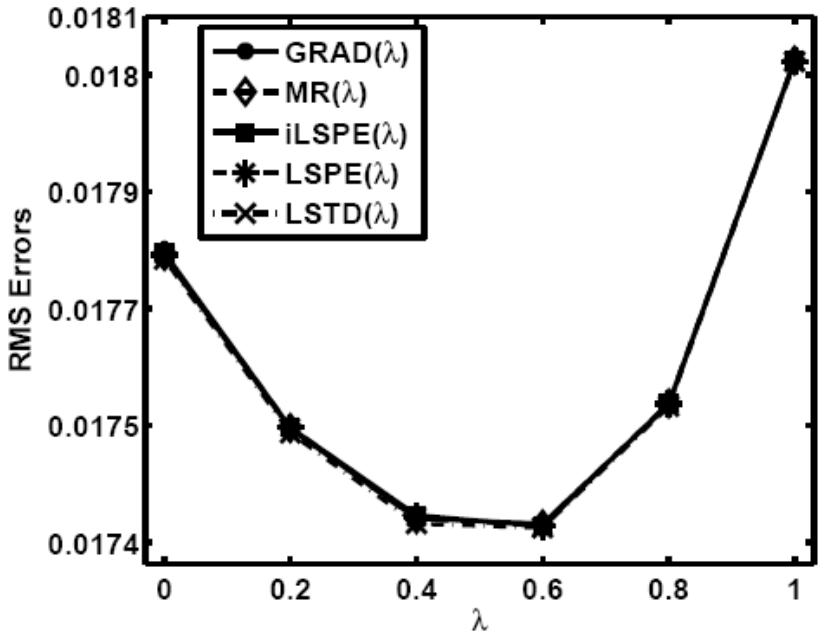


Figure 6. The RMS errors at 90000th visit by GRAD(λ), MR(λ), iLSPE(λ), LSPE(λ) and LSTD(λ).

Experimental Results

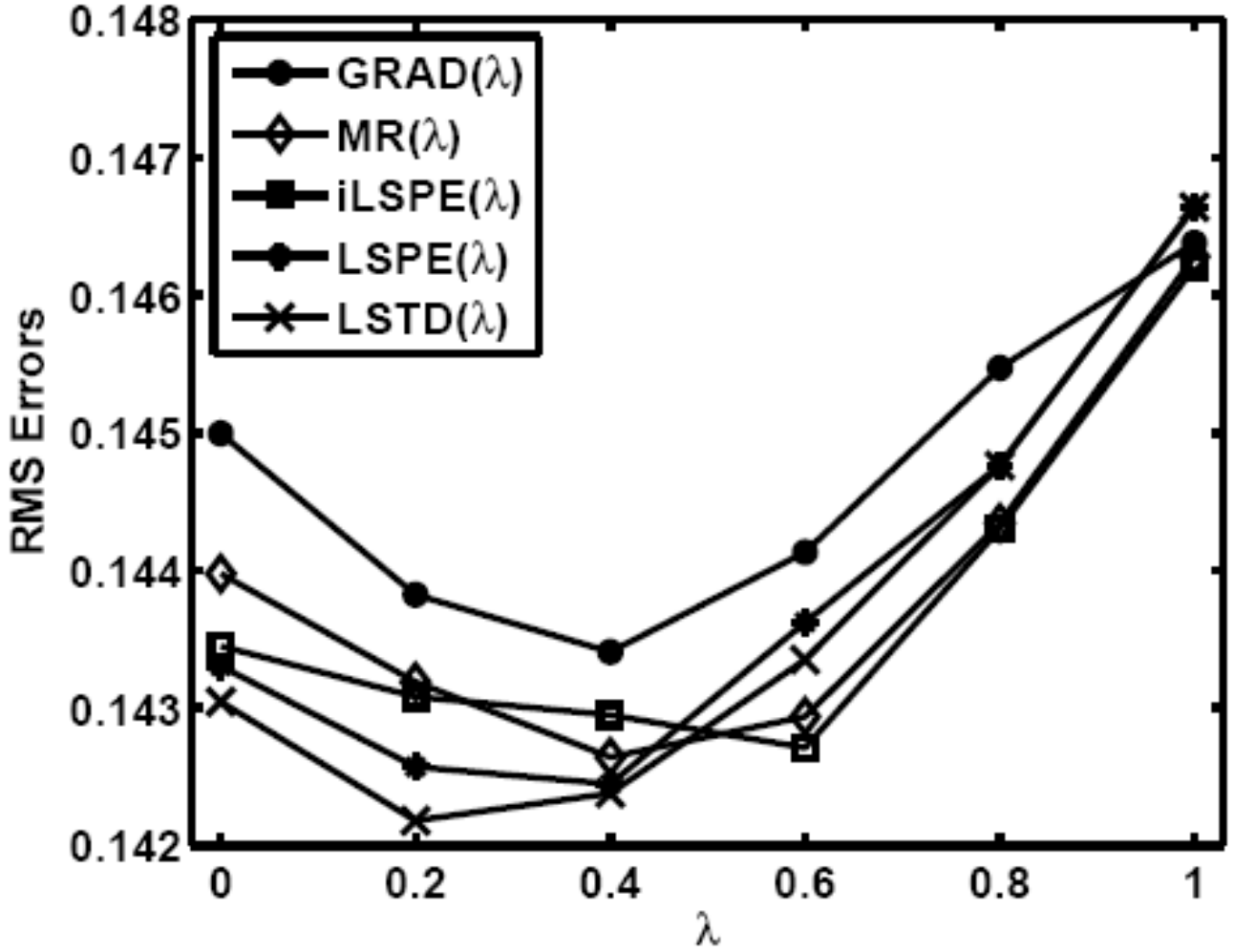
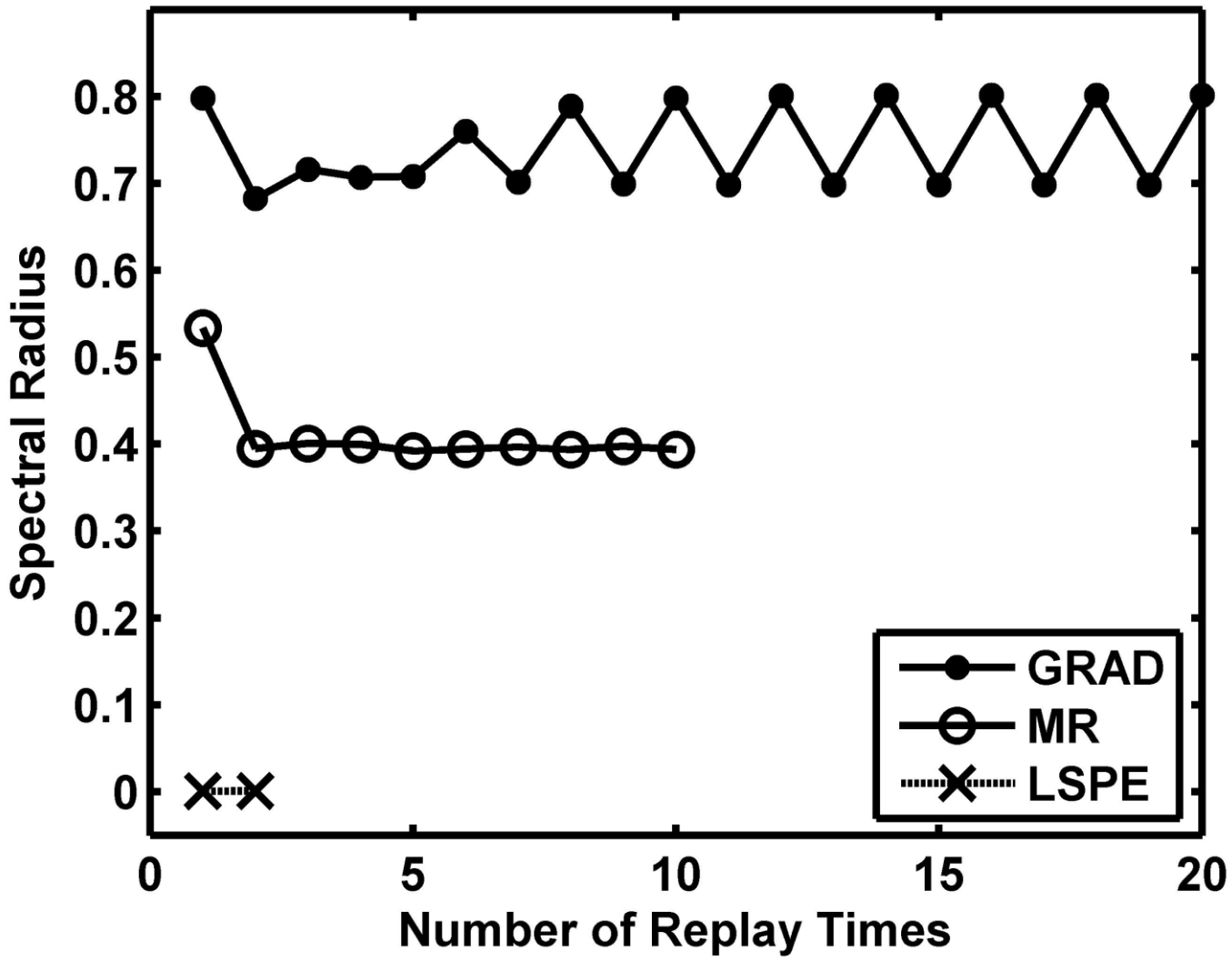


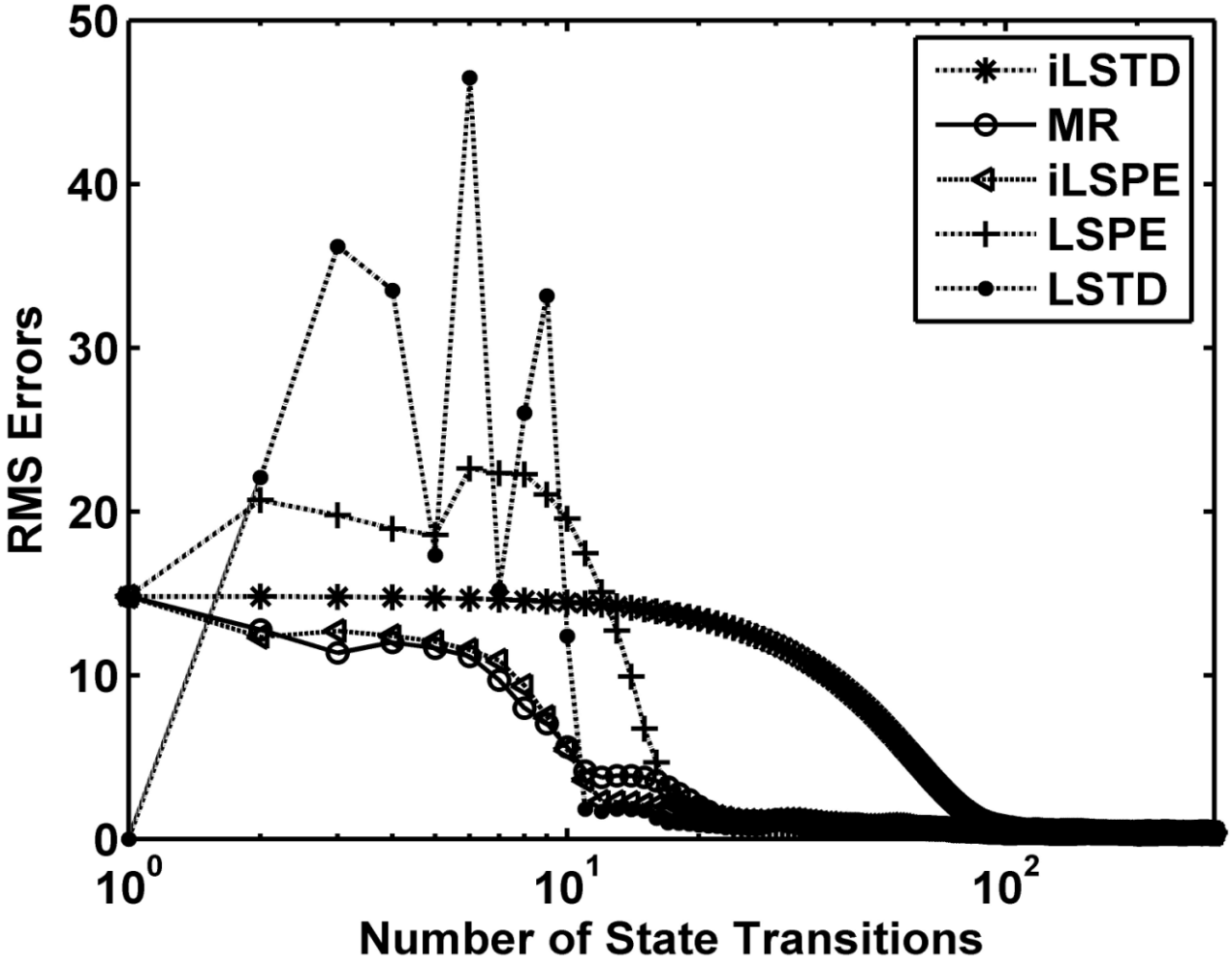
Figure 7. Comparison of convergence rate in terms of RMS errors at 900th state visit.

Why preconditoning is faster?

Spectral Radius



•On-line Stepsize outperform diminishing one
•LSTD is instable in the face of small amount of data; while incrmental Preconditioning alg's are stable with small amount of data, and as fast as LSTD when data is sufficient.



CPU time

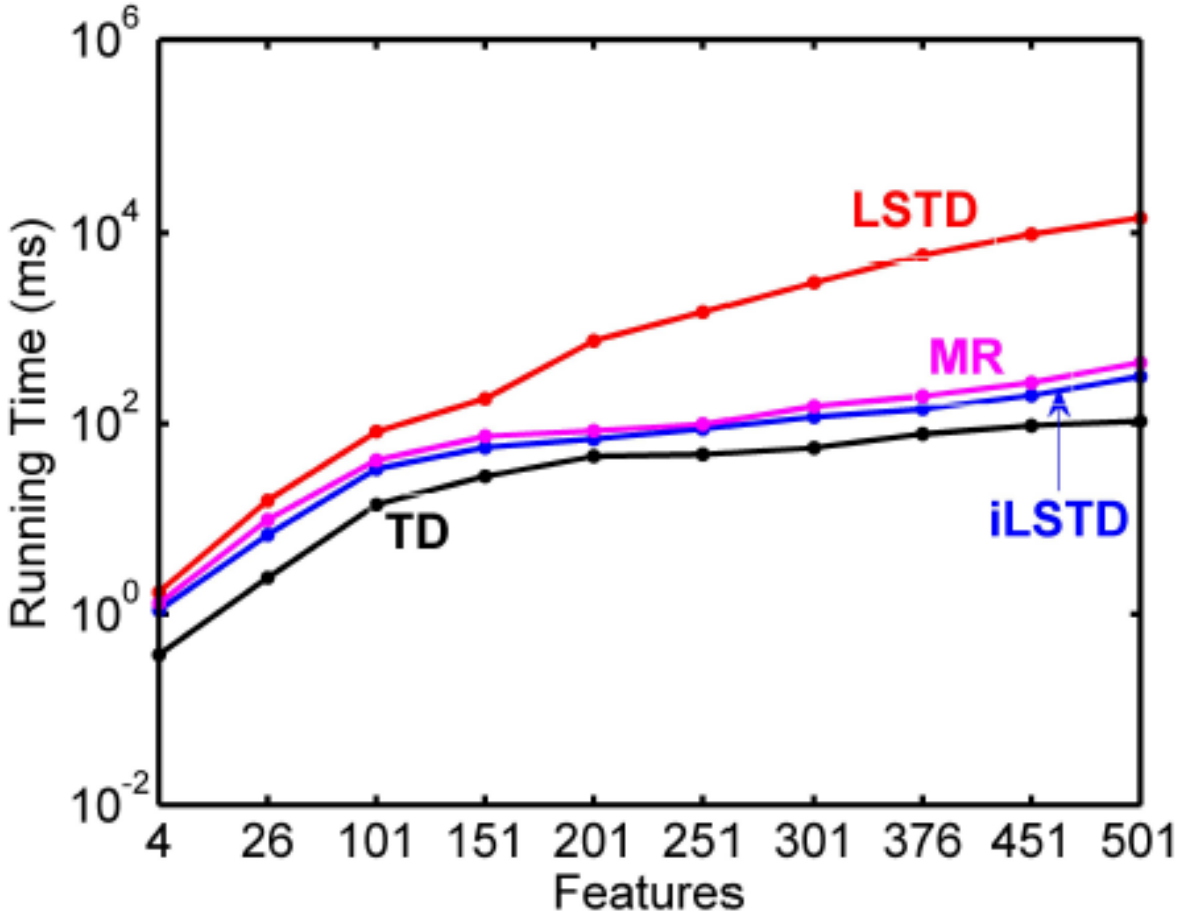


Figure 4: Comparison of running time per trajectory. Each point in the figure is averaged over 1000 trajectories.

Conclusion

- Preconditioning is an important powerful technique from numerical analysis that is very useful to RL.
- Existing policy evaluation algorihtms can be viewed as Preconditioning.
- Incremental Preconditioned alg's are faster and more stable than LSTD in the face of limited data.
- On-line Step-size performs much faster than diminishing step-size for iLSTD and other Preconditioned algorithms.
- Incremental PTD with CSR Representation has complexity near to that of TD for sparse RL tasks.

- Further
  - Convergence
  - Large Scale applications

Thanks!