Adaptive λ Least-Squares Temporal Difference Learning

E1-277 Reinforcement Learning Course Project

Braghadeesh L, Sethupathy P

Instructors : Prof Shalabh Bhatnagar
Prof Gugan Thoppe
Mentor : Dr. Vinayaka Yaji

IISc, Bangalore

May 30, 2020

Outline

Introduction

- In many practical applications of RL, Model of the environment is not available
- Agent learns from the environment, often by sampling state transitions and rewards
- **Naive ways:**
  - Monte Carlo Update: For episodic tasks. Unbiased estimate, but high variance
  - Temporal Difference Update: For continuing tasks. Low Variance, but high bias.
- We face Bias-Variance trade-off

Can we do better?

TD(λ)

- TD(λ) effectively overcomes the Bias-Variance trade off, by defining λ return
- To handle large state space, we use linear function approximation of value function
- λ return is given by $G_t^{(\lambda)} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_{t:t+n}$

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1}V(S_{t+n}) \tag{1}$$

$$\sum_{n=1}^{\infty}(1-\lambda)\lambda^{n-1} = 1$$

- Let T be the terminal time step. G_t then becomes

$$\boxed{G_t^{(\lambda)} = (1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}G_{t:t+n} + \lambda^{T-t-1}G_t}$$ [3]

- For state S_t at time t, estimate of value function is $\hat{V}(S_t, \theta) = \theta^T \phi(S_t)$, where $\phi(S_t)$ is the feature vector of S_t

  $\theta,\ \phi(S_t) \in \mathbb{R}^d$

- θ is unknown. Estimate of $\hat{V}(S_t, \theta) \iff$ Estimate of θ

How do we estimate θ?

- Use sampled transitions $(S_t, A_t, R_{t+1}, S_{t+1})$ observed till time t as labels
- Minimize Mean Squared Value Error (MSVE), given by $\sum_s \mu(s)(V_\pi(s) - \hat{V}(s, \theta))^2$
- **Solution:** Stochastic Gradient Descent $\theta_{t+1} = \theta_t + \alpha(G_t^\lambda - \hat{V}(S_t, \theta_t))\nabla \hat{V}(S_t, \theta_t)$

Forward View TD(λ)

- For linear function approximation, update rule becomes:

$$\theta_{t+1} = \theta_t + \alpha(G_t^{\lambda} - \hat{V}(S_t, \theta_t))\phi(S_t) \tag{2}$$

- The forward view gives theoretical view of λ return
- One obviously cannot have n-step returns in hand to implement TD(λ).

How do we implement TD(λ)?

Backward View TD(λ)

- Eligibility trace is used to implement TD(λ) on-line

$$Z_t = \phi(S_t), \quad Z_{t+1} = \lambda\gamma Z_t + \phi(S_{t+1}) \tag{3}$$

- On-line update: $\theta_{t+1} = \theta_t + \alpha\delta_t Z_t$, where δ_t is one step TD error $\delta_t = R_{t+1} + \gamma\hat{V}(S_{t+1}, \theta_t) - \hat{V}(S_t, \theta_t)$

- Total error in on-line updates is: $(\sum_{k=t}^{\infty}(\lambda\gamma)^{k-t}\delta_k)\phi(S_t) = (G_t^\lambda - \hat{V}(S_t, \theta_t))\phi(S_t)$

- On-line update of TD(λ) converges to fixed point solution, given by

$$\mathbf{d} + \mathbf{c}\theta_\lambda = 0[2] \tag{4}$$

where, d and $\mathbf{c}$ are:

$$\mathbf{d} = \mathbb{E}\left[\sum_{i=0}^{T} z_i R_i\right] \tag{5}$$

$$\mathbf{c} = \mathbb{E}\left[\sum_{i=0}^{T} z_i(\phi(S_{i+1}) - \phi(S_i))^T\right] \tag{6}$$

Merits and Demerits of TD(λ)

Merits

- Balances Bias-Variance trade off by "suitable" choice of λ
- Per step computation in the iterative update is less
- Generalizes TD(0) and Monte-Carlo updates

Demerits

- TD(λ) is sensitive to the step size parameter α
- TD(λ) never make efficient use of data that it observed.
- TD(λ) is sensitive to $||\theta_{\lambda} - \theta_{initial}||$ [2]

Contd . . .

How to make efficient use of data?

- Do sample mean estimate of **d** and **c** over many trajectories
- Calculate θ_λ for the obtained estimates.
- By LLN, sample mean estimates converge to true mean with large #trajectories
- This gives rise to a new method, called Least Squares Temporal Difference (LSTD(λ)) and it depends on λ

Outline

LSTD(λ)

- Unlike TD(λ), LSTD(λ) doesn't perform iterative update and discard samples.
- LSTD(λ) directly computes θ_λ from (4) by estimating $\mathbf{c}$ and $\mathbf{d}$
- $\mathbf{c}$ is negative definite and hence $\mathbf{c}^{-1}$ exists.[1]
- For $\lambda = 1$, the solution obtained from LSTD(1) is same as solution obtained by linear regression.[2]

Have we solved Bias-Variance trade off?

Not Yet

Outline

LOTO-CV

- Recall: Mean Value Squared Error(MSVE)for the estimate of unknown θ is given by

$$\sum_{s}\mu(s)(V_{\pi}(s)-\phi(s)^{T}\theta_{\lambda})^{2} \tag{7}$$

- λ controls θ_{λ}

- Solution to trade off : value of $\lambda \in \Lambda \subseteq [0,1]$ that achieves minimum MSVE.

- Denote the sample mean estimates of -**c** and **d** as A_{λ} and b_{λ}, given by: [4]

$$\boxed{A_{\lambda}=\sum_{i=1}^{n}\sum_{t=1}^{H}z_{i,t}^{\lambda}w_{i,t}^{T}} \qquad \boxed{b_{\lambda}=\sum_{i=1}^{n}\sum_{t=1}^{H}z_{i,t}R_{i,t}} \tag{8}$$

where

$$\boxed{z_{i,j}=\sum_{t=1}^{H}(\lambda\gamma)^{j-t}x_{i,t}} \qquad \boxed{w_{i,t}=(x_{i,t}-\gamma x_{i,t+1})} \tag{9}$$

LOTO-CV

- LOTO-CV is used to find the best choices of λ
- LOTO-CV leaves out one trajectory and build sample mean estimates of -**c** and **d**.
- The above estimates are independent.
  Total variance $\propto 1/\#$Trajectories
- Fresh sample mean estimate of the above estimates is unbiased.
- As #Trajectories is increased, we get low variance.

Tackled Bias-Variance trade off!!!

LOTO-CV Implementation

- Fix a single value $\lambda \in [0, 1]$

$$\boxed{C_i = \sum_{j \neq i} \sum_{t=1}^{H} z_{j,t}^{\lambda} w_{j,t}^{T}} \tag{10}$$

$$\boxed{y_i = \sum_{j \neq i} \sum_{t=1}^{H} z_{j,t} R_{j,t}} \tag{11}$$

$$\boxed{\theta_i = C_i^{-1} y_i} [4] \tag{12}$$

(10) and (11) are the sample mean estimates of -**c** and **d** respectively, by leaving out i^{th} trajectory. (12) gives the estimate of θ for i^{th} trajectory.

- For each trajectory i, LOTO-CV error, which is MSVE for that trajectory, is calculated

$$l_i = \frac{1}{H}\sum_{t=1}^{H}\left(x_{i,t}^T\theta_i - \sum_{j=t}^{H}\gamma^{j-t}R_{i,j}\right)^2 \tag{13}$$

- (10) can also be re-written as

$$C_i = A_\lambda - \sum_{t=1}^{H} z_{i,t}^{\lambda} w_{i,t}^T \tag{14}$$

- (14) suggest use of Recursive Sherman-Morrison Update to calculate the inverse of C_i

Algorithm: RSM Update

1 **Require**: M, a *dxd* matrix, $\mathscr{D} = \{(u_t, v_t)\}_{t=1}^{T}$ a collection of $2T$ d dimensional column vectors

2 $\widetilde{M}_0 \leftarrow M$

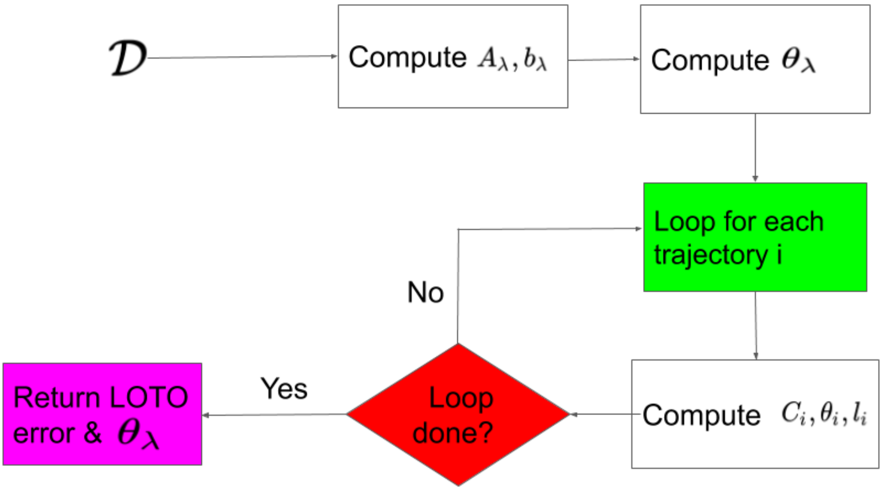
3 **for** $t = 1, 2, \ldots, T$ **do**

4 $$\widetilde{M_t} \leftarrow \tilde{M}_{t-1} - \frac{\widetilde{M}_{t-1} u_t v_t^T \widetilde{M}_{t-1}}{1 + v_t^T \widetilde{M}_{t-1} u_t}$$

5 **end**

6 **return** $\widetilde{M}_T$

LOTO-CV Implementation



LOTO-CV Implementation

- Naively, for each value in finite set of possible $\lambda \in [0, 1]$, Mean LOTO-CV error is calculated.
- Output λ with least Mean LOTO-CV error
- Naive implementation need inverse of A_λ, direct computation is expensive

Can we do better?

Outline

- A_λ can be re-written as:

$$\boxed{A_\lambda = \sum_{i=1}^{n}\sum_{t=1}^{H} u_{i,t} v_{i,t}^T + A_0} \tag{15}$$

[4] where $u_{i,t} = (z_{i,t}^{\lambda} - x_{i,t})$ and $v_{i,t} = (x_{i,t} - \gamma x_{i,t+1})$

- (15) implies direct inverse computation can be avoided
- Instead, use RSM update to compute inverse of A_λ
- For each λ, the LOTO errors are computed.
- Output λ with least Mean LOTO-CV error

How is it different from Naive implementation?

ALLSTD

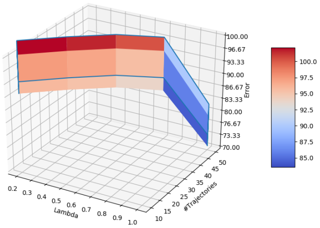
- For every λ in finite set, we exhaustively search for λ for which LOTO error is minimum.
- Each search involves computation of A_{λ}^{-1}, which now can be computed by RSM update
- Lesser computational difficulties enable expansion of search space of λ
- Naive LOTO-CV+LSTD takes $O(kd^3 + knHd^2)$
- ALLSTD takes only $O(kd^2 + knHd^2)$

Outline

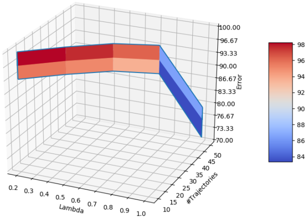
Experiment Setup

- We implemented Naive LOTO-CV+LSTD and ALLSTD on Mountain Car, 2048 game and Random Walk environments.
- We plotted the error plots of both the approaches vs #Trajectories and λ and compared them with TD.
- We also plotted the error bar plots for better visualization.
- For Mountain car setup: We took $\gamma = 1$ and
  For 2048 Random Walk setup : We took $\gamma = 0.95$
- #Trajectories $\in \{10, 20, 30, 40, 50\}$ and $\lambda \in [0.2, 0.4, 0.6, 0.8, 1]$
- We followed policies mentioned in [4]

Plots (Mountain Car Environment)



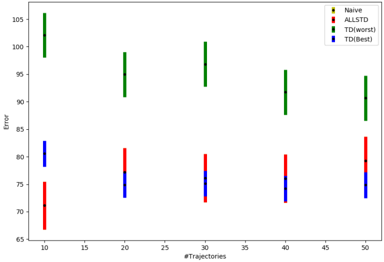
(a) Error Plot for Naive



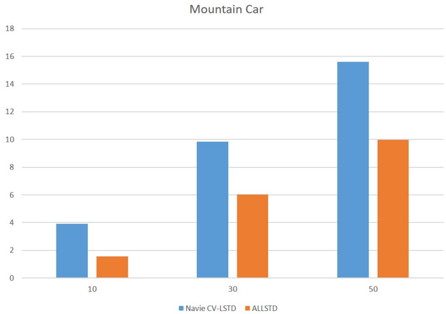
(b) Error Plot for ALLSTD

Figure: Error Plots Mountain Car Environment

Plots (Mountain Car Environment)

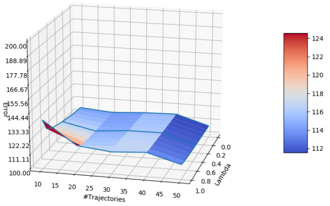


(a) Error Bar Plot

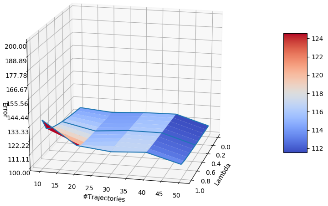


(b) Time Plot

Figure: Error Plots Mountain Car Environment



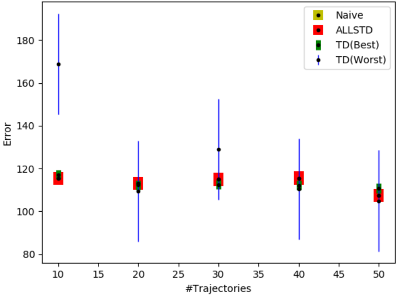
(a) Error Plot for Naive



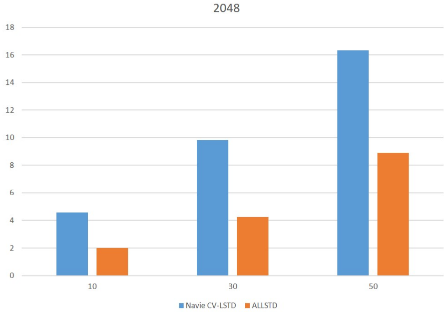
(b) Error Plot for ALLSTD

Figure: Error Plots 2048 Environment

Plots (2048 Environment)



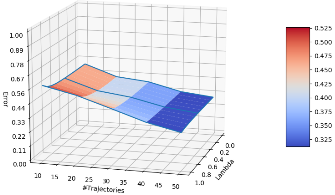
(a) Error Bar Plot



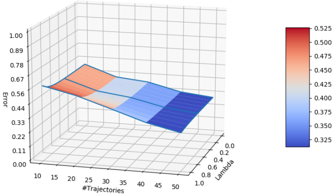
(b) Time Plot

Figure: Error Plots 2048 Environment

Plots (Random Walk Environment)



(a) Error Plot for Naive



(b) Error Plot for ALLSTD

Figure: Error Plots Random Walk Environment

Plots (Random Walk Environment)

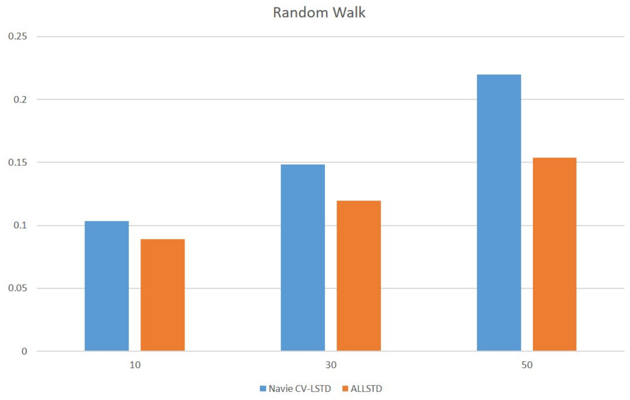


Figure: Time Plot

Inference from Experiments

- For Mountain car, Error is least for $\lambda = 1$ and highest for $\lambda << 1$.
- For 2048 and Random Walk, Error is least for $\lambda << 1$ and highest for $\lambda = 1$.
- Irrespective of Environment, ALLSTD beats Naive LOTO-CV+LSTD in time.
- Better error performance without knowledge of step size parameter,unlike in TD(λ)

Outline

Summary

- Studied LSTD(λ)
- Learnt how LSTD(λ) can be improved by choosing appropriate λ
- Studied Naive way to improve LSTD(λ)
- Addressed the limitation in the Naive way i.e, LOTO-CV+LSTD(λ)
- Learnt how RSM update can overcome the computational difficulty
- Successfully implemented the proposed ALLSTD algorithm

References

- Dimitri Bertsekas and John Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.
- Justin Boyan. "Technical Update: Least-Squares Temporal Difference Learning". In: *Springer* 39 (2002).
- Steven Bradtke and Andrew G. Barto. "Linear Least-Squares Algorithms for Temporal Difference Learning". In: *Computer Science Department Faculty Publication Series* 9 (1996).
- Timothy A. Mann et al. "Adaptive Lambda Least-Squares Temporal Difference Learning". In: *CoRR* (2016). arXiv: 1612.09465.

Thank You