Reproducing "Temporal Difference Updating without a Learning Rate"

Ritam Saha

June 8, 2025

1 Introduction

At the heart of reinforcement learning lies the problem of decision making under uncertainty. An agent must learn to choose actions that maximize cumulative reward over time, despite not knowing the consequences of its actions in advance. To do this effectively, the agent must estimate the expected future return of a given state or state-action pair—a process known as value estimation.

In tabular environments, where the state or state-action space can be explicitly enumerated, a prominent class of algorithms for value estimation is temporal difference (TD) learning. These methods update value estimates incrementally based on the agent's experience, using observed transitions to refine predictions in a sample-efficient, online manner. Algorithms such as TD(λ) [SB98] are widely used in these settings.

A common challenge in TD methods is their reliance on a manually tuned learning rate parameter α, which can have a significant impact on learning performance and often requires task-specific tuning. To address this, Hutter and Legg [HL07, Leg08] proposed HL(λ), a variant of TD(λ) that replaces α with a statistically derived update rule. This approach removes the need to manually tune the learning rate a while also outperforming previous optimized TD-based algorithms on both small and large Markov Reward Processes, as well as in full reinforcement learning settings.

In this report, we re-derive the learning rate update rule used in HL(λ) and replicate the experiments presented in their work to validate the reported results. Additionally, we perform further experiments to investigate and provide a potential explanation for an unexplained phenomenon that appears in one of their experiments.

2 Derivation

The definition of the future discounted reward of a state s is,

$$\overline{V}_s := \mathbb{E}\left[r_k + \gamma r_{k+1} + \gamma^2 r_{k+2} + \cdots \mid s_k = s\right],$$

where $r_k, r_{k+1}, r_{k+2}, ...$ are geometrically discounted by $\gamma < 1$. From this, we can also rewrite $\overline{V}_s$ as,

$$\overline{V}_s = \mathbb{E}[r_k + \gamma \overline{V}_{s_{k+1}} \mid s_k = s]. \tag{1}$$

Our goal at time t is to estimate V_s^t of $\overline{V}_s$ for each state s. The only information given in this environment is the history of states $s_1, s_2, ..., s_t$ visited at time t and their respective rewards $r_1, r_2, ..., r_t$. Equation (1) provides the intuition that if $r_k + V_{s_{t+1}}$ is larger than V_{s_t}, then the estimate V_s^t should be increased, and vice versa. This leads us to the following equation, TD(0).

$$V_{s_t}^{t+1} := V_{s_t}^t + \alpha\left(r_t + \gamma V_{s_{t+1}}^t - V_{s_t}^t\right) \tag{2}$$

A future version iterated upon TD(0) downsides of only iterating on only the previous visited state, by introducing TD(λ) which used eligibility traces to keep track of a heuristic of how many times and how recently a state was visited relative to the current time step t. The parameter λ is used to control the rate at which a state's eligibility trace is discounted.

$$E_s^t := \begin{cases} \gamma\lambda E_s^{t-1} & \text{if } s \neq s_t, \\ \gamma\lambda E_s^{t-1} + 1 & \text{if } s = s_t, \end{cases}$$

$$V_s^{t+1} := V_s^t + \alpha E_s^t \left(r + \gamma V_{s_{t+1}}^t - V_{s_t}^t \right). \tag{3}$$

To find a rule that can dynamically calculate the learning rate, we can think of deriving the equation that minimizes the distance between our estimated values, and our ground truth values V_s^t. To calculate our loss, we will use v_k, the empirical future discounted reward of s_k, which use the actual rewards that follow in the sequence rather than the expected rewards.

$$v_k := \sum_{u=k}^{\infty} \gamma^{u-k} r_u, \tag{4}$$

Given that computing v_k is impossible though, we can substitute it with calculating the future discounted reward for state s_k for $k < t$. This is possible since v_k is self consistent, meaning that it can be written by some immediate reward and the value function at a future time t. This allows us to reformat Equation (4) into the following.

$$v_k = \sum_{u=k}^{t-1} \gamma^{u-k} r_u + \gamma^{t-k} v_t \tag{5}$$

Now, since we have a bootstrapped metric for our future discounted reward, we can create a loss function, and then minimize it to obtain an update rule, from which we can extract an expression for the learning rate. In non-stationary environments, its also important to discount old evidence by a parameter $\lambda \in (0, 1]$.

$$L := \frac{1}{2} \sum_{k=1}^{t} \lambda^{t-k} \left(v_k - V_{s_k}^t \right)^2, \quad \lambda \in (0, 1] \tag{6}$$

$$\begin{aligned} \frac{\partial L}{\partial V_s^t} &= -\sum_{k=1}^{t} \lambda^{t-k} \left(v_k - V_{s_k}^t \right) \delta_{s_k s} \\ &= V_s^t \sum_{k=1}^{t} \lambda^{t-k} \delta_{s_k s} - \sum_{k=1}^{t} \lambda^{t-k} \delta_{s_k s} v_k \\ &= 0 \end{aligned}$$

We are able to transform $V_{s_k}^t$ into V_s^t by using the Kronecker $\delta_{s_k s}$ where $\delta_{xy} = 1$ if $x = y$ and $\delta_{xy} = 0$ if $x \neq y$.

To continue, we can recognize that $\sum_{k=1}^{t} \lambda^{t-k} \delta_{s_k s}$ represents the number of state visits for a specific state discounted by how long ago it occurred from the current time step t. When we substitute the summation with N_s^t, we get the following expression.

$$V_s^t N_s^t = \sum_{k=1}^{t} \lambda^{t-k} \delta_{s_k s} v_k \tag{7}$$

When we substitute v_k for the bootstrapped version defined in Equation (5) and rearrange the summations, we derive the following result.

$$\begin{aligned} V_s^t N_s^t &= \sum_{u=1}^{t-1} \sum_{k=1}^{u} \lambda^{t-k} \delta_{s_k s} \gamma^{u-k} r_u + \sum_{k=1}^{t} \lambda^{t-k} \delta_{s_k s} \gamma^{t-k} v_t \\ &= \sum_{u=1}^{t-1} \lambda^{t-u} \sum_{k=1}^{u} (\lambda\gamma)^{u-k} \delta_{s_k s} r_u + \sum_{k=1}^{t} (\lambda\gamma)^{t-k} \delta_{s_k s} v_t \\ &= R_s^t + E_s^t v_t \end{aligned}$$

Above, we define $E_s^t := \sum_{k=1}^{t} (\lambda\gamma)^{t-k} \delta_{s_k s}$ to be the eligibility trace of state s and $R_s^t := \sum_{u=1}^{t-1} \lambda^{t-u} E_s^u r_u$ to be the discounted reward with eligibility. To get an actual estimate we can use, we can bootstrap v_t to be our estimated V_s^t.

$$V_s^t N_s^t = R_s^t + E_s^t V_{s_t}^t \tag{8}$$

When $s = s_t$ this simplifies into the following.

$$V_{s_t}^t = \frac{R_{s_t}^t}{N_{s_t}^t - E_{s_t}^t}$$

Now, we can plug in this expression for $V_{s_t}^t$ into Equation (8) to get the following.

$$V_s^t N_s^t = R_s^t + E_s^t \frac{R_{s_t}^t}{N_{s_t}^t - E_{s_t}^t} \tag{9}$$

Although this gives us an expression for V_s^t, for our updating algorithm we would want to have a expression to compute the value for the next timestep. We can do this by replacing t with $t+1$ and use the definitions for N_s^{t+1}, E_s^{t+1}, and R_s^{t+1} to derive the following update rule.

$$\begin{aligned} N_s^{t+1} &= \lambda N_s^t + \delta_{s_{t+1} s}, & N_s^0 &= 0 \\ E_s^{t+1} &= \lambda\gamma E_s^t + \delta_{s_{t+1} s}, & E_s^0 &= 0 \\ R_s^{t+1} &= \lambda R_s^t + \lambda E_s^t r_t, & R_s^0 &= 0 \end{aligned}$$

$$\begin{aligned} V_s^{t+1} N_s^{t+1} &= R_s^{t+1} + E_s^{t+1} \frac{R_{s_{t+1}}^{t+1}}{N_{s_{t+1}}^{t+1} - E_{s_{t+1}}^{t+1}} \\ &= \lambda R_s^t + \lambda E_s^t r_t + E_s^{t+1} \frac{R_{s_{t+1}}^t + E_{s_{t+1}}^t r_t}{N_{s_{t+1}}^t - \gamma E_{s_{t+1}}^t}. \end{aligned}$$

We can then solve Equation (8) for R_s^t and plug its result in to get the following.

$$\begin{aligned} V_s^{t+1} N_s^{t+1} &= \lambda \left(V_s^t N_s^t - E_s^t V_{s_t}^t \right) + \lambda E_s^t r_t + E_s^{t+1} \frac{N_{s_{t+1}}^t V_{s_{t+1}}^t - E_{s_{t+1}}^t V_{s_t}^t + E_{s_{t+1}}^t r_t}{N_{s_{t+1}}^t - \gamma E_{s_{t+1}}^t} \\ &= \left(\lambda N_s^t + \delta_{s_{t+1} s} \right) V_s^t - \delta_{s_{t+1} s} V_s^t - \lambda E_s^t V_{s_t}^t + \lambda E_s^t r_t \\ &\quad + E_s^{t+1} \frac{N_{s_{t+1}}^t V_{s_{t+1}}^t - E_{s_{t+1}}^t V_{s_t}^t + E_{s_{t+1}}^t r_t}{N_{s_{t+1}}^t - \gamma E_{s_{t+1}}^t}. \end{aligned}$$

Now, we can simplify the equation by dividing by N_s^{t+1}, which is also equivalent to $\lambda N_s^t + \delta_{s_{t+1} s}$.

$$\begin{aligned} V_s^{t+1} = V_s^t &+ \frac{-\delta_{s_{t+1} s} V_s^t - \lambda E_s^t V_{s_t}^t + \lambda E_s^t r_t}{\lambda N_s^t + \delta_{s_{t+1} s}} \\ &+ \frac{\left(\lambda \gamma E_s^t + \delta_{s_{t+1} s} \right) \left(N_{s_{t+1}}^t V_{s_{t+1}}^t - E_{s_{t+1}}^t V_{s_t}^t + E_{s_{t+1}}^t r_t \right)}{\left(N_{s_{t+1}}^t - \gamma E_{s_{t+1}}^t \right) \left(\lambda N_s^t + \delta_{s_{t+1} s} \right)}. \end{aligned}$$

Now, if we make the denominators equivalent, and expand the second term, we get the following.

$$V_s^{t+1} = V_s^t + \frac{\lambda E_s^t r_t N_{s_{t+1}}^t - \lambda E_s^t V_{s_t}^t N_{s_{t+1}}^t - \delta_{s_{t+1}s} V_s^t N_{s_{t+1}}^t - \lambda\gamma E_{s_{t+1}}^t E_s^t r_t}{\left(N_{s_{t+1}}^t - \gamma E_{s_{t+1}}^t\right)\left(\lambda N_s^t + \delta_{s_{t+1}s}\right)}$$
$$+ \frac{\lambda\gamma E_{s_{t+1}}^t E_s^t V_{s_t}^t + \gamma E_{s_{t+1}}^t V_s^t \delta_{s_{t+1}s} + \lambda\gamma E_s^t N_{s_{t+1}}^t V_{s_{t+1}}^t - \lambda\gamma E_s^t E_{s_{t+1}}^t V_{s_t}^t}{\left(N_{s_{t+1}}^t - \gamma E_{s_{t+1}}^t\right)\left(\lambda N_s^t + \delta_{s_{t+1}s}\right)}$$
$$+ \frac{\lambda\gamma E_s^t E_{s_{t+1}}^t r_t + \delta_{s_{t+1}s} N_{s_{t+1}}^t V_{s_{t+1}}^t - \delta_{s_{t+1}s} E_{s_{t+1}}^t V_{s_t}^t + \delta_{s_{t+1}s} E_{s_{t+1}}^t r_t}{\left(N_{s_{t+1}}^t - \gamma E_{s_{t+1}}^t\right)\left(\lambda N_s^t + \delta_{s_{t+1}s}\right)}.$$

After canceling out equivalent terms where we assume $x = y$ in all Kronecker δ_{xy}, and factoring out E_s^t, we get the following for the righthand side of the expression.

$$V_s^t + \frac{E_s^t\left(\lambda r_t N_{s_{t+1}}^t - \lambda V_{s_t}^t N_{s_{t+1}}^t + \gamma V_s^t \delta_{s_{t+1}s} + \lambda\gamma N_{s_{t+1}}^t V_{s_{t+1}}^t - \delta_{s_{t+1}s} V_{s_t}^t + \delta_{s_{t+1}s} r_t\right)}{\left(N_{s_{t+1}}^t - \gamma E_{s_{t+1}}^t\right)\left(\lambda N_s^t + \delta_{s_{t+1}s}\right)}.$$

When dividing by $\lambda N_s^t + \delta_{s_{t+1}s}$, we derive our final update rule where $\beta_t(s,\, s_{t+1})$ which replaces the original learning rate parameter α in Equation (3) is defined below.

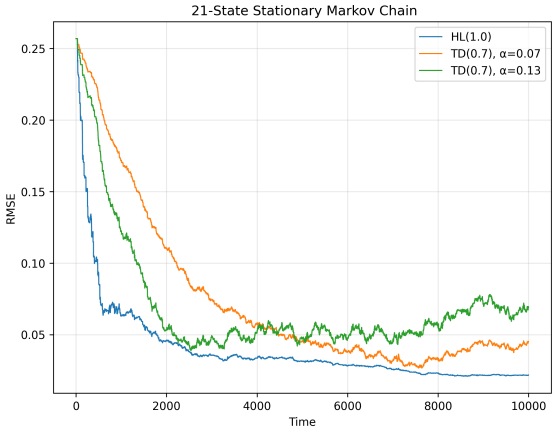
$$\beta_t(s,\, s_{t+1}) := \frac{1}{N_{s_{t+1}}^t - \gamma E_{s_{t+1}}^t} - \frac{N_{s_{t+1}}^t}{N_s^t}$$

$$V_s^{t+1} = V_s^t + E_s^t \beta_t(s,\, s_{t+1})\left(r_t + \gamma V_{s_{t+1}}^t - V_{s_t}^t\right),$$

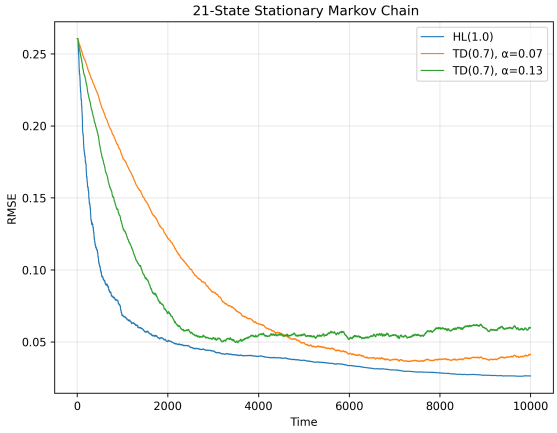
3 Experiments

3.1 21-State Stationary Markov Process

We test HL(λ) against TD(λ) on a stationary 21-state Markov Reward Process. States 0-19 give no reward and have a equal probability to move from state s to state $s+1$ and state $s-1$. State 0 and 20 always transition to state 10, with the transition from 0 to 10 giving a reward of 1.0, and the transition from 20 to 10 giving a reward of -1.0. We observe that HL(λ) is superior to TD(λ) is used with optimized parameters.



(a) HL(λ) vs. TD(λ) RMSE averaged over 10 runs

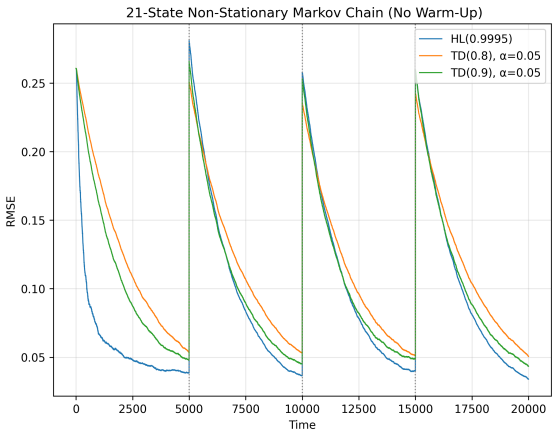


(b) HL(λ) vs. TD(λ) RMSE averaged over 100 runs

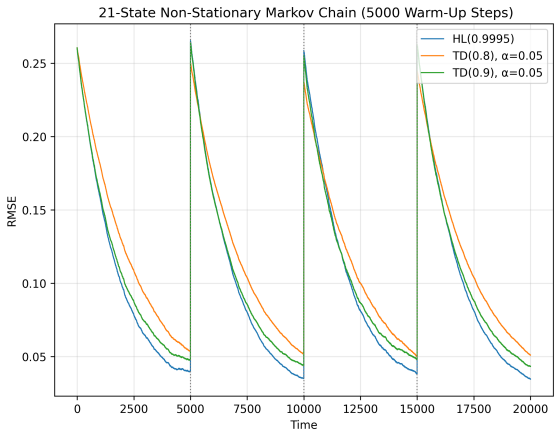
3.2 21-State Non-Stationary Markov Process

In this environment, we use the 21-state Markov Reward Process defined earlier. Every 5,000 steps, we switch the reward from state 20 to state 10, changing it from -1.0 to 0.5.

In the original paper, it was noted that HL(λ) learned quickly in the first iteration, but in later iterations its performance resembled that of TD(λ). The cause of this early advantage was unclear. To investigate, I conducted an additional experiment where I warmed up the state visitation counters over the first 5,000 steps. These counters significantly influence the learning rate used at each step. After applying the warm-up, the previously observed discrepancy disappeared. This suggests that the instability in early HL(λ) learning was due to the lack of convergence in the discounted state counters and eligibility traces. Further exploratory analysis was conducted to examine whether there were sharp spikes in the β values, which influence learning updates. In the non-warmed-up case, such spikes were indeed observed, further supporting the conclusion that unstable learning dynamics in the early phase were due to volatile, unconverged learning rate signals.



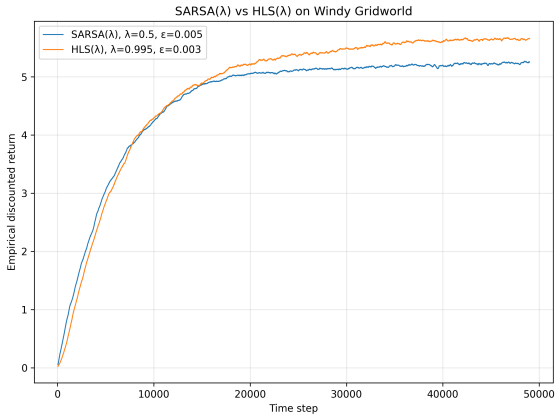
(a) HL(λ) vs. TD(λ) without warm-up



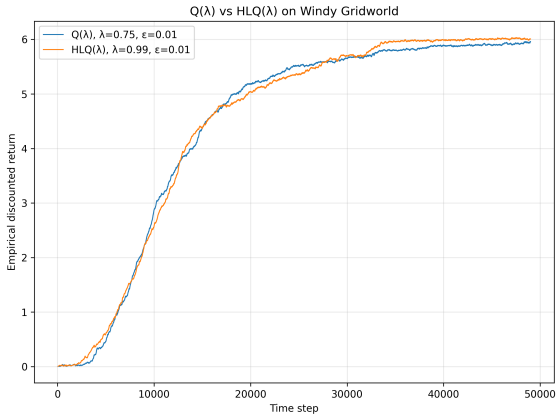
(b) HL(λ) vs. TD(λ) with a 5,000 step warm-up

3.3 Windy Gridworld

For our final test, we replace the TD(λ) implementations in Sarsa(λ) and Q(λ) with HL(λ), creating HLS(λ) and HLQ(λ), respectively. We evaluate these reinforcement learning algorithms on the Windy Gridworld environment—a 7×10 grid in which the agent can move up, down, left, or right. Certain columns in the grid introduce wind effects that push the agent *upward*, making navigation more difficult. Columns 4, 5, 6, and 9 apply a wind that moves the agent up by 1 row, while columns 7 and 8 apply a stronger wind that moves the agent up by 2 rows. The agent begins each episode in the 4th row of the 1st column and receives a reward of 1 upon reaching the goal in the 4th row of the 8th column[HL07, SB98]. To model a continuing task, the agent is automatically returned to the start state after reaching the goal. In this environment, we also observe superior performance over both Sarsa(λ) and Q(λ), though the performance gap is much more evident in Sarsa(λ).



(a) HLS(λ) vs. Sarsa(λ)



(b) HLQ(λ) vs. Q(λ)

4 Conclusion

This project focused on re-deriving and implementing the HL(λ) algorithm, a version of temporal difference learning that eliminates the need for a manually tuned learning rate. I worked through the full derivation from the original paper and replicated several key experiments to compare HL(λ) with standard TD(λ) approaches. The results confirmed the algorithm's strong performance, especially in environments with changing dynamics or delayed rewards.

One of the most challenging parts of the project was building the algorithm from scratch with no existing code or implementation to refer to. The original paper was mathematically dense and sometimes ambiguous, so translating the theory into a working implementation required a lot of careful reasoning, trial and error, and debugging.

Along the way, I learned a lot about the mechanics of temporal difference learning, especially how eligibility traces and state visitation counts can be used to construct a dynamic learning rate. I also gained a much better understanding of how theoretical ideas translate into real learning behavior—especially when things don't work as expected. Investigating and explaining the strange behavior seen early in training in non-stationary environments turned out to be one of the most rewarding parts of the project.

References

[HL07] Marcus Hutter and Shane Legg. Temporal difference updating without a learning rate. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 20, Vancouver, Canada, 2007.

[Leg08] Shane Legg. *Machine Super Intelligence*. PhD thesis, Università della Svizzera italiana, Lugano, Switzerland, 2008. Ph.D. thesis, supervised by Marcus Hutter.

[SB98] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.