CS780: Deep Reinforcement Learning

Mid-Semester Exam

Name: James Bond
Roll NO.: 0000007

Solution to Problem 1: Random-Maze Environment Implementation

The Random Maze Environment is represented as a grid world, as shown in Figure 1. The environment consists of 11 states, including a start state (S), a goal state (G), a hole state (H), and a wall. The agent receives rewards based on its actions and the state it transitions to. The transitions in the environment are stochastic, with four possible actions: left, top, right, and bottom.

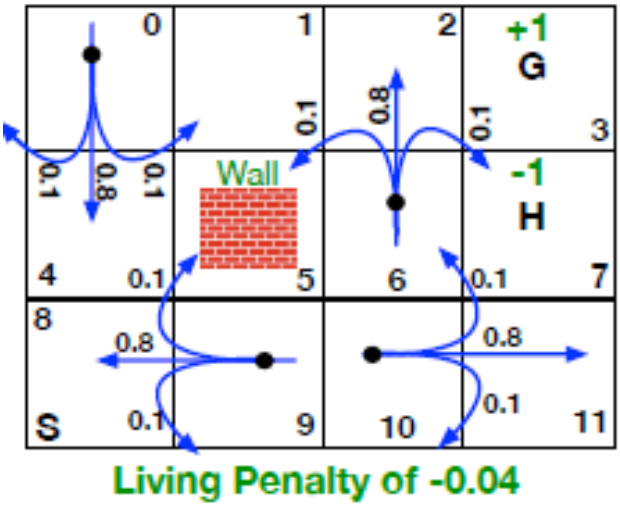


Figure 1: Random Maze Environment

- State Representation: The states in the environment are represented as follows:
  - Start State (S)
  - Goal State (G)
  - Hole State (H)
  - Wall
  - Non-terminal states (0 to 11)
- Rewards: The agent receives the following rewards:
  - +1 when reaching the goal state (G)
  - -1 when reaching the hole state (H)
  - Living Penalty of -0.04 for every non-terminal state
- Transitions: The transitions in the environment are stochastic, with 80% chance of going in the intended direction and 20% chance of going in either of the orthogonal directions.

  Description: This test case generates sample trajectories to be used for learning the value function.

  1. Set the seed for reproducibility.

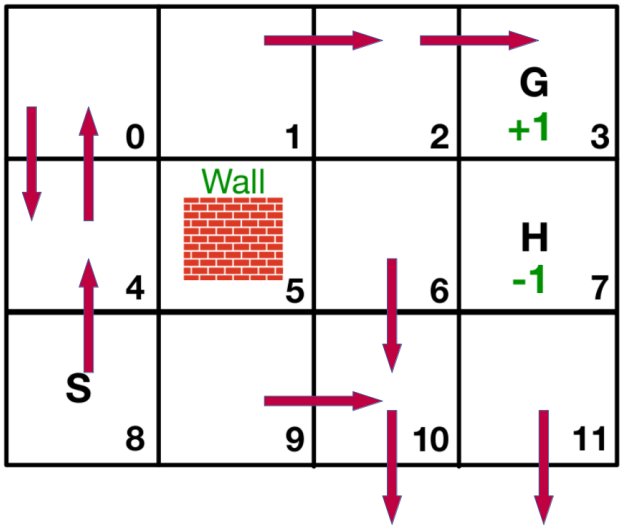
2. Reset the environment.
3. Call the `generateTrajectory` function with the policy and the maximum number of steps.
4. Print the generated trajectory.

Solution to Problem 2: RME Optimal Policy via Dynamic Programming

1. Random Policy Initialization

In this section, we describe the random policy initially chosen for the Random Maze environment. The random policy is implemented using the function `initialize_random_policy`.

The random policy is represented as a 2D array, where each row corresponds to a state, and the entries are probabilities of taking each action. The random policy is then visualized using a hand-drawn diagram with arrows indicating the chosen actions, as depicted in Figure.

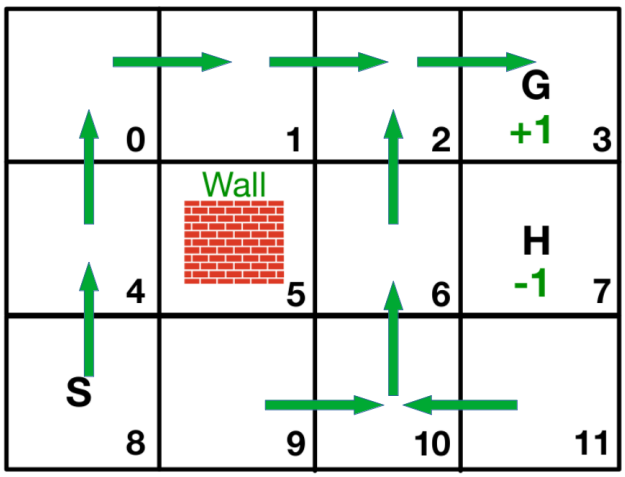


The arrows in the diagram represent the chosen actions for each state in the random policy. This random policy serves as the initial policy for the Policy Iteration algorithm.

Policy Iteration

The Policy Iteration algorithm is applied to find the optimal policy for the Random Maze environment. The main steps of Policy Iteration include policy evaluation and policy improvement.

The algorithm converges to an optimal policy, and the results include the final optimal policy represented as a 1D array `pi`. The optimal policy is then visualized with arrows indicating the chosen actions, as shown in Figure.



The optimal policy diagram visually represents the actions chosen for each state in the final policy obtained through Policy Iteration.

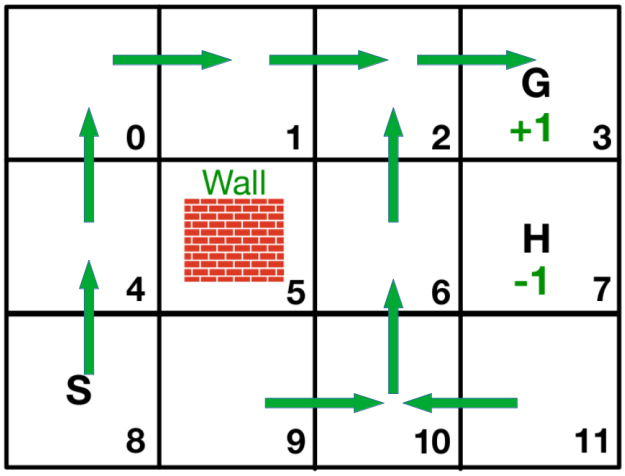
Convergence Details

The Policy Iteration algorithm converged after a certain number of iterations. The total number of iterations required for convergence is 5103.

2. Value Iteration

The Value Iteration algorithm is applied to find the optimal policy for the Random Maze environment. The main steps of Value Iteration include iteratively updating the value function and computing the action-value function based on the Bellman equation.

The algorithm converges to an optimal policy, and the results include the final optimal policy represented as a 1D array `pi`. The optimal policy is then visualized with arrows indicating the chosen actions.



The optimal policy diagram visually represents the actions chosen for each state in the final policy obtained through Value Iteration.

Convergence Details

The Value Iteration algorithm converged after a certain number of iterations. The total number of iterations required for convergence is 2116.

3. Comparison of PI and VI Convergence

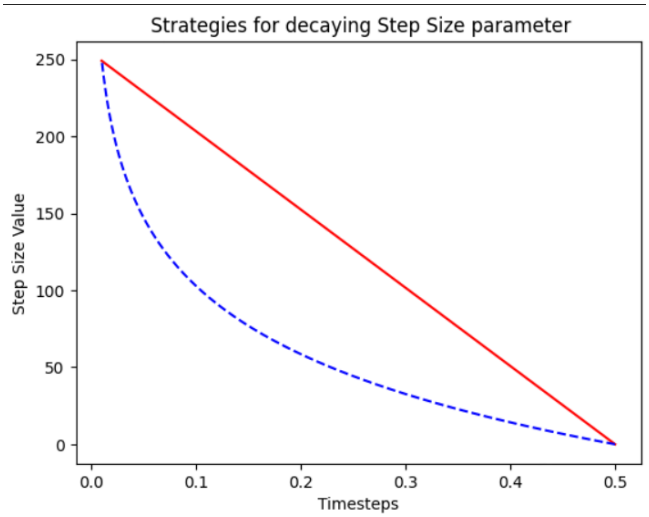
In the random maze environment, Value Iteration (VI) exhibited faster convergence with 2116 iterations, while Policy Iteration (PI) required 5103 iterations. The discrepancy in iteration counts underscores VI's efficiency in reaching convergence.

In general, the optimal policy obtained via PI and VI may not always be the same. Algorithmic disparities and convergence paths can lead to variations in resulting policies. The observed differences highlight the importance of considering specific environment characteristics when choosing between PI and VI.

Solution to Problem 3: RME Prediction with MDP Unknown

1. generateTrajectory function implemented in the code and it generates sample trajectories for learning the value function in the Random Maze Environment (RME). It returns a list of state, action, next state, reward tuples.The maxSteps parameter is used to terminate the episode if it exceeds a certain count. If the episode terminates early, the partial trajectory is discarded and an empty list is returned.

2. Function implemented in Notebook and plots are fillowing.



3. Monte Carlo Prediction can work in both First-Visit Monte Carlo (FVMC) and Every-Visit Monte Carlo (EVMC) settings, depending on the firstVisit parameter. The accuracy and convergence speed of the Monte Carlo method depend on the number of episodes (noEpisodes), the length of each episode (maxSteps), and the choice of the step size parameter (alpha).

   The Monte Carlo Prediction algorithm showcased its ability to estimate the value function effectively for the RME. The convergence patterns and gradual refinement of state values demonstrate the algorithm's suitability for predicting state values in a stochastic environment.

4. The Temporal Difference (TD) Prediction algorithm is implemented to solve the prediction problem for the Random Markov Environment (RME). It iterates through each episode and updates the value function estimates based on the temporal difference error, which is the difference between the TD target and the current estimate of the value function for each state.

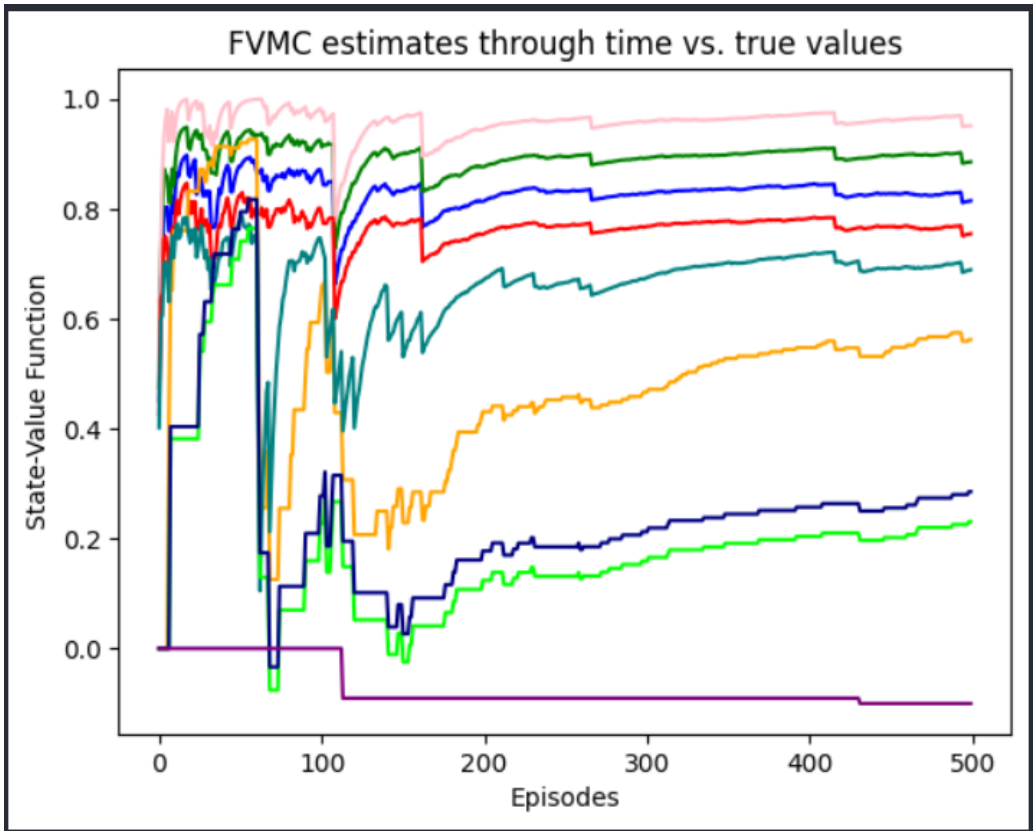
5. The n-Step Temporal Difference (TD) Learning algorithm was implemented to estimate state values in the Random Maze Environment. The algorithm leverages n-Step updates to balance short-term and long-term considerations when estimating state values.
   The n-Step TD Learning algorithm was tested with 500 episodes. The evaluation focused on the convergence patterns and the accuracy of state value estimation.
   By considering the last 3 steps for updating state values, it provided a nuanced estimation capturing temporal dependencies in the environment.
   The n-Step Temporal Difference Prediction algorithm offers a balanced approach, efficiently capturing both short-term and long-term aspects. The computational efficiency and nuanced estimation make it a favorable choice for predicting state values in dynamic environments.

6. The TD(λ) algorithm combines elements of Monte Carlo and Temporal Difference (TD) learning, utilizing eligibility traces to update the value function. It iteratively updates state values based on TD error and eligibility traces.

7. True Values:

   (a) State 0 - 5/6
   (b) State 1 - 4/6
   (c) State 2 - 3/6
   (d) State 4 - 2/6
   (e) State 6 - 1/6
   (f) State 8 - 1/6
   (g) State 9 - 1/6
   (h) State 10 - 1/6
   (i) State 11 - 1/6

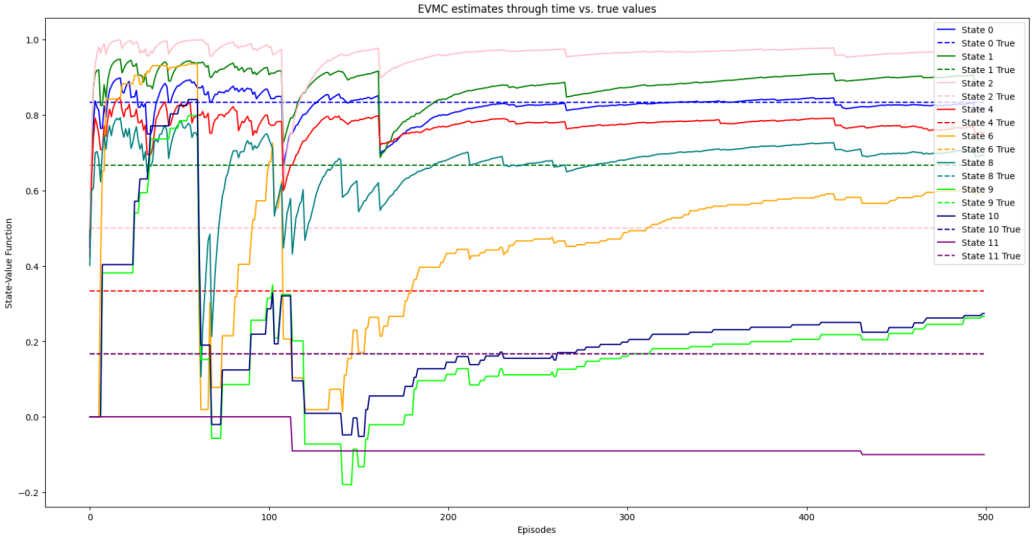
8. The Monte Carlo First-Visit Prediction algorithm was employed to estimate state values in the Random Maze Environment. The algorithm provides estimates for each non-terminal state based on the first visit to that state in each episode.
   The algorithm was tested for a maximum of 500 episodes. The learning rates (α) varied according to the specified schedule. The state-value estimates for each non-terminal state were tracked throughout the episodes.
   Figure presents the Monte Carlo First-Visit estimates for selected non-terminal states throughout the episodes. True values are plotted for comparison. The color-coded lines correspond to different non-terminal states.



Observing the plot, we notice that the Monte Carlo First-Visit estimates converge towards the true values for each non-terminal state. The varying learning rates contribute to the exploration-exploitation balance, allowing the algorithm to refine its estimates over time.

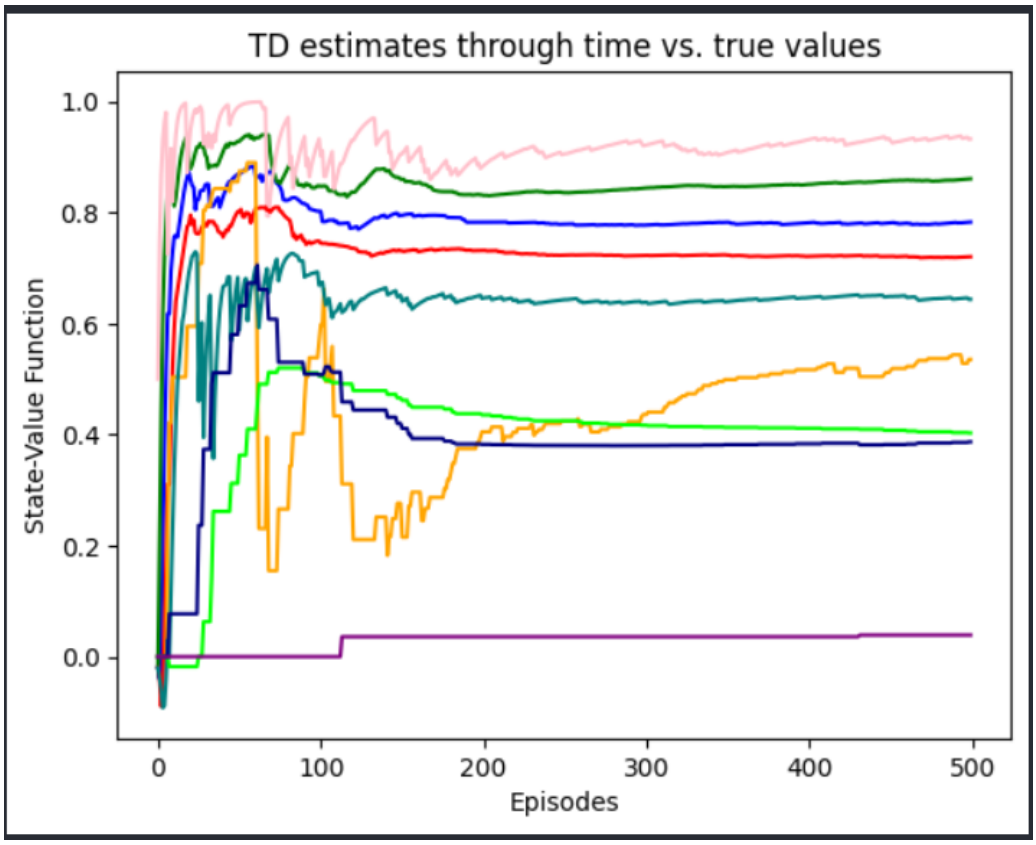
The Monte Carlo First-Visit Prediction algorithm effectively estimates state values in the Random Maze Environment. The chosen schedule for learning rates demonstrates a balanced exploration-exploitation strategy, leading to convergence.

9. EVMC (Every-Visit Monte Carlo) tends to produce estimates with higher variance compared to FVMC (First-Visit Monte Carlo) due to its inclusion of all occurrences of a state in the calculation, resulting in noisier estimates.
   EVMC may require more episodes to converge compared to FVMC, especially in environments with stochastic transitions or high variance in state values.
   The choice of step size parameter (α) can significantly affect the rate of convergence and stability of MC estimates. A small constant value of may result in slower but more stable convergence, while a decreasing exponential decay of can initially provide faster convergence followed by more stable estimates.



EVMC may provide more accurate estimates in certain scenarios but requires more computational resources due to its higher variance, while FVMC may offer faster convergence with smoother estimates but may overlook valuable information from multiple visits to states within episodes

10. The Temporal Difference (TD) Prediction algorithm was applied to estimate state values in the Random Maze Environment. The algorithm provides estimates for each non-terminal state based on temporal differences between successive state values.

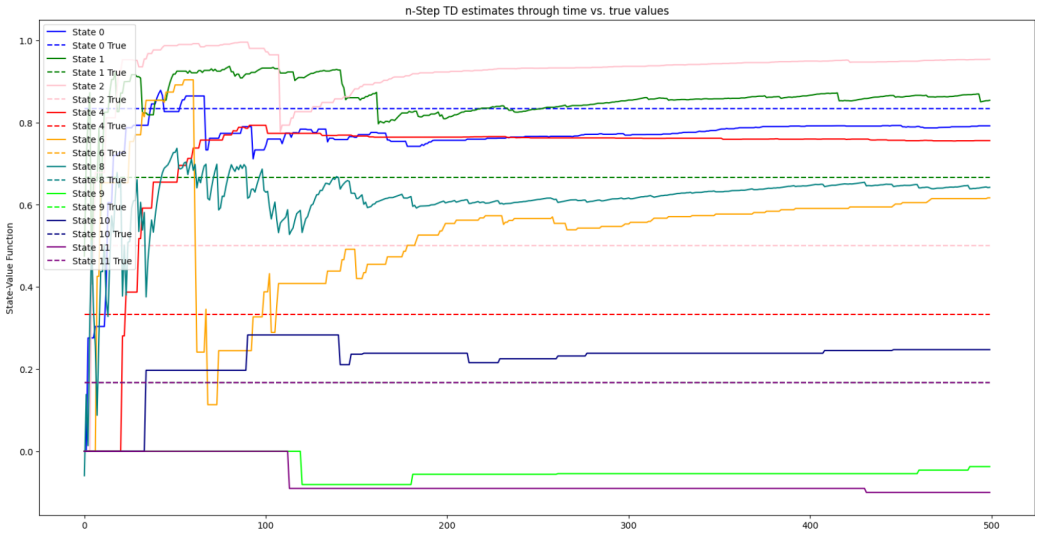


Observing Figure, we see that the Temporal Difference estimates converge towards the true values for each non-terminal state. The varying learning rates contribute to the exploration-exploitation balance, allowing the algorithm to refine its estimates over time.

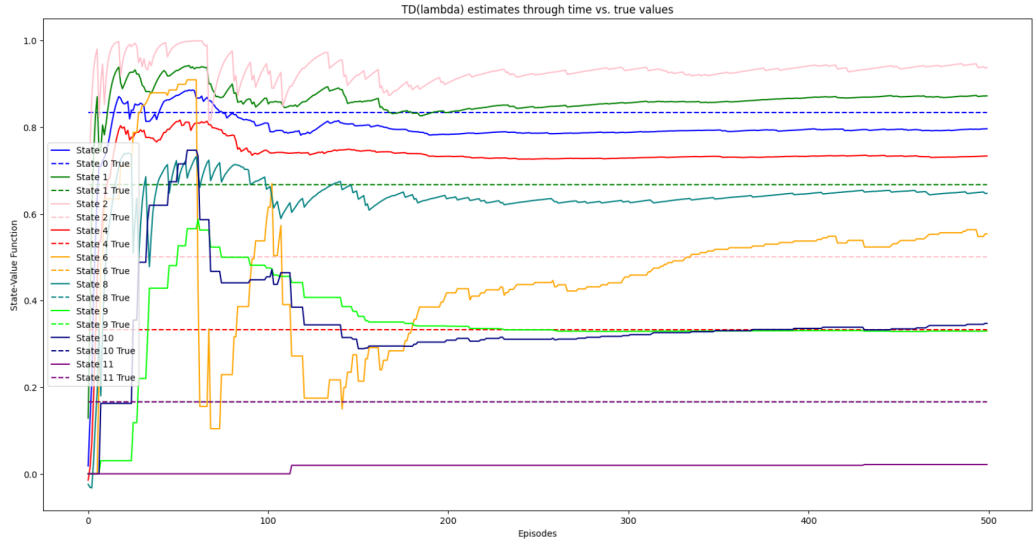
The exponential decrease in α followed by a constant value allows for more aggressive updates in the initial episodes, promoting exploration. As the episodes progress, the smaller α values ensure more stable and fine-tuned updates, aiding convergence.

The Temporal Difference Prediction algorithm effectively estimates state values in the Random Maze Environment. The chosen schedule for learning rates demonstrates a balanced exploration-exploitation strategy, leading to convergence.

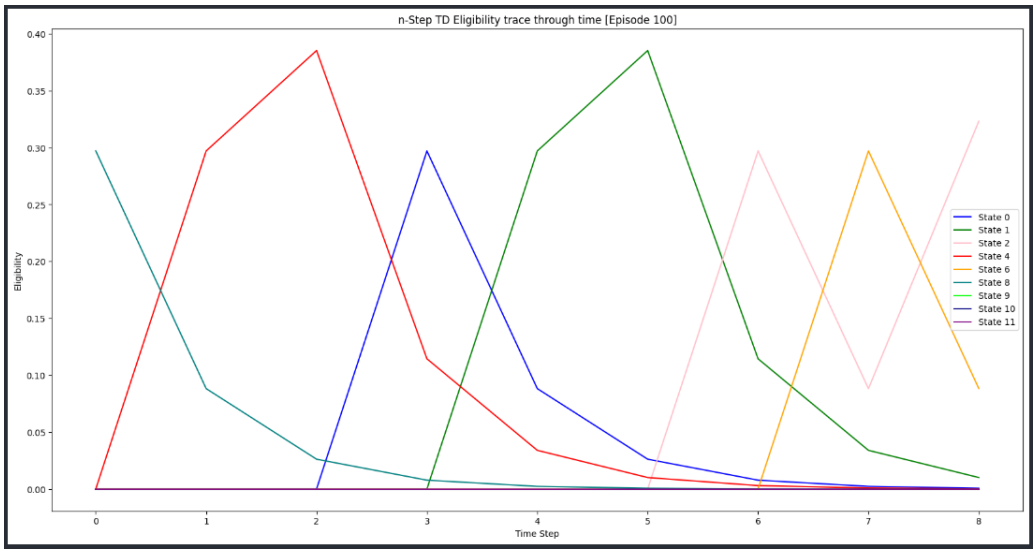
11. In the n-Step TD estimates plot, we observe a gradual convergence of the estimated state values towards their true values as the number of episodes increases. Initially, there is significant variance in the estimates, but with more episodes, the estimates become smoother and tend to approach the true values more closely. The choice of the step size parameter (α) can affect the rate of convergence and the stability of the estimates, with smaller values leading to slower but more stable convergence. Overall, the n-Step TD algorithm provides a flexible approach for estimating state values, balancing computational efficiency with estimation accuracy.



12. The TD(λ) algorithm gradually converges toward the true values of the state-value function as the number of episodes increases. Initially, there is variance in the estimates, especially in states with sparse rewards, but with more episodes, the estimates become smoother and approach the true values more closely. The choice of parameters, such as the step size (α) and eligibility trace (λ), influences the convergence rate and stability of the estimates. Smaller values of α and λ lead to slower but more stable convergence, while larger values may result in faster but more volatile estimates. Overall, TD(λ) offers a flexible approach to estimating state values, balancing exploration and exploitation while adapting to different learning environments.



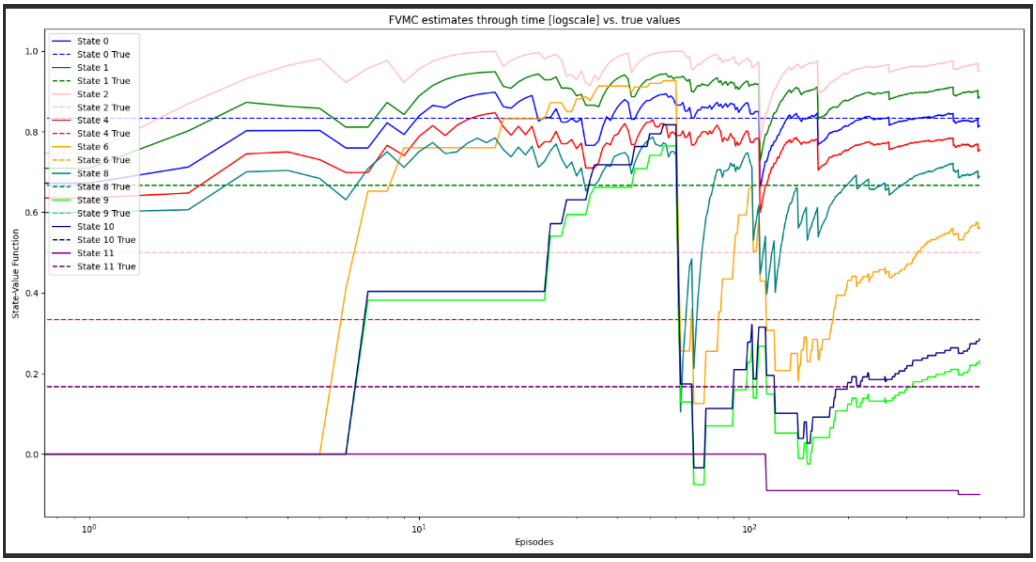
13. The TD(λ) algorithm was applied to estimate state values in the Random Maze Environment. The eligibility trace for each non-terminal state was tracked for episode number 100.The algorithm was tested for 500 episodes, and the eligibility trace records were analyzed specifically for episode 100.



Observing Figure, the eligibility trace for each non-terminal state demonstrates its evolution over time steps in episode 100. Different colors represent different non-terminal states. The eligibility trace captures the impact of each time step on the update of state values.

The eligibility trace analysis provides insights into the TD(λ) algorithm's behavior in episode 100. The plot visualizes how different non-terminal states' eligibility traces evolve over time steps during the episode.

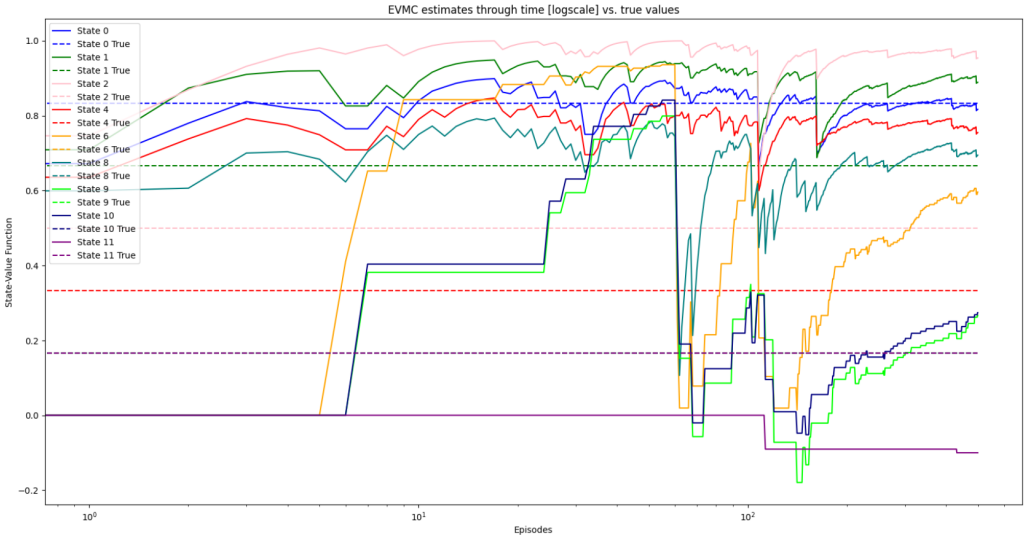
14. The Monte Carlo First-Visit MC (FVMC) algorithm was applied to estimate state values in the Random Maze Environment. The analysis focuses on the progression of estimates through different episodes, with the x-axis (episodes) presented in log-scale. The algorithm was tested for 500 episodes, utilizing varying settings for the step size parameter (α). The analysis is conducted to observe the behavior of estimates, especially in the initial stages.



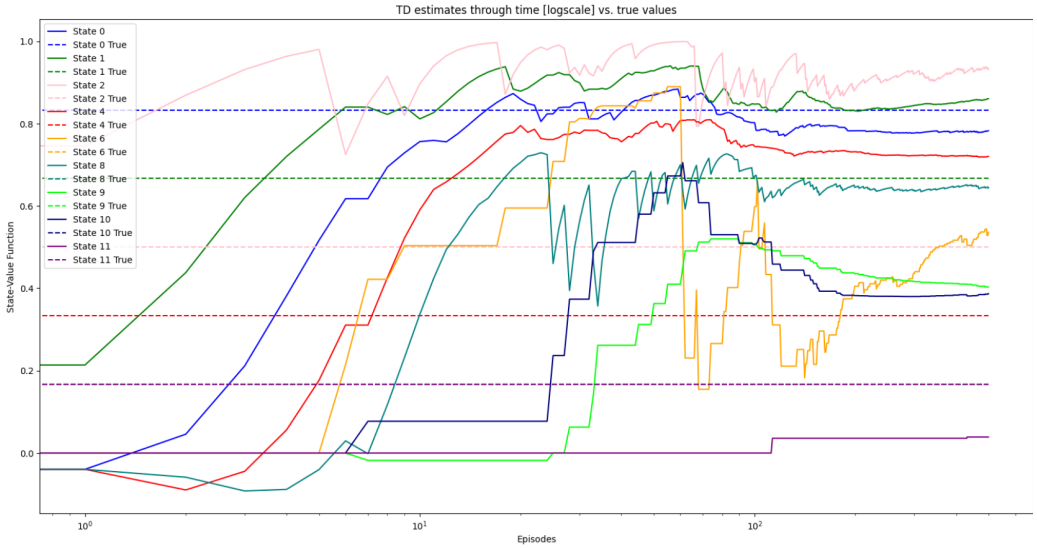
Observing Figure, the log-scale presentation of episodes provides a clearer view of the early-stage behavior of state estimates. The estimates demonstrate varying convergence patterns based on different non-terminal states. The logarithmic scale allows a focused analysis of the initial episodes, revealing nuances in the convergence of state values. Different α settings can influence the speed and stability of convergence, affecting the observed behavior.

The log-scale analysis enhances the understanding of the MC-FVMC algorithm's behavior, especially in the early episodes. The plot provides valuable insights into how state estimates evolve during the initial learning stages.

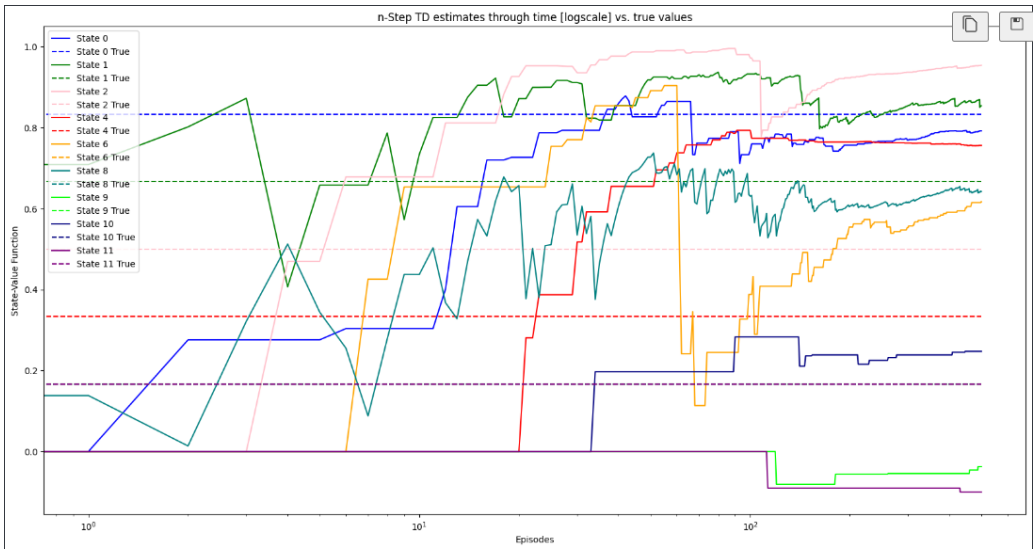
15. The MC-EVMC estimates of non-terminal states in the RME show rapid convergence, especially in the early episodes, with varying rates across states. EVMC tends to converge faster than FVMC due to its averaging effect. The logarithmic scale on the x-axis provides insights into the early convergence dynamics, revealing EVMC's effectiveness in approximating true values swiftly.



16. In the plotted TD estimates of each non-terminal state of RME, we observe varying convergence rates across states, particularly in the initial stages of training. The logarithmic scale on the x-axis allows us to focus on the early episodes, revealing rapid convergence trends. Overall, TD estimation demonstrates effectiveness in approximating true values, albeit with fluctuations, especially in the early learning phase.

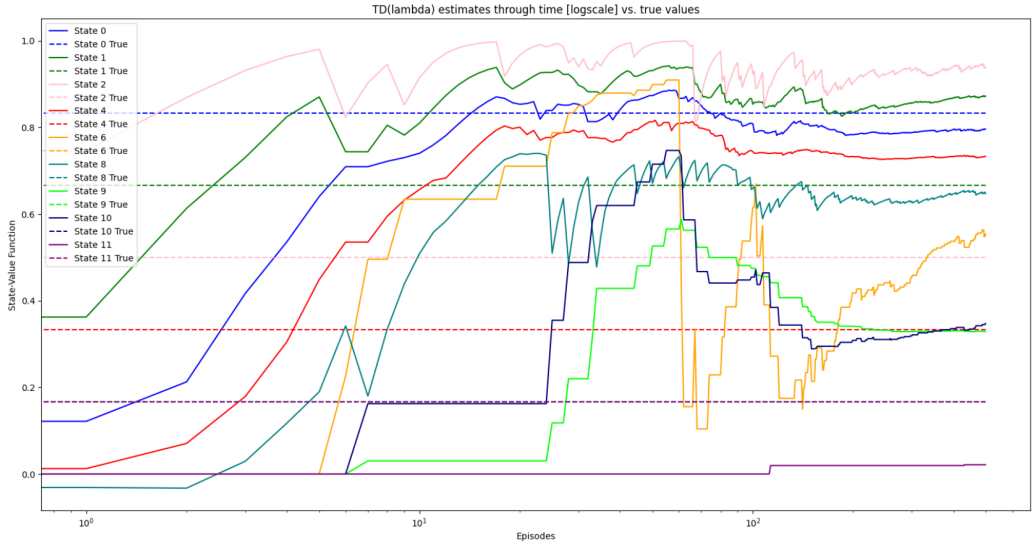


17. The $n - Step$ Temporal Difference (TD) algorithm was applied to estimate state values in the Random Maze Environment. The analysis focuses on the progression of estimates through different episodes, with the $x - axis$ (episodes) presented in $log - scale$. The algorithm was tested for 500 episodes, utilizing different settings for the step size parameter (α). The log-scale presentation of episodes aims to provide a clearer view of the initial stages of estimate behavior.



Observing Figure, the log-scale presentation allows a focused analysis of the early-stage behavior of state estimates. Each non-terminal state demonstrates unique convergence patterns. The choice of α and the n parameter can significantly impact the convergence speed and stability of the n-Step TD algorithm. Differences in state estimate behaviors may stem from these parameter choices. The log-scale analysis provides insights into the early-stage behavior of state estimates in the n-Step TD algorithm. Understanding the impact of parameter choices on convergence is crucial for effective reinforcement learning.

18. In the plotted TD(λ) estimates of each non-terminal state of RME with a logarithmic scale on the x-axis, we observe varying convergence behaviors across different states. The TD(λ) algorithm demonstrates the ability to approximate true values effectively, although convergence rates vary among states. This plot allows us to zoom in and observe the initial stages of learning, where rapid adjustments occur due to the exploration of state spaces. Overall, TD(λ) shows promise in accurately estimating state values, particularly in the early learning phase.



19. Based on the plots and comparisons of Monte Carlo (MC) methods, Temporal Difference (TD) methods, n-Step TD, and TD() approaches, the following observations can be made:

    Monte Carlo (MC) Methods

    - MC-FVMC and MC-EVMC exhibit slow initial convergence due to the need to complete entire episodes before updating state values.
    - MC-EVMC shows slightly faster convergence compared to MC-FVMC due to early value updates.

Temporal Difference (TD) Methods

- TD methods, including TD and TD(), demonstrate faster initial convergence compared to MC methods as they update state values after each step.
- TD() typically exhibits smoother convergence compared to standard TD, thanks to the eligibility trace mechanism.

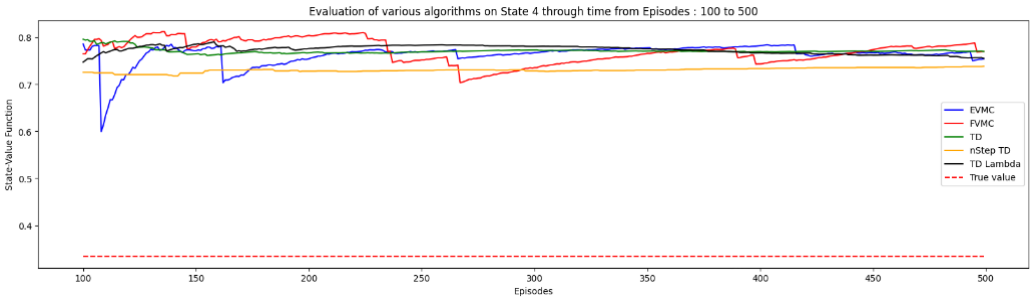
n-Step TD

- n-Step TD shows similar convergence characteristics to TD but may have different convergence rates depending on the chosen value of "n."

Comparison Plot

In the comparison plot for State 4, we observe:

- MC-FVMC and MC-EVMC exhibit slower convergence initially but eventually reach the true value.
- TD, n-Step TD, and TD() show faster initial convergence and closely approximate the true value function.



TD-based methods tend to converge faster compared to MC-based methods due to their online nature, while TD() provides a balance between bias and variance, leading to smoother convergence. However, the choice of algorithm may depend on the specific problem characteristics and computational considerations.

20. The Monte Carlo First-Visit MC (FVMC) algorithm was applied to estimate the Target Value (Gt) for a specific non-terminal state in the Random Maze Environment. The analysis focuses on the progression of Gt values through different episodes.

    The algorithm was tested for 500 episodes, with the step size parameter (α) following the specified settings. The Gt values for a non-terminal state (State 4) were recorded and compared with the true value.

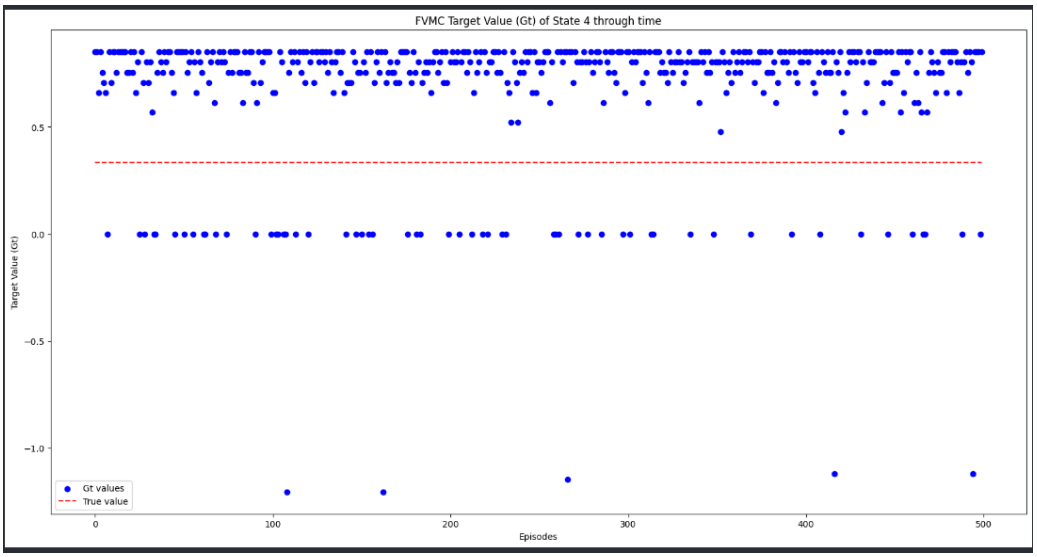
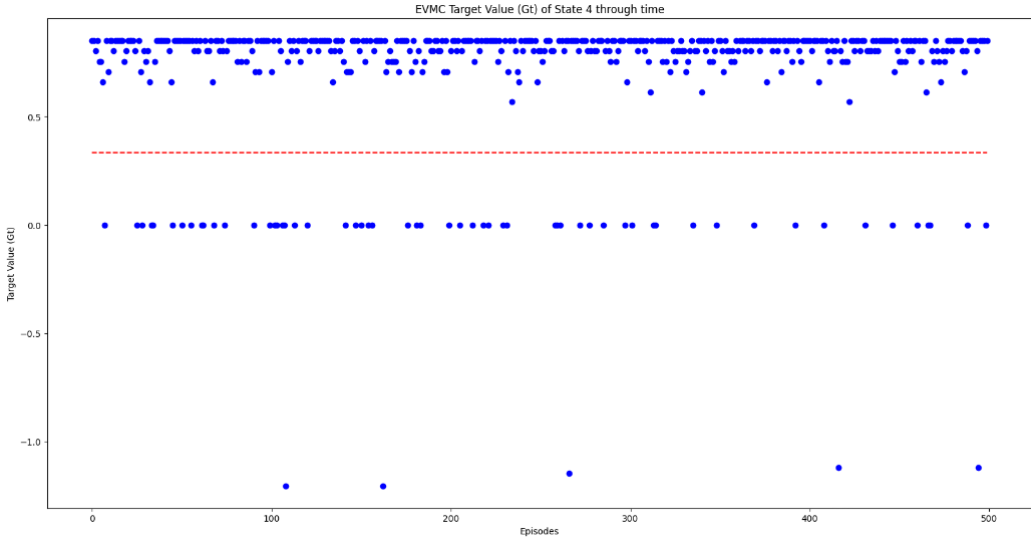


Figure illustrates the progression of Gt values for State 4 through different episodes. The red dashed line represents the true value of the state.

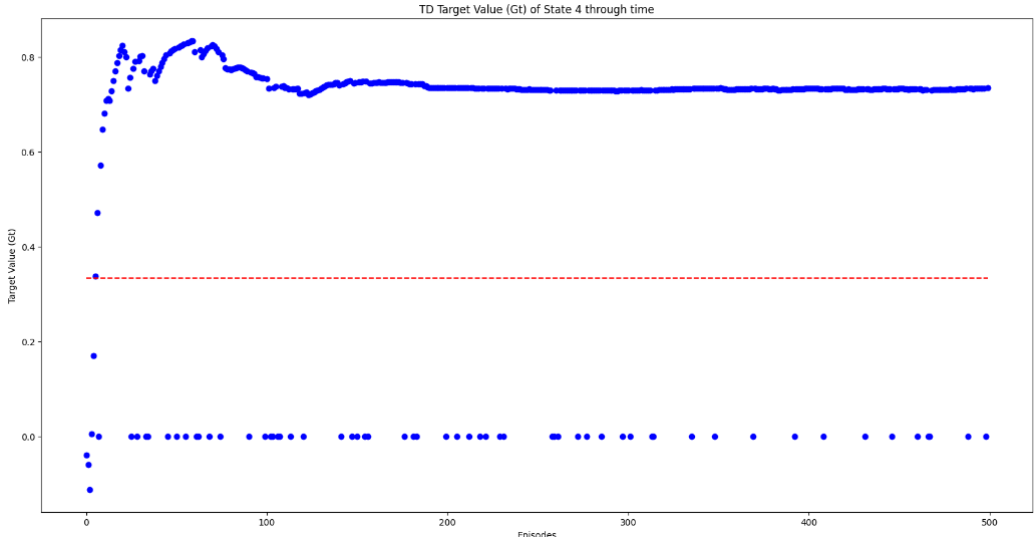
The observed behavior in the plot indicates the convergence of Gt values towards the true value as the number of episodes increases. The variance in Gt values diminishes over time, reflecting the learning process of the algorithm.

The analysis of MC-FVMC Target Value (Gt) provides insights into the convergence of estimates for a non-terminal state. The observed behavior aligns with the expected learning process, showcasing the algorithm's ability to approximate true values over episodes.

21. The plot shows the MC-EVMC Target values (Gt) for State 4 of RME over multiple episodes. Initially, there is high variance, but the values gradually converge towards the true value of the state. This convergence reflects the learning process of the agent, influenced by factors such as exploration, randomness, and the number of episodes sampled. Overall, the plot demonstrates the method's ability to estimate state values through iterative learning, eventually approximating the true value of the state.



22. The plot depicts the TD Target values (Gt estimates) for State 4 of RME across different episodes. Similar to the MC-EVMC case, there is initially high variance, but the values gradually converge towards the true value of the state. This convergence reflects the learning process of the TD algorithm, where the estimates are updated iteratively based on the observed rewards and predictions. As more episodes are sampled, the estimates tend to approximate the true value of the state more closely.
The plot demonstrates the TD algorithm's ability to learn and estimate state values over time.



23. **Comparison of MC-FVMC, MC-EVMC, and TD Targets**

The plots comparing MC-FVMC, MC-EVMC, and TD targets provide the following observations:

MC-FVMC Target:
Initially, there is high variance in target values, indicating uncertainty.
Over episodes, the target values gradually converge towards the true value of the state.
The variance remains relatively high throughout the episodes compared to other methods.

MC-EVMC Target:
The behavior is similar to MC-FVMC initially, with high variance in target values.
However, there is quicker convergence towards the true value compared to MC-FVMC.
The variance in the estimates is lower than MC-FVMC, suggesting more stable estimates.

TD Target:
The convergence pattern differs from MC methods.
Convergence starts earlier, and the estimates stabilize faster.
Generally, there is lower variance in the estimates compared to MC methods.

TD targets show faster and smoother convergence with lower variance compared to MC-FVMC and MC-EVMC targets. MC-EVMC targets exhibit faster convergence than MC-FVMC but with slightly higher initial variance. This suggests that TD methods may offer more efficient and stable updates for state value estimation.