## *Multi Armed Bandit Problem*

**Generating the Testbed:** Following the procedure given in the textbook and using these values we generate the 10 bandit testbed:

- Runs: 2000
- Time Steps: 1000
- K = 10
- q* = N(0,1) where N = normal distribution

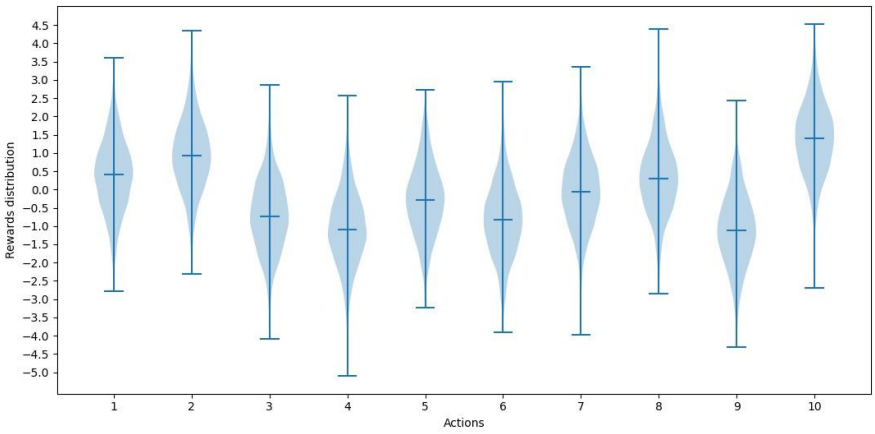



Fig1. 10 armed testbed

**Epsilon Greedy Action Selection:** Used five different epsilon values to compare performance:

- Ɛ = [0,0.01,0.1,0.5,0.9]

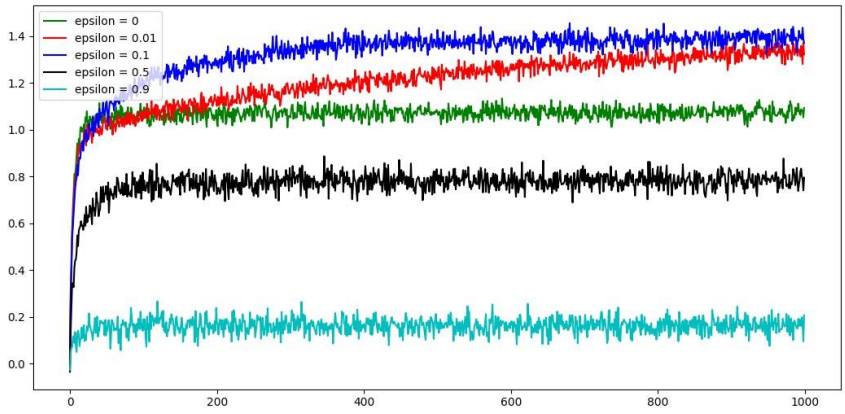



Fig2. Average Reward v/s Time Step for various epsilon values

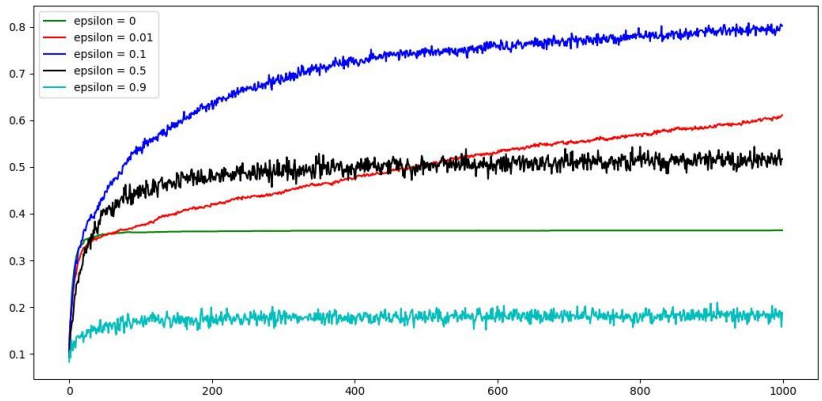



Fig3. %Optimal Action v/s Time Step for various epsilon values

In both the graphs, Ɛ = 0.1 outperforms the other values. Ɛ = 0.01 learns at a slower rate as it's less exploratory than others, but with time it catches up to Ɛ = 0.1 in average reward. Higher values like 0.5 and 0.9 are clearly less than ideal to use as parameters, in both the long and short run. This is because even after the optimal value is found, it keeps exploring which decreases average reward earned. Such high values actually end up performing worse than a simple greedy algorithm and thus are of no use to us. The graph for simple greedy algorithm shows a quick increase and then flattens, this is attributed to the fact that once it finds a optimal value, it keeps chosing the same action without updating the others.

**Optimistic Initial Value:** Comparison shown between Optimistic Greedy using initial value = 5 & Ɛ = 0 and realistic epsilon greedy using initial value = 0 & Ɛ = 0.1. Alpha in both cases was a constant = 0.1

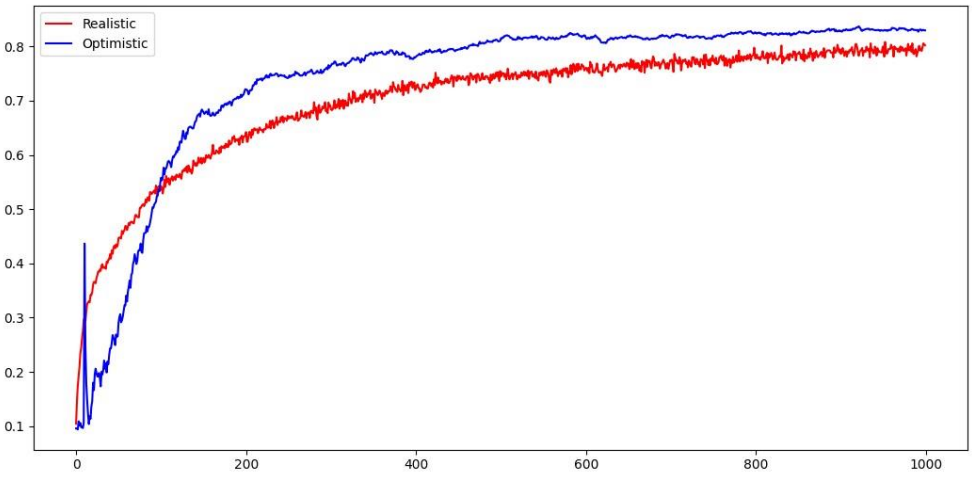



Fig4. %Optimal Action v/s Time Step for Optimistic Initial value v/s Realistic epsilon

The Optimistic Initial Value strategy promotes exploration without altering the epsilon value. In the initial stages, its performance lags as it prioritizes exploration. However, over the long term, it excels because exploration diminishes over time, and it identifies a high-reward value that yields substantial returns during its exploration phase. I also made efforts to fine-tune the parameters:

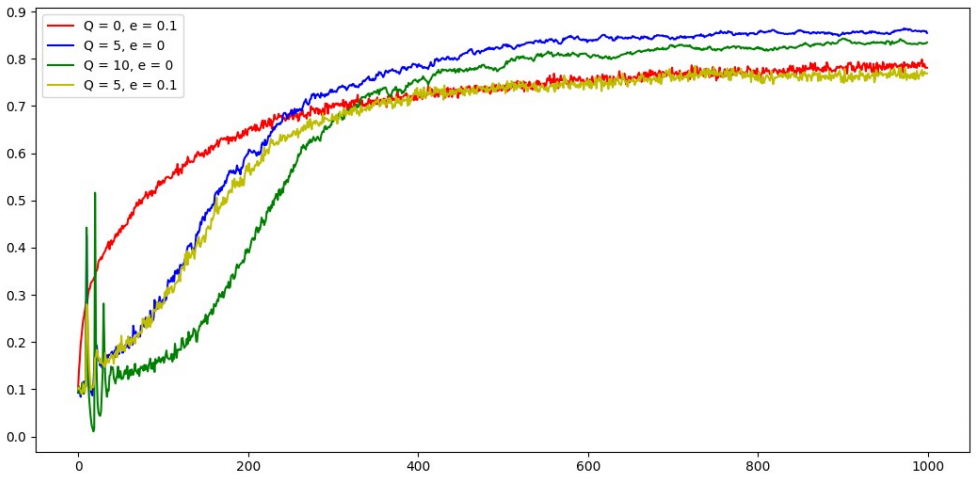



Fig5. %Optimal Action v/s Time Step for different values

- In the above graph for the first ten steps (10 arms in this case) the chance that we take an optimal action is lower since we are doing exploration initially.(even though epsilon is 0, it becomes exploration because of the optimistic values).
- At the 11th step since all arms have been visited at least once, we now choose the arm with maximum reward
- If we increase the value of Q, then the optimistic graph will move upwards
- models with Ɛ = 0 and optimistic initial value outperform those with added exploratory power when Ɛ > 0 in the long run.

**Upper Confidence Bound (UCB) Action Selection**: Using c = 2 and & Ɛ = 0.1 and following the procedure done in textbook, this is the graph that was created:

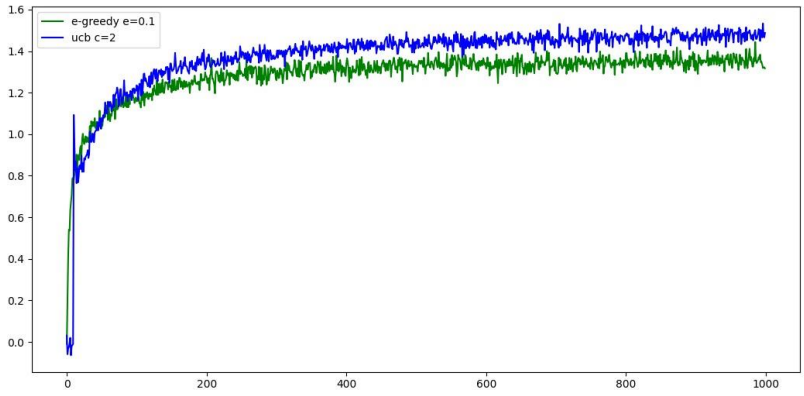


Fig6. Average Reward v/s Time Step for UCB vs Simple epsilon greedy method

Because UCB learns to balance trying out novel actions and sticking with tried-and-true actions more skillfully by taking uncertainty in action values into account, it performs better than epsilon-greedy. Epsilon-greedy is erratic and less able to adjust to the environment as it changes.

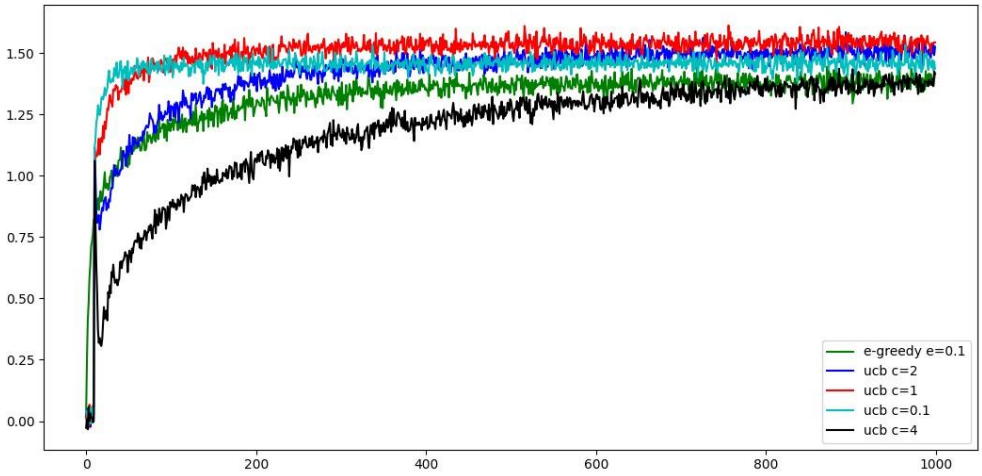



Fig7. Average Reward v/s Time Step for UCB vs Simple epsilon greedy method – tuning parameters

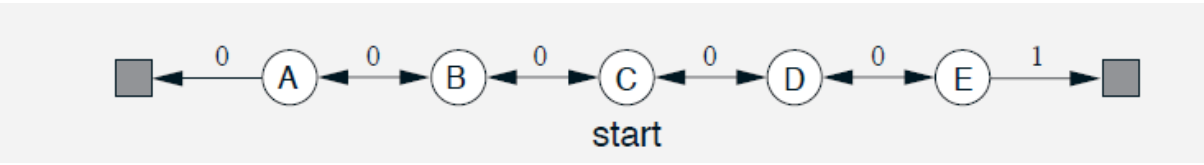
As can be seen in both graphs, c = 1 performs better than the rest. Greedy performs better than c = 3 but worse than c = 1 and c = 2. The optimal action is discovered significantly more quickly by C = 0.1, but it immediately reaches a plateau and, after 400 iterations, has a lower optimal action percentage than greedy, while still having greater average returns. All of the UCB techniques outperform greedy over the long run in terms of average returns.

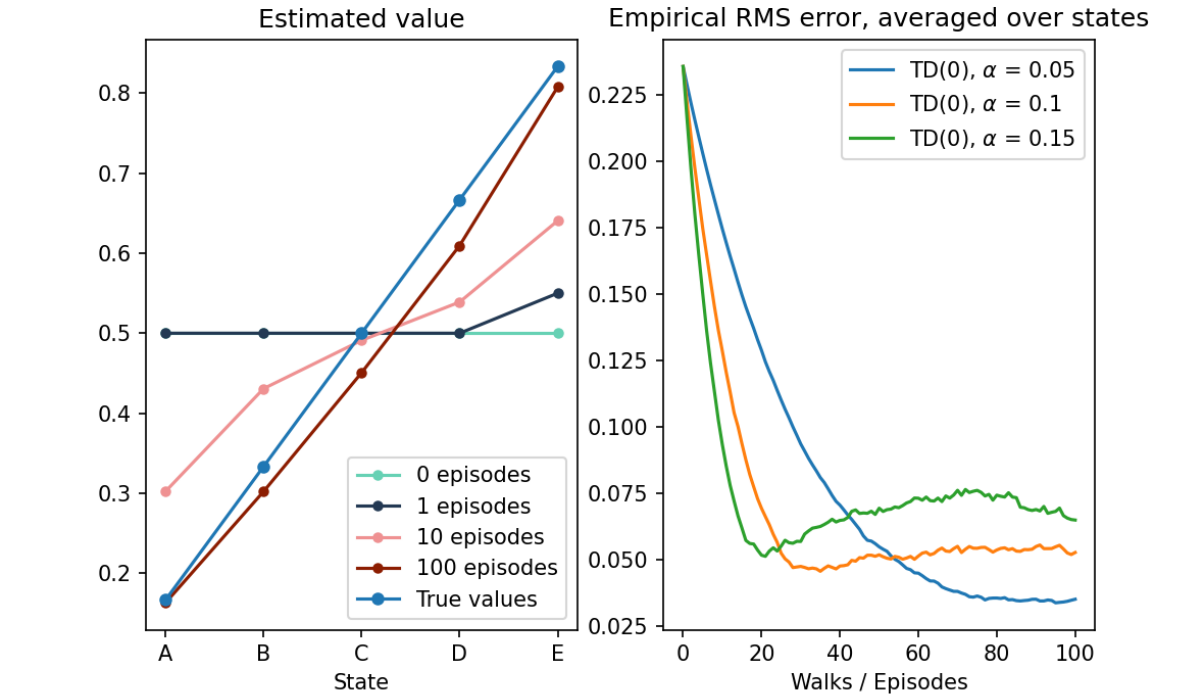
**Question 2:**

**In this case, we apply the following Markov reward process and empirically assess the prediction capabilities of TD(0) and constant alpha MC:**

A Markov Reward Process (MRP) is essentially a Markov Decision Process (MDP) without any actions involved. MRPs are commonly used when we are primarily concerned with prediction and do not need to differentiate between the environment's dynamics and the agent's actions. In this specific MRP, all episodes begin in the central state, denoted as "C," and then progress either to the left or right by one state in each step, with an equal probability for each direction. Episodes end when they reach either the far-left state or the far-right state. If an episode terminates on the right side, a reward of +1 is received, while all other state-action pairs yield rewards of zero. For instance, a typical episode might follow this sequence: C, 0, B, 0, C, 0, D, 0, E, 1. Since this task is not subject to discounting, the

actual value of each state is equivalent to the probability of terminating on the right side when starting from that particular state. Consequently, the true value of the central state, v(C), equals 0.5.
True values of A,B,C,D,E are 1/6,2/6,3/6,4/6,5/6




The values acquired after varying numbers of episodes on a single run of TD(0) are displayed in the left graph above. The estimations after 100 episodes are as near to the actual values as they have ever been. The value of alpha here is 0.1
The learning curves for the two approaches for various values of alpha are displayed on the right graph. The root mean square (RMS) error between the value function learned and the true value function, averaged over the five states and averaged over 100 runs, serves as the performance indicator. The intermediate value V (s) = 0.5, for every s, was used to initialise the approximation value function in each case.
In TD(0) learning, the root-mean-squared errors (RMSE) do not always converge to zero. Several variables, including as the selection of step size (alpha), the characteristics of the environment, and the beginning conditions, affect whether they converge to zero.
In the above case it does move towards zero but doesn't converge necessarily.

It depends on :
**Alpha :** The updates may be too aggressive and cause divergence rather than convergence if alpha is too large. If alpha is set too low, convergence may take a while since the updates are too cautious. Convergence depends on the selection of the alpha.

**Initial Values**: Convergence may be impacted by the first estimates of state values. It could take longer for the estimates to converge if they are distant from the true values.

**When alpha is set to 1/n**

When we use the sample average update rule with alpha = 1/n:

- "n" represents the number of times a state has been visited (sample count).
- The update rule becomes: new_estimate = old_estimate + (1/n) * (target - old_estimate).
- Here, "target" is the actual observed return (cumulative reward) from that state.
- The effect of this rule is that it averages the returns observed from each visit to a state.
- setting alpha to 1/n in TD(0) learning effectively results in a sample average update rule. It averages observed returns over visits to states, leading to stable and slowly changing value estimates that converge to the true values as more experience is gained.