Report for Exercise 6 in WSI

Adam Szostek, Index Number 331443

February 11, 2025

1 Implemented Algorithm

1.1 Q-learning

The implemented algorithm is Q-learning, which is one of the most popular algorithms in reinforcement learning. The main goal of the algorithm is to construct a Q-table that assigns values to each possible action available to the agent in a given state.

Algorithm 1: Q-learning Algorithm

Data: t_{max}, γ, β, e_{max}
Result: Q-table
Function `Q-learning`(t_{max}, γ, β, e_{max})**:**
  $Q_0 \leftarrow$ initialize;
  $e \leftarrow 0$;
  while $e < e_{max}$ **do**
    $t \leftarrow 0$;
    $x_t \leftarrow$ initialize;
    while $t < t_{max}$ ***and*** $x_t \notin$ *absorbing states* **do**
      $a_t \leftarrow$ choose action(x_t, Q_t);
      $r_t, x_{t+1} \leftarrow$ execute action a_t;
      $\Delta \leftarrow r_t + \gamma \max_a Q_t(x_{t+1}, a) - Q_t(x_t, a_t)$;
      $Q_{t+1} \leftarrow Q_t + \beta\Delta$;
      $t \leftarrow t + 1$;
    $e \leftarrow e + 1$;

1.2 Action Selection Strategies

Three action selection strategies have been implemented:

- **Epsilon-Greedy Strategy**: The `Epsilon-Greedy` strategy involves choosing a random action with probability ϵ, and otherwise (with probability $1 - \epsilon$) selecting the action with the highest value in the Q-table for the given state. This is a popular method balancing exploration (random actions) and exploitation (selecting the best-known action).

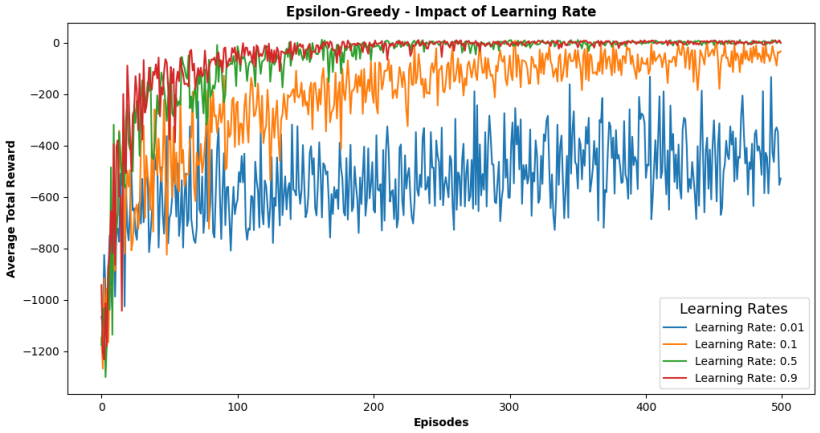
- **Boltzmann Strategy**: The `Boltzmann` strategy (also known as softmax) selects an action based on a probability distribution dependent on the Q-values of the actions and a temperature parameter. The higher the Q-value for an action, the greater the probability of its selection. The temperature (T) controls the degree of exploration: the higher the temperature, the more random the choices become.

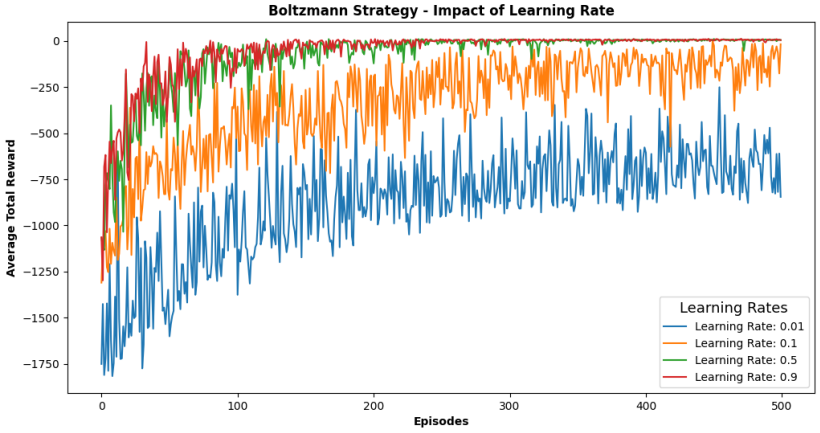
- **Count-Based Strategy**: The `Count-Based` strategy modifies the Q-values based on the number of visits to a given action in a specific state. It works by rewarding less frequently chosen actions by assigning them additional points in the Q-table, which motivates the agent to explore them.

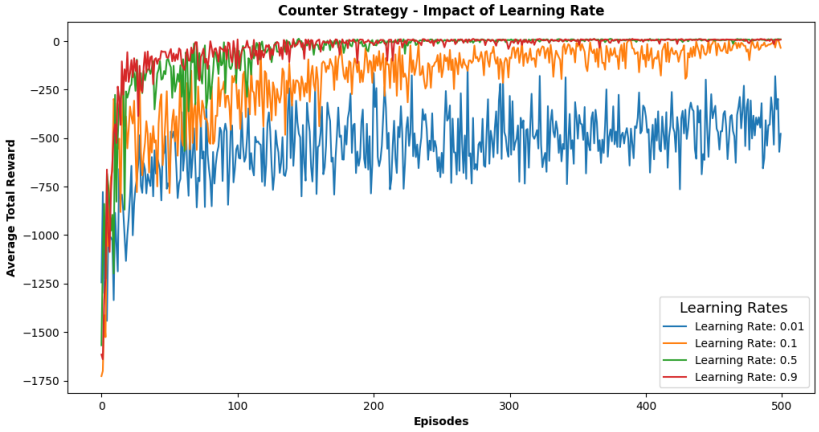
2 Planned Experiments

Q-learning was tested in the Taxi-v3 environment. This is an environment from the OpenAI Gym library that simulates a taxi transportation problem in a city. The agent's (taxi's) goal is to transport passengers from one point to another, learning the optimal strategy based on rewards. The agent receives positive rewards for delivering passengers to the designated destination and is penalized for unsuccessful attempts and inefficient actions, such as moving aimlessly. Using this environment, the effectiveness of the three implemented action selection strategies and the impact of the learning rate on the algorithm were examined. For this purpose, plots of cumulative rewards per episode for the agent were created depending on the number of training episodes. Each curve is an average of five training trials of the agent.

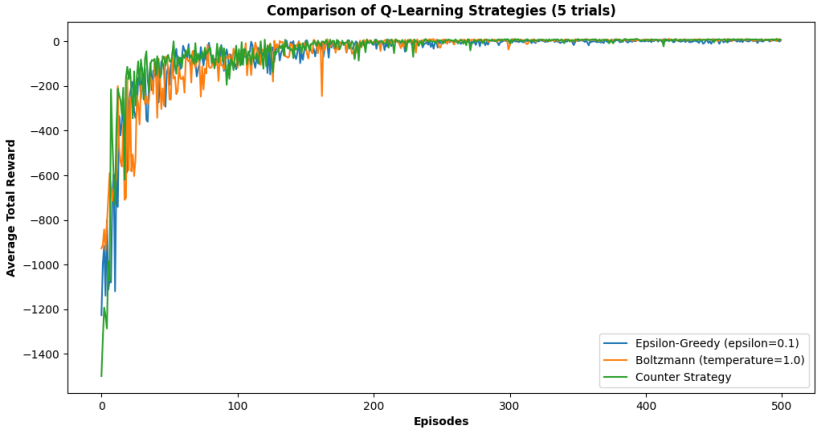
3 Obtained Results







In the plots examining the influence of the learning rate on the Q-learning algorithm, it is evident that for each strategy, the best results are achieved with 0.9 and 0.5, while the remaining values, although initially also seem to approach the optimum, provide much less stable and overall worse results. In further studies, a Learning Rate of 0.9 was used.



In the above plot, it is visible that the tested strategies achieve almost identical results. The only noticeable difference is the initial convergence speed of the curves to the optimum—the count-based and epsilon-greedy strategies reach it slightly faster than the Boltzmann strategy.

4 Conclusions from the Results

Based on the conducted numerical experiments, it can be concluded that the learning rate has a significant impact on the performance of the Q-learning algorithm. Choosing too small a learning rate results in the agent achieving much poorer average rewards per episode, despite using the same strategy. Additionally, it is observed that for values closer to 0.9, the curves are much more stable, i.e., we do not observe their rapid oscillation, which occurs with smaller values. These oscillations may result from the agent's limited ability to adapt based on new information, leading to the selection of suboptimal actions that often result in negative rewards.
When comparing strategies, it can be stated that the result achieved after a set number of episodes is practically identical. As noted in the results description, initially, the

Boltzmann strategy converges to the optimum more slowly. The reason for such behavior may be that the other two strategies have much simpler exploration mechanisms, allowing the agent to identify good actions for given states more quickly. In the case of the epsilon-greedy approach, the optimal solution may be found faster, but due to its more deterministic action selection, high-potential actions may be overlooked. In a testing environment with a larger number of states and actions, the count-based strategy might prove to be the most effective due to its mechanism of rewarding less frequently explored actions. Conversely, in environments where exploration is crucial and rewards are harder to predict, the Boltzmann approach may be optimal. It is worth mentioning that for the Boltzmann and epsilon-greedy strategies, parameter selection (temperature t for Boltzmann and epsilon for epsilon-greedy) is crucial and has a significant impact on their performance. For example, with too high epsilon values, the approach will choose random actions too often, preventing the agent from achieving optimal rewards.
In summary, in long-term experiments, all strategies converge to a similar level of average rewards, indicating their effectiveness. However, there are scenarios where one strategy may be more optimal than the others. Meanwhile, the hyperparameter learning rate has a crucial impact on the algorithm regardless of the chosen strategy.