David Shimshoni, Daniel Richardson, and Rohan Vora

Learning to Play Tic-Tac-Toe

Abstract

In this project, we investigate how the policy parameter of a Q-learning agent can be optimized in order to maximize the agent's performance as it learns how to play the game tic-tac-toe. Specifically, we contrast a constant value of epsilon, the policy parameter, to a value that decays with training time. In this case, epsilon acts as the probability on any given iteration that the agent will "explore" by selecting a random action, rather than "exploit" by employing the policy that it has determined will yield the maximum reward. In order to evaluate the success of these two methods of setting epsilon, our experiment pits the agent against a completely random agent as it learns over 100,000 episodes. Our experiment reveals that a decaying value of epsilon yields stronger results (i.e. a higher win-rate) than a constant value of epsilon, based on the values and formulas chosen. Specifically, the win-rate is maximized when epsilon is defined as the inverse of a logarithmic function with respect to time.

- David: Wrote up the code for the project to train the agent, test values of epsilon, and display the results of training. Helped edit the slides and report.
- Daniel: Helped in writing and editing the slides and the report, as well as organizing the code repository.
- Rohan: Helped in writing and editing the slides and the report, as well as recording the presentation.

Introduction

Tic-tac-toe is a simple game in which two players take turns placing a symbol (either an "X" or "O", based on player) on a 3x3 grid, with the goal of winning by placing three symbols in a row. The game is solved to the extent that an experienced player can algorithmically force a tie, or a win if their opponent misplays, with absolute certainty. However, as a Q-learning agent always works towards maximizing the expected value of the reward function, it remains unclear how it would learn tic-tac-toe, and whether its behavior would ever mirror that of a human (i.e. optimal). In order to form a policy that results in a high win-rate, the agent must be able to balance its willingness to evolve and improve its policy by learning the rewards of random actions with its ability to consistently win games by employing its policy. In order to solve this, we ask, how can the policy parameter (epsilon) of Q-learning be optimized to perform as well as possible in tic-tac-toe?

Previous research on the application of reinforcement learning to tactical games is extensive. A paper by H. Wang, M. Emmerich, and A. Plaat compares the effectiveness of a Q-learning agent with that of a Monte Carlo tree search algorithm akin to the one used in the

AlphaGo AI[1]. Their findings show that Q-learning tends to converge, albeit slower than an MTCS method. However, the fact that Q-learning is eventually effective at all at developing a strong policy leads us to hypothesize that a decaying rate for epsilon over the number of games played will lead a Q-learning agent to perform better than with a consistent epsilon.

Methods

The first necessary part of the experiment was a scalable and versatile tic-tac-toe environment. We chose gym-tictactoe, an open-source environment built for artificial intelligence use by Kim Jeong Ju, Dustin Michels, and Johann Miller, as it was simple to use and contained all the tools necessary for a Q-learning agent to use it. Each Q-learning agent was given 100,000 episodes (games) to train and compete, which allowed the agents to sufficiently train themselves without consuming too much time or processing power. The agents were pitted against a completely random opponent each episode, allowing them to learn from a variety of states during exploration. A non-random opponent would not achieve this; for example, one such opponent may consistently provide a certain path to victory that is not present with other opponents, creating a heavy bias in the policy.

The next step was to choose how the policy parameter (epsilon) was set in each case. Epsilon is the probability on a given turn of the game that the agent opts towards "exploration" (taking random steps) rather than "exploitation" (taking steps to maximize the expected value of the reward). Epsilon is a value between 0 and 1, and in general, an effective algorithm will have a value somewhat close to 0. When fixing constant epsilon, we settled on values of 0.05, 0.1, and 0.2, as they mostly covered the range of sensible values for constant epsilon. In terms of formulas for decaying epsilon, we chose the reciprocal of ep, the reciprocal of the square root of ep, and the reciprocal of log(ep-1+e), where ep is the number of the present episode. These formulas provide a value of epsilon between 0 and 1, and they asymptotically limit towards 0 as ep increases, meeting the requirements of a decaying epsilon.

The metric of success is the win-rate of the final 1,000 games, as each agent should be sufficiently trained in the first 99,000 games to represent its respective policy parameter.

Results

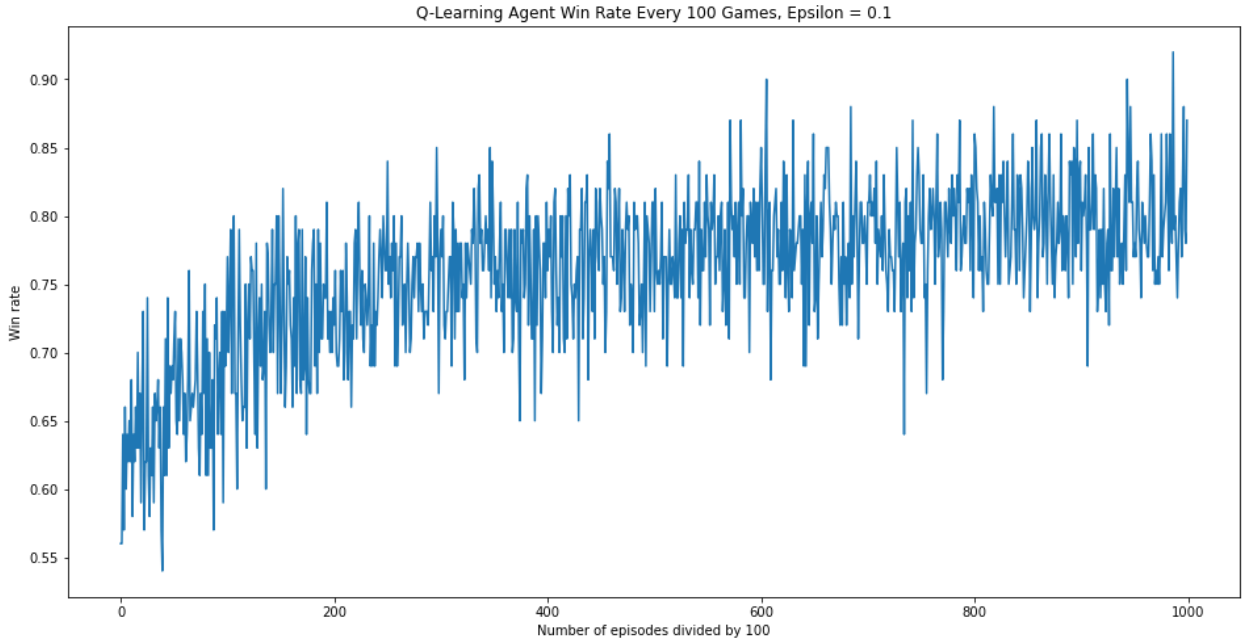
The win-rate of the final 1,000 games was highest for the reciprocal logarithmic function at 0.827, while no other agent with decaying epsilon outperformed a fixed epsilon of 0.1, with a win-rate of 0.807. The full list of win-rates is as follows:

```
The win rate in the last thousand games for the epsilon setting 0.05 is 0.779
The win rate in the last thousand games for the epsilon setting 0.1 is 0.807
The win rate in the last thousand games for the epsilon setting 0.2 is 0.767
The win rate in the last thousand games for the epsilon setting reciprocal is 0.7
The win rate in the last thousand games for the epsilon setting reciprocal_square_root is 0.777
The win rate in the last thousand games for the epsilon setting reciprocal_nat_log is 0.827
```

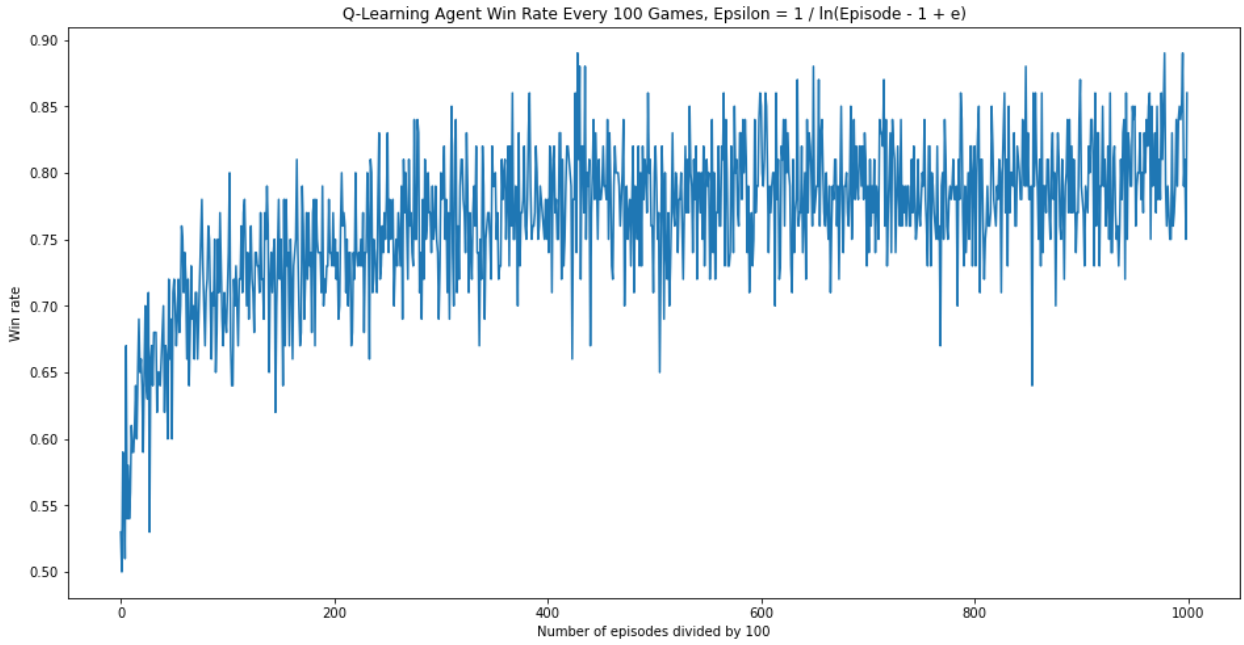
[1] Wang, Hui, et al. "Monte Carlo Q-Learning for General Game Playing." *ArXiv.org*, Cornell University, 21 May 2018, arxiv.org/abs/1802.05944.

Furthermore, the win-rates for each hundred games in each agents' training period was recorded.

For epsilon = 0.1, the win-rate progressed as shown:



The agent with epsilon based on the reciprocal logarithmic function provided the following figure:



From this, it can be seen that the reciprocal logarithmic function provided heavy policy improvement early on before stabilising after about 50,000 episodes. On the other hand, the rate of improvement seems bimodal in the case of epsilon = 0.1, wherein the policy improves at a

strong pace for the first 25,000 episodes before regressing to a slow, stable rate of improvement for the remaining episodes. However, we can also see that a fixed epsilon provides something of a soft upper limit on the win-rate of the agent, as it always has a significant propensity to take random, potentially suboptimal moves. Epsilon = 0.05 learns extremely slowly compared to other agents with fixed epsilon, while epsilon = 0.2 suffers from the upper limit issue more so than epsilon = 0.1, meaning epsilon = 0.1 is, for the purposes of this experiment, the standard by which to evaluate agents with decaying epsilon. As previously stated, the reciprocal logarithmic function clearly outperformed both other agents with decaying epsilon, as well as the agent with epsilon fixed to 0.1, leading us to conclude that agents with decaying epsilon can categorically outperform agents with fixed epsilon.

Discussion

In observing the data, our hypothesis is shown to be correct by the winning performance of the agent with the reciprocal logarithmic decaying value of epsilon. This leads us to conclude that for a tic-tac-toe environment, the agent benefits greatly from acting greedily and exploiting rather than exploring. In training agents to effectively play tic-tac-toe, this experiment demonstrated that acting greedily yields greater reward, and that Q-learning is a good method to do so.

These findings support Hui et al.'s findings that Q-learning is very effective for use in tactical games, and the convergent nature of the agent trained with a reciprocal logarithmic value of epsilon mirrors that experiment's results as well. However, their use of MTCS and higher level methods would probably outperform even our best agent, and leaves lots of room for further investigation into this field.

In this vein, there are factors to be explored in Q-learning beyond the epsilon parameter. For example, the learning rate and discount factor were discussed in our initial framing of the problem, but for the sake of this experiment we focused solely on adjusting exploitation with epsilon. An interesting further investigation into these findings could focus on adjusting alpha or gamma for the agent.

There is also room for improvement in the nature of the agent's opponent. A random opponent was the most straightforward to implement and provides a useful metric of the agent's success, though true tic-tac-toe opponents would play far better and provide a bigger challenge for the agent. In fact, tic-tac-toe players that act perfectly can only win or tie; as our agent still faces some losses to the random opponent, this indicates room for further optimization of the Q-learning parameters or perhaps investigation into more powerful reinforcement learning algorithms, such as those explored by Hui et al.