10/08/2024

Project 4: TIC-TAC-TOE with RL

Dr. Barak Or

Introduction

In this project, you will delve into the world of reinforcement learning by developing an advanced version of the classic Tic Tac Toe game. You will be responsible for coding the game's logic and employing a variety of reinforcement learning strategies, including Monte Carlo method, Q-learning, Deep Q-Network (DQN), and Double Deep Q-Network (DDQN). Your goal will be to train models that master the game. You will construct the game environment, apply cutting-edge reinforcement learning techniques, and assess how well your models perform.

This hands-on experience is designed to equip you with a solid understanding of reinforcement learning fundamentals and its real-world applications in creating intelligent systems. By the end of the project, you will have practical skills in training AI agents to tackle complex challenges.

Description

1. Monte Carlo Method: Tuning Hyperparameters and Analyzing Impact on Game Performance

In this exercise, you'll modify various hyperparameters of the TicTacToe MonteCarlo class and observe how these changes affect the performance of the trained agent. This will help you understand the underlying principles of Monte Carlo methods in reinforcement learning. Below are the tasks:

a. Set Initial Parameters: Begin with the following default settings:
size = 3 (3x3 board)
learning_rate = 0.001
discount_factor = 0.99
epsilon = 0.1 (exploration rate)
episodes = 100,000 (number of training episodes)

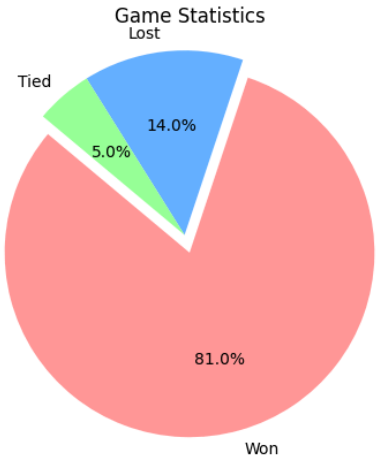
b. Modify and Test Each Parameter:

- Learning Rate: Increase and decrease the learning rate. Observe how faster or slower learning rates affect the convergence speed and the stability of the learning process.
- Discount Factor: Experiment with higher and lower values close to 1 and 0, respectively. Assess how this impacts the agent’s prioritization of immediate versus future rewards.

- Epsilon: Vary the epsilon value to alter the balance between exploration and exploitation. Note changes in the agent's ability to discover new strategies versus refining known strategies.
- Episodes: Adjust the number of episodes. Evaluate how the amount of training impacts the agent's overall performance and ability to generalize.

c. Plot Game Statistics:

Choose 2 parameter adjustments, plot the win rate, loss rate, and tied rate before and after adjustment. Use these plots to visually represent how the performance metrics evolve with different settings. Example of a plot:



2. Implementing and Tuning Q-Learning for Tic Tac Toe

In this exercise, you will implement the Q-learning update rule in a function and experiment with various hyperparameters to see how they affect the performance of the Q-learning agent. Your task will also involve completing the missing parts of the Q-table update function based on the Q-learning formula:

a. Complete the “update_q_table” function by filling in the missing parts. Specifically, you need to compute the maximum Q-value for the next state, which is a key component in the update rule. Here's the partial function. Implement the following equation:

$$Q(s,a) = Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$

b. Changing the learning rate to 0.1. How do changes in the learning rate affect the convergence and stability of the agent's learning?

c. How does adjusting the discount factor to 0.9 and 0.5 influence the agent's prioritization of immediate versus future rewards?

d. What is the impact of modifying epsilon to “1” on the trade-off between exploration and exploitation?

e. How does changing the number of training episodes to 10,000,000 influence the learning outcomes?

f. Change the tie_reward to negative value. What is the impact of this modification? Explain.

3. Boosting Q-Learning with Deep Learning: Investigating DQN and DDQN

To enhance your grasp and engagement with DQN (Deep Q-Network) and DDQN (Double Deep Q-Network) in the context of a Tic Tac Toe game implemented with deep learning techniques, answer the following questions:

a. Review the DQNAgent class and explain: where and how the neural network should be integrated in Q-learning? What are the advantages/disadvantages?

b. Propose a neural network architecture suitable for the DQN agent managing a Tic Tac Toe game. Detail the architecture by specifying the number and type of layers (e.g., dense, dropout), activation functions, and any pertinent parameters. Subsequently, integrate this architecture into the existing DQNAgent class structure. Complete the _build_model(self) function and describe the result.

c. **Bonus**: Switch to DDQN and explain if it resulted in a noticeable improvement over DQN regarding overestimation minimization and training stability? Analyze win rates, loss rates, and tie rates under both algorithms.