Game Playing AI

Group 9 Members

Satyam Patel - 8975977
Yakub Yekini - 8939553
Harsh Rajkumar Yadav - 8977229

Overview

In this project, we aim to develop a Reinforcement Learning (RL) agent that can play and excel at classic Atari games. The objective is to create an agent that learns to play these games from scratch, achieving high performance and demonstrating sophisticated strategies. This report outlines our plan, detailing the problem we aim to solve, the algorithms and tools we will use, and the comparisons we intend to conduct to evaluate our agent.

Problem Description

Type of Problem

The primary problem we are addressing is the development of an intelligent agent capable of playing Atari games autonomously. These games, known for their simplicity and complexity, provide an ideal testbed for RL algorithms. Our goal is to have the agent learn optimal strategies through interactions with the game environment, maximizing its score or achieving specific in-game objectives.

Objectives

- **Develop an RL Agent for Atari Games**: Create an agent that can play a variety of Atari games, learning to navigate their unique challenges.
- **Achieve High Performance**: Aim for the agent to perform at or above human level, maximizing game scores and demonstrating efficiency.
- **Implement Learning Algorithms**: Use and fine-tune advanced RL algorithms to enable the agent to improve over time through experience.

Algorithm and Tools ↗

Algorithms

For this project, we will focus on two primary algorithms:

Deep Q-Networks (DQN)

- **Description**: DQN is a model-free RL algorithm that combines Q-Learning with deep neural networks. It handles high-dimensional sensory inputs, such as game frames, by learning to approximate the Q-value function. This function estimates the expected reward for taking an action in a given state, guiding the agent to make decisions that maximize cumulative rewards.
- **Why DQN**: DQN has been proven effective in various Atari games, achieving human-level performance in many cases. Its ability to handle complex input data makes it suitable for our project.

Monte Carlo Tree Search (MCTS)

- **Description**: MCTS is a search algorithm used for decision-making processes involving randomness and incomplete information. It builds a search tree based on random sampling of the action space and uses the results to make more informed decisions. By simulating different action sequences, MCTS helps the agent plan long-term strategies.
- **Why MCTS**: Integrating MCTS with DQN can enhance decision-making, especially in complex game scenarios requiring strategic planning. It allows the agent to explore different possibilities and choose the best course of action.

Tools and Technologies

OpenAI Gym: A toolkit that provides a wide range of environments, including Atari games, for testing and benchmarking RL algorithms.

TensorFlow/PyTorch: Frameworks for developing and training neural networks. We will use these to build and optimize our DQN models.

AlphaZero: While primarily used for Chess and Go, the principles of combining neural networks with MCTS from AlphaZero can inspire our approach for Atari games.

Focus Areas

Algorithm vs. Environment

Our primary focus will be on the algorithmic development rather than the environment. While the environment (Atari games) is crucial, our project aims to push the boundaries of RL algorithms' capabilities. We will utilize the well-established OpenAI Gym environments to ensure our agent's performance is measured against standardized benchmarks.

Key Areas of Focus

Algorithm Implementation: Developing and fine-tuning DQN and integrating MCTS for enhanced performance. We will experiment with different network architectures and hyperparameters to optimize the agent's learning process.

Training Process: Ensuring the agent undergoes extensive training to learn from its interactions with the game environment. This involves running numerous episodes, adjusting the learning rate, and implementing techniques like experience replay to stabilize training.

Performance Metrics: Evaluating the agent's performance based on game scores, learning efficiency, and comparison with baseline models. We will track metrics such as cumulative rewards, win rates, and the time taken to achieve certain performance levels.

Comparisons and Evaluations ↗

Comparative Analysis

To validate the effectiveness of our approach, we will conduct several comparisons:

DQN vs. Human Performance: Assessing how the DQN agent's performance stacks up against average human players. This comparison will highlight the agent's capability to learn and execute strategies that humans typically employ.

DQN with MCTS Integration vs. Standalone DQN: Evaluating the performance improvements gained by integrating MCTS with DQN. We expect the combined approach to demonstrate better strategic planning and decision-making.

Baseline Models: Comparing our agent's performance with other established RL models for Atari games. This will help us understand how our approach stands relative to existing solutions.

Evaluation Metrics

Game Score: The primary metric, indicating the agent's proficiency in playing the game. Higher scores reflect better performance and understanding of the game mechanics.

Learning Rate: How quickly the agent improves its performance over time. We will monitor the agent's progress through training curves and determine how efficiently it learns.

Efficiency: Computational resources and time required for training and inference. Efficient algorithms will require less computational power and time to achieve high performance.

Conclusion

This project aims to develop a sophisticated RL agent capable of playing Atari games at a high level. By focusing on advanced algorithms like DQN and MCTS, we seek to push the boundaries of what RL agents can achieve in game environments. The detailed comparisons and evaluations will provide insights into the strengths and weaknesses of our approach, guiding future improvements and innovations in the field of reinforcement learning.