Heidelberg University

End of the Semester Project
Lecture: Machine Learning Essentials

Report

Using Reinforcement Learning Methods to Train an Agent to Play Bomberman

Institute: Computer Vision and Learning Lab
Biomedical Genomics Group

Authors: Leonie Boland, MatNr. 4055040
Kevin Klein, MatNr.
Berkay Günes, MatNr.

Version from: September 27, 2023

Supervisor: Prof. Dr. Ullrich Köthe

1 Declaration

Contents

2 Introduction

The integration of artificial intelligence (AI) into the realm of video gaming has led to groundbreaking advancements in gameplay and immersive experiences. One of the most intriguing challenges in this endeavor is the development of AI agents capable of mastering complex and dynamic games such as Bomberman. Bomberman, a classic arcade-style game, presents a rich and multifaceted environment with intricate decision-making, spatial reasoning, and strategic planning. Solving Bomberman with an AI agent using reinforcement learning techniques is a fascinating and intricate problem, one that holds significant promise for the AI and gaming communities.

Reinforcement learning (RL), a subfield of machine learning, offers a compelling approach to tackle the Bomberman challenge. RL revolves around training agents to learn optimal strategies by interacting with their environment, taking actions, and receiving feedback in the form of rewards or penalties. In the context of Bomberman, RL techniques can be instrumental in enabling AI agents to navigate mazes, strategically place bombs, avoid traps, and outsmart opponents. Here are a few noteworthy reinforcement learning techniques that hold potential in solving the Bomberman game:

Q-Learning: Q-learning is a classic RL algorithm that could be applied to Bomberman. It enables agents to learn a value function that maps state-action pairs to expected cumulative rewards. By exploring different actions in the game and updating Q-values, an AI agent can eventually converge on an optimal strategy.

Deep Q-Networks (DQN): DQN extends Q-learning by employing deep neural networks to approximate the Q-value function. This technique has been successful in handling complex and high-dimensional state spaces, making it a strong candidate for Bomberman.

Policy Gradient Methods: Policy gradient methods aim to directly learn a policy that specifies the agent's actions in different game states. This approach can be effective in scenarios where the optimal policy is not easily represented by a value function, as it allows for a more flexible and direct mapping from states to actions.

Proximal Policy Optimization (PPO): PPO is a state-of-the-art RL algorithm that focuses on optimizing policy functions. It offers stability and strong performance in challenging environments, making it suitable for Bomberman's dynamic and adversarial setting.

Actor-Critic Methods: Actor-critic methods combine value-based and policy-based approaches, leveraging both a critic (value function) and an actor (policy). This combination can provide the agent with a more robust learning framework, enhancing its decision-making capabilities.

In the pursuit of solving Bomberman with an AI agent, the selection and fine-tuning of the appropriate RL technique, as well as the integration of domain-specific features, will play pivotal roles in achieving success. This endeavor not only serves as a compelling testbed for reinforcement learning but also has the potential to elevate the overall gaming experience by delivering AI opponents that are challenging, adaptive, and engaging.

3 Methods

In this section, we elucidate our methodology and approaches concerning the various machine learning models that we have developed for the Bomberman game agent. We will also substantiate what has proven effective, articulate the concepts we have discarded, and expound upon the rationale for ultimately selecting the model we have submitted.

3.1 Our final best model (first project)

Explained by Berkay

3.2 Our second best model (second project)

dwedwedw dwdwdwd

3.3 Our final best model

wdwdwd dwdwdwd

3.4 Other aproaches that we had

frfrf rfrfr

3.4.1 Q-Tables

swsws cxwcw

3.4.2 Coin-Collector Agent that sees only one coin

fsdff wfwfw

4 Training

5 Experiments and Results

6 Conclusion