Dominating Dominion

Abstract

Our project's intent is to develop a reinforcement learning (RL) agent capable of effectively playing the board game *Dominion*. To achieve this, we will use *PyDominion* to establish an environment where the agent can learn. This dataset consists of the game's structured rules and objects. The primary goal is to develop a functional RL agent that can make strategic decisions within the game. Furthermore, we hope to employ different RL methodologies and compare their performance. This study will involve designing an appropriate reward system and addressing integration challenges between *PyDominion* and the RL framework.

Introduction

The primary focus of this project is to explore reinforcement learning techniques by training an agent to play *Dominion*, a popular deck-building card game. *PyDominion* provides an implementation of the game in our language of choice, which serves as our foundation for creating a learning environment. We will construct an RL-compatible environment that allows the agent to iteratively improve its performance.

With the idea of developing an AI agent to play a complex board game, we settled on *Dominion* for its strategic depth, complexity, and stochastic nature. The features make a game that appropriately pushes the limits of our ability while staying within a what we think is a reasonable scope. Specifically, *Dominion's* non-deterministic design adds an extra layer of complexity, making it an intriguing challenge for reinforcement learning. Our goal is to create an RL model that learns optimal strategies through repeated gameplay.

Methods

We plan to develop our RL agent for *Dominion* through the following stages:

1. **Environment Setup**: We will integrate *PyDominion*, to create an environment for reinforcement learning frameworks. This requires defining the observation space (game-state representation) and the action space (available moves).
2. **Reward System Design**: Creating a robust reward function is integral to guiding the agent's learning. It will need to balance short-term vs. long-term gains. The exact structure is yet to be determined but potential reward mechanisms could involve victory points, deck composition, and game progression.
3. **Implement the RL Agent**: We will explore RL methodologies to find good fits. Currently, Deep Q-Networks (DQN) and Monte Carlo Tree Search (MCTS) are appealing but others may be selected depending on our findings. We will then implement these methods as an actor in our environment.
4. **Training the RL Agent**: We will run simulations in the environment to allow the RL agent to learn the game. This will likely involve tweaking our agent implementations and reward systems.
5. **Evaluation and Analysis**: The agent's performance will be measured by its win rate against baseline strategies (e.g., random moves). If time permits, we will compare the effectiveness of the RL approaches.

Challenges we anticipate include:

- Properly establishing effective training given the game's complexity.
- Tuning the reward systems to encourage optimal play.

One important factor in our developing this model is the game-state representation. The table below provides an example of how the agent will perceive and interact with the game environment:

Feature	Example Value	Description
Entire Deck	5 Estates, 7 Coppers	Cards in the player's entire collection
Current Deck	2 Estates, 4 Coppers	Cards currently in the player's deck
Hand	Smithy, Silver, Copper	Cards available for play this turn
Supply Cards	Village (8), Market (10)	Remaining cards available for purchase
Victory Points	3 VP	Player’s current victory point total
Actions/Draws Left	1 action, 2 draws	Available turn actions

Expected Outcomes

If the project is successful, we expect to develop an RL agent capable of playing *Dominion* at a competitive level. The agent should demonstrate a learned understanding of strategic decision-making, such as optimizing deck composition and selecting strong card combinations. Ideally, we will also be able to compare different RL methods to determine their effectiveness in a game like *Dominion*.

Group Considerations

- Explore ML methods (33% RH/ 33% KB/ 33% CW)
- Reward Function (33% RH/ 33% KB/ 33% CW)
- Implement agents: Each group member will implement a distinct RL approach, allowing us to compare methodologies for playing *Dominion*.
    - Implementing agent 1 (~90% RH)
    - Implementing agent 2 (~90% KB)
    - Implementing agent 3 (~90% CW)
- Evaluating performance of the agent (33% RH/ 33% KB/ 33% CW)
- Tuning Reward function (33% RH/ 33% KB/ 33% CW)
- Compare performance of agents (33% RH/ 33% KB/ 33% CW)
- Compile results and create visualization (33% RH/ 33% KB/ 33% CW)
- Produce poster (33% RH/ 33% KB/ 33% CW)

Other things to consider

- **Potential Visualizations**
    - ML Agent Architecture
    - Game-state representation
    - Reward function
    - Agent performance (potentially against other agents/iterations)
    - Game statistics (i.e. average turns, scores, deck size)
- **GitHub:** https://github.com/KirstenBauck/Dominating_Dominion