Training a Dominoes player with Deep Reinforcement Learning

Project proposal

Susana Hahn and Lukas Seiling

Cognitive Systems, University of Potsdam

1 Motivation

Reinforcement Learning (RL) is currently one of the most researched topics in Machine Learning. Over the last decades, systems using this technique have achieved ground-breaking success. In the field of Game AI, Deep Reinforcement Learning has reached super-human game play in complex games such as Go and Chess [3].

Dominoes is a popular game dating back around three millennia. Its history has generated many variants with different complexity. Nowadays, there are several online platforms where users can play this game against the computer. However, there is not much research into how these systems were created. The aim of this project is to create a system using Deep Reinforcement Learning that can play a simple variant of the game of Dominoes.

2 The game

The typical domino set consists of 28 domino tiles with two faces. Each face contains an integer value from 0 to 6 generating all possible combinations between such values. The dominoes are randomly and evenly distributed at the beginning of the game. The game starts with the first player placing one of its dominos on the table to start a chain. On each subsequent turn, a player plays exactly one domino matching one of its faces to one of the two ends of the current chain of dominoes. Only when none of the players tiles can be played, the player is allowed to pass. The game ends when one player wins by finishing all their dominoes, or when there are no more possible moves and the game becomes blocked. The losing player gets negative points equal to the sum of all face values on their unplayed tiles. In the case of a blocked game, the player with less points wins the match.

In this project we will work with a two player game where both players get 14 tiles at the beginning of the game. This variant leads to a game of perfect information where every players are aware of which tiles belong to their opponent.

3 Metodology

3.1 Game representation and model dynamics

Before training the AI agent we must to define the dynamics of the game. This involves the creation of a virtual environment that simulates the game play. This environment will define the states, possible actions and transitions. Eventoght the game is simple, the action space can be difficult to define as the actions will depend on the current tiles of the player and the tiles already played. To approach this, we will consider using indexes on the tiles as proposed by Charlesworth [2] for card games.

To simulate this environment we will use the language Answer Set Programming (ASP) [1] suited for Knowledge Representation. A python wrapper around the ASP system *clingo* will allow us to generate the possible actions given a state and to compute the following states for every action. The state represented with logic predicates will then be transformed into a numerical feature vector for the training process.

3.2 Training

We will train a deep neural network to play the game, through self-play. The different layers of the network should be able to abstract more general concepts of the game. Using Deep Q-Learning (DQL) we will learn an optimal policy defining the game-play strategy.

The history of the game defining which player made which move under which context represents a fundamental role in the game strategy. We will explore how the state representation can be enriched with some of this history to improve the system overall performance. Further, we will compare how features representing expert knowledge on the game can improve the performance as suggested by Tesauro in TDGammon [4].

3.3 Evaluation

The system will be evaluated against a player performing random moves and against experienced human players. Additionally, we will use the available libraries from OpenAI[1], which improve the DQL algorithm, as a benchmark to evaluate the system.

References

1. Christian Anger, Kathrin Konczak, Thomas Linke, and Torsten Schaub. A glimpse of answer set programming. 19:12–, 2005.
2. Henry Charlesworth. Application of self-play reinforcement learning to a four-player game of imperfect information. *CoRR*, abs/1808.10442, 2018.

[1] `https://spinningup.openai.com/en/latest/algorithms/ppo.html`

3. David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484 EP –, 01 2016.
4. Gerald Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Comput.*, 6(2):215–219, March 1994.