Introduction to Artificial Intelligence Final Project - Durak with Reinforcement Learning

Vitaly Rajev, Ziv Mahluf, Idan Yamin, Eyal Diskin

Hebrew University of Jerusalem
August 2020

Table of contents

Introduction

Durak (Russian: , meaning: 'fool') is a popular russian card game. The objective of the game is to get rid of all one's cards when the deck has no cards remaining in it. The last player which has cards in their hand loses and is the 'durak' (fool).
Durak is a flexible game with regards to the rules with which it can be played, and there are many variations of the game as a result, including ones which allow cheating (and it's the players' resposibility to notice in time), attacks of multiple cards, addition of cards after the round is over (in case of failure to defend), and more.
In this project, we've implemented AI agents for a simple variation of the game using reinforcement learning approaches.

Rules of the Game

Durak is traditionally played by two to six players with a deck of 36 cards, which results from a standard 52-card deck from which all cards with values 2 to 5 were removed.
Before the cards are dealt, the deck is shuffled, and a random card is revealed from the deck, the suit of the card (hearts, diamonds, spades, clubs) determines the 'trump suit' for the game, and the revealed card is then plaed at the bottom of the deck (usually face-up).
After shuffling the deck, each player is dealt 6 cards, and if all starting hands are legal, then the game begins, otherwise, the deck is reshuffled and the cards are redealt as needed. A starting hand is considered legal if there are no 5 cards or more with the same suit in it.
After dealing the cards, the player with the lowest-valued trump card is attacking first.
Each game of durak consists of rounds, each consisting of attacks and defences. A round starts with the attacking player placing an attacking card. The defending player is then required to place a defending card, which is a card with the same suit and a higher value, or, if the attacking card is not a trump card, any trump card. If the defending player is unable or unwilling to defend, they take all cards currently on the table. After the first attack, attacking players can only place cards with values that appear on the table for an attack. After the defender plays a defending card, each player, strating from the attacker, and going clockwise, except the defender, has a chance to play an attacking card. If the player sdoes not attack, the next player has a chance, and if the player attacks with a card, the defender has a chance to defend or take the cards. If not player attacked, or the number of cards the defender defended from is 6 (the original hand size), or the defender has no cards remaining in their hand, the cards on the table are discarded and the players draw cards from the deck until they have 6 cards, starting from the first attacker, by order of attack chance, and ending with the defender. After drawing cards, the defender becomes the first attacker, and a new round begins. Any player with an empty hand at the end of the round is considered a winner and is removed from the game, and the attacker is chosen to be the next player clockwise (if needed).
The game ends when the deck is empty and there is at most one player with cards in their hand. If there is only one player, he is considered the loser (the 'durak'), and if there are no players with cards in their hands (for example, if the attacker and the defender had one card each and were the last players, and the defender defended from the attacking card) the game is considered to have no loser.

Approach and Methods

For this game, we've decided to use reinforcement learning techniques to train the AI agents with two different algorithms. Both kinds of agents use a neural network to approximate the Q-value of a state for different actions, but use a different learning method.The use of a neural network to approximate the Q-values of a state different actions was chosen since the number of possible states in Durak is too large to store in memory, even without taking into account memory regarding the cards in other players' hands, the cards in the current player's hands, and discarded cards.

DQN Agent: This agent is trained using the Q-Learning algorithm on a neural network. The neural network takes a representation of a state as an input, and outputs a vector of the size of the action space, in which each value corresponds to the expected reward for doing the action corresponding to the position of the value. The weights and biases of the network are then updated by applying the update formula - $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot (r_t + \gamma \cdot max_a Q(s_{t+1}, a) - Q(s_t, a_t))$ (where α is the learning rate, γ is the discount factor, and $max_a Q(s_{t+1}, a)$ is the maximal possible expected reward from the next state) - to the output of the network, and using gradient descent to update the weights and biases themselves.

PPO Agent: ———————————

Training and Results

Conclusions

How to Run the Code

References

BONUS PART - DANK MEMEZ