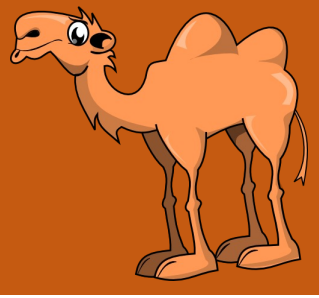


Implementation of an Intelligent Agent for the Camel Up! Board Game

Alec Gironda & Luke Lorenz

CSCI 0311 – Fall 2022
Department of Computer Science

Project Goal

The goal of this project is to develop an intelligent agent for the board game Camel Up. Using reinforcement learning and an expectimax algorithm, our agent should be able to determine the optimal policy at each game state and take actions accordingly.

Background

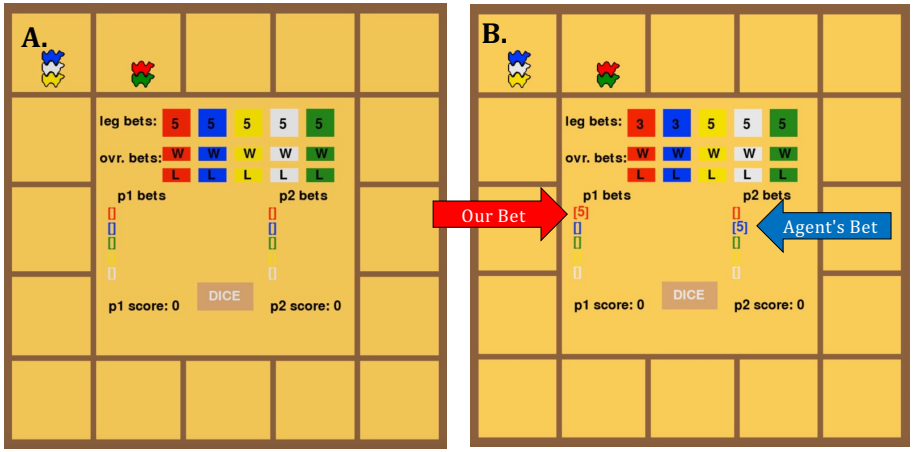
- Board games have long been used as a benchmarks for AI progress, notably those with imperfect information and stochasticity.
- Camel Up contains a restricted decision space, but many probabilistic events, making it a novel challenge for AI development.
- While a probabilistic AI for Camel Up has been developed, we sought to combine reinforcement learning with an expectimax algorithm to push the envelope of novel AI in dynamic board game situations.

Gameplay

- 2 players bet on five racing camels. The player with the most money at the end of the race wins. The race ends when any camel returns to the start.
- There are 5 colored dice—one for each camel. Rolled dice move the corresponding camel and any on top of it.
- If a camel lands on an occupied space, it stacks on top of the camels already on the space.
- The furthest camel leads, with ties resolved by tallest height.
- Each turn, the player chooses to either randomly roll one of the remaining dice, bet on a leg winner, or bet on a race winner/loser. If a die color is rolled, it can't be rolled again.
- A race 'leg' ends when all five dice have been rolled. Then, a new leg starts, clearing all leg bets and returning all dice to the shaker.
- Rolling a die pays 1 coin, betting on an overall winner/loser pays 8 or 5 coins, and leg bets pay 5, 3, or 2 coins. The earlier a bet is placed, the higher it pays.
- At the end of each leg, correctly placed leg bets pay out. Correct overall winner and loser bets pay out at the end of the race.

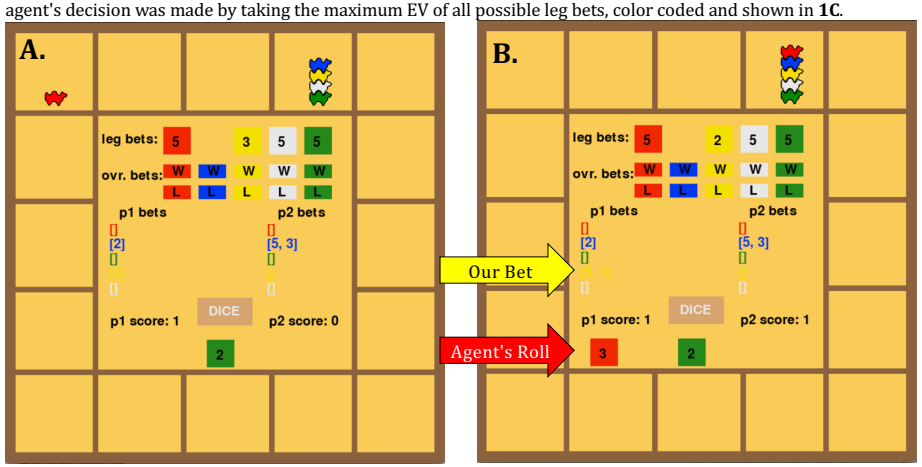
Methods

- Our final AI agent, SmartPlayer, chooses moves based on the current state of the game using a combination of a neural network and an expectimax algorithm.
- We calculated the expected values (EV) of leg bets and made moves based on the highest EVs across the leg bets.
- When each leg bet's EV was less than the EV to roll a die, 1, the neural network predicted if betting on a final winner or rolling a die would be more advantageous.
- We trained the neural network for SmartPlayer on 100 simulated games between RandomPlayer (takes moves at random), and MaxPlayer (only uses expectimax), and observed actions through a pygame interface.



C. EV's: [0.740, 0.533, 0.566, **1.599**, 1.06]

Figure 1. **1A** is the board state before we take our turn. We decide to place a leg bet on the red camel. **1B** shows the agent's decision to place a leg bet on the blue camel. The decision is sensible: blue lies on top of two camels who have yet to roll, increasing the chance of blue being carried forwards while retaining position over those below. The agent's decision was made by taking the maximum EV of all possible leg bets, color coded and shown in **1C**.



C. EV's: [0.380, 0.606, 0.0, 0.0, 0.781]

Figure 2. **2A** is the board state before we take our turn. We decided to place a leg bet on the yellow camel. Now, it is the agent's turn. **2B** shows the agent's decision to roll. It rolled a 3 on the red die, moving the red camel 3 spaces. The agent chose to roll because none of the EVs of leg bets in **2C** were greater than 1, and the agent's neural network predicted that the EV of rolling was greater than the EV of placing an overall winner/loser bet at this time.

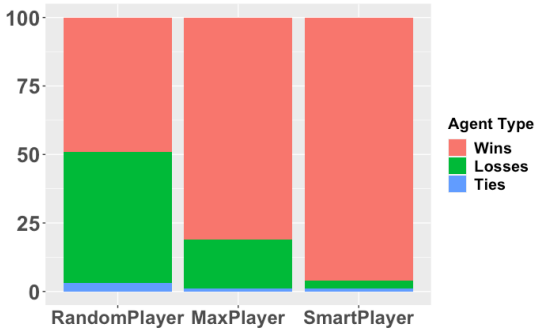


Figure 3. Results of 100 simulated games between various agents and RandomPlayer. Agents are briefly described in Methods and further described in Abstract Table 1.

Results & Conclusions

- We simulated games with RandomPlayer, MaxPlayer, and SmartPlayer against a RandomPlayer. Based on wins and losses over 100 games for each matchup, SmartPlayer outperformed MaxPlayer, and MaxPlayer outperformed RandomPlayer. Figure 3 displays these results.
- Prioritizing leg bets is advantageous, for correct leg bets allow a player to quickly accumulate large sums of money. MaxPlayer and SmartPlayer took advantage of this, hence their dominance over RandomPlayer.
- Betting on an overall winner/loser before the other player is more advantageous than rolling, early in the game. SmartPlayer often did this, hence SmartPlayer's dominance over MaxPlayer.
- A strategic flaw in our agent is that it takes bets to maximize payout when it's already winning by a large margin. This gives the other player opportunities to win bets and close that margin. Instead, our agent should roll to try to end the game sooner.
- Future research extensions involve expanding the game to 4 players, adding other game mechanics, and addressing this strategic flaw.
- We view this early demonstration as a stepping-stone for advanced machine learning techniques for even more complex board games in future research.

References & Acknowledgements

• Baron, Tyler. "Sir Humpfree Bogart" (2018). https://github.com/trbarron/Camel_Up_Cup_2K18/blob/master/Sir_Humpfree_Bogart.py
• Blomqvist, E. (2020). Playing the Game of Risk with an AlphaZero Agent (Dissertation).
• "Camel Up Rules". F.G. Bradley's, https://www.fgbradleys.com/game_rules.asp
•De Mesentier Silva, Fernando & Lee, Scott & Togelius, Julian & Nealen, Andy. (2017).
•AI-based playtesting of contemporary board games. 1-10. 10.1145/3102071.3102105.
•Heyden, Cathleen. (2009). Implementing a Computer Player for Carcassonne.

This work was made possible by computational resources, data, and expertise provided by Professor Linderman.