Lucas Ege
Zac Espinosa
Isaac Westlund

Learning to play an adversarial game of BS (cheat)

Our focus is on the game of BS, otherwise known as Cheat, which is a card game where the goal is to get rid of all your cards. The specific rules outline that you can only remove cards on a "cycle" i.e. the first player can discard 1's, then the next turn the second player can discard 2's and so on. A player can discard whatever and however many cards they want - even cards that are not on the current cycle. This introduces the interesting part of the game for us - the deception of playing non-cycle cards and calling them out as cycle cards. If another player "calls their bluff" on this, the bluffing player must pick up all the cards in the discard pile, or if the player wasn't bluffing then the calling player must pick up all the cards. We plan to develop an agent that will play a game of BS to reach a high level of accuracy when calling out bluffs by other players and "calling out" reasonable bluffs in order to achieve a high probability of winning.

Our inputs for this would come in multiple possible forms. One such form could be entire game state logs, where we outline what cards were played under what pretense (which we could use to determine if it was a bluff or not), and a classification of if it was called out for a bluff or not. Our eventual output from this would be to take a game state up to a certain point and a next move from an adversary and then output confidence of a bluff. We also plan to play games against the agent utilizing reinforcement learning with a reward function contingent on discarding all cards in hand, with input being our play and output being a chosen best move (between play or call "BS"). Our simulation for this game is a fairly simple text-based engine that can be used to test and train our agent against data inputs or a human player.

Success might be defined in a couple ways: namely, prediction accuracy of "BS" calls against actual cases of BS, or efficiency of games won vs. games played. Our baseline is to never cheat unless necessary (i.e. no cards left, in which case the agent could either pick up all cards or play 1 random card), and it would never call out a cheat. This would obviously be a lower bound since it would not be utilizing any of the game's mechanics in any intelligent way.

The oracle for this player could actually be another agent that we might develop. While we are attempting to create a learning agent, making an agent that plays the ideal move in every possible situation might not be extremely difficult as we could just develop an agent that counts cards played and keeps track of who picks them up, after a certain point in most games, this agent will know every card and will be able to call BS correctly every time. If we do not end up developing this agent, we can simply say that oracle level is simply correctly calling every other player's BS.

To actually implement the agent we plan to test and report on a variety of methods, at the moment as we have not covered much else that is applicable, we are thinking of using primarily

MDPs, and Q-learning. As we cover more advanced topics later on in the course we will develop and test those as well.

There are a variety of challenges that we might face developing a BS playing agent. First, and foremost, as BS is not exactly a huge priority in industry, there is essentially no pre-collected data on the internet that we would be able to utilize. We plan on creating all data-points necessary ourselves. During initial development and testing we will most likely be feeding the agent random decisions for the previous player. As the agent gets more advanced we will train it against one or multiple instances of itself to train on. Eventually if we do create the oracle state machine as specified above, we will train it against that as well to see how close a learning agent could get to ideal play.

Related works:

https://arxiv.org/pdf/1609.05559.pdf - This paper addresses the topic of an adversarial game and predicting and modeling an opponent's behavior, which will be a part of what we are hoping to accomplish.

https://arxiv.org/pdf/1709.09451.pdf - This paper specifically talks about using Monte Carlo Tree Search to handle imperfect information games, such as BS.