CS6240 Parallel Data Processing (Spring 2021) Chris Amato

Students: Gregorieu, Juan Agustin - Vigliecca, Guzman

Project Proposal

1. What problem are you solving?

   Our project aims to create an AI agent that can play the Truco card game. We will only work with the 2-player, Argentinian version of the game, but it can be played with 4 and 6 people too. The game is a turn-based game and it is played with Spanish Cards. We will omit the game's detailed explanation since it can be pretty long, but it can be found in [1]. To put it shortly, each player is dealt 3 cards, then a first special round is played. This round is called "Envido" and any of the players can say "Envido" in its turn. If the other player accepts, the player with better cards wins 2 points, if the other player declines, the person who called it wins 1 point. Then the "Truco" round is played, in which, one by one, each player puts down one card, and the player with better cards wins. Similarly, as in the "Envido" stage, any player can say "Truco" and if the other player accepts, then whoever wins in the end, wins 2 points. If the other player declines, the player who said it wins 1 point. It is a game of bluffing because if my cards are petty bad, I can say "Truco" or "Envido" to make the other player think I have good cards and call of the round if the player declines.

2. Describe the problem formally from a computational perspective. What are the inputs and outputs (exactly)?

   In the Spanish Deck, there are 4 suits: Coin, Cups, Swords, and Batons. Truco uses every suit, and numbers from 1 to 7 and 10 to 12.

   - Initial State: The initial state S_0 is composed by:
     (a) The agent's hand of 3 cards.
     (b) The agent's score at the moment. At start it will be equal to zero.
     (c) The round number. The initial value is 1.
     (d) A boolean value indicating whether is the agent's turn to play or not.
   - PLAYER(s):

$$PLAYER(s) = \begin{cases} Agent, & \text{if } ((s-1).round < s.round \wedge (s-1).winner = Agent) \\ & \vee((s-1).player \neq Agent) \\ Opponent, & \text{otherwise} \end{cases}$$

   - ACTION(s): The possible actions the agent can perform will be a subset A' of $A = \{Envido, Truco, Wanted, Notw$
     An action will be included in the actions set for an specific game state if they follow the following rules:
     - The Action $Envido \in A'$ if the player is in round 1, and Envido has not been played before.
     - Both actions, $Wanted$ and $NotWanted$, will be the only ones included in A' if $Envido$ or $Truco$ has been played on the previous round by the opponent.
     - The action $Fold$ can be played always.
     - The action $Truco$ can be played only if it has not being played before.
   - RESULT(s, a):
     - If the action is $Fold$, the game is over and the opponent wins a point.
     - If $Truco$ was played by the opponent, we replied with $Wanted$ and we lost the hand. Opponent wins 4 points.
     - If $Truco$ was played by the opponent, we replied with $Wanted$ and we win the hand. Agent wins 4 points.

- If $Truco$ was played by the opponent, and we replied with $NotWanted$. Opponent wins 1 point.
- If $Envido$ was played by the opponent, we replied with $Wanted$ and we lost the bet. Opponent wins 2 points.
- If $Envido$ was played by the opponent, we replied with $Wanted$ and we win the bet. Agent wins 2 points.
- If $Envido$ was played by the opponent, and we replied with $NotWanted$. Opponent wins 1 point.
- All the above rules apply on the opposite scenario, when the agent plays $Truco$ or $Envido$ and the opponent has to reply.
- Playing a card in a round where the other player has already played, ends the round.

- GOAL_STATE(s):
  - If in the previous state a $Fold$ action was played. The game is over.
  - If is the second round the agent wins or the second round the agent loses. The game is over.
  - If agent replies to $Truco$ with $Nowanted$, the game is over. Same applies to if the opponent replies with $Nowanted$ to $Truco$.
- UTILITY(s): We are thinking about creating a heuristic that takes into account the cards in our hand and their strength. And combines this measure with the current score in the game.

3. What data are you using (exactly)?

   We do not need any data that cannot be generated within the game itself. As with the Pacman project, the agent should be able to play using the digital version of the game.

4. Why is it interesting?

   First, we have a personal interest since it is a game that we love and it is very traditional in our countries. We also couldn't find any agent already done for this particular version of the game (there are a couple for the Brazilian Truco but it's not the same) which will make the project really challenging and innovative. Finally, any good Truco player knows that you are as good in the game as you are bluffing, so we want to see if an algorithm can actually play this game well.

5. What algorithms do you use?

   We intend to use three algorithms:

   (a) Minimax (with alpha/beta pruning)

   (b) Expectimax

   (c) Reinforcement Learning

   And a combination of the above algorithms depending on the particular scenario we are facing.

6. Why are these algorithms appropriate?

   The game is turn-based, partially observable (since we don't know the other's player card), and it has a score, which makes Minimax and Expectimax suitable. The fact that we have a score that directly translates to rewards in Reinforcement Learning together with the fact that we don't have a large database of recorded games that we can use in supervised Machine Learning, makes Reinforcement Learning appropriate too.

7. How are these algorithms typically used, and how are you using them?

(a) Minimax (with alpha/beta pruning): Minimax is typically used in game theory when our intention is to build an agent that minimizes the possible loss for a maximum loss scenario. The agent assumes that the opponent will always make the best decision possible. In our problem, the agent will be the maximizer, and the opponent the minimizer. And we will use a measure of utility in function of the current score and the strength of the cards in our hand. This will be made by Guzman Vigliecca.

(b) Expectimax: Expectimax is a variation of Minimax. The idea is to take into account the probabilities of certain outcomes when deciding the next action. Since we do not know what cards the opponent has, but we can make a guess based on our current hand and the cards already played, it makes sense to use this algorithm. This will be made by Agustin Gregorieu.

(c) Reinforcement Learning

8. Have other people use similar algorithms to solve your problem before?

   We could only find two public Github repositories that supposedly are an AI agent for Brazilian Truco ([2] and [3]) but they don't even have a readme so it is likely that they are of no use to us. There is a pretty interesting paper that proposes a Markovian model for the Brazilian Truco [4] but it is a much more complicated version of the game, and they don't use any of the algorithms we propose here.

9. What results do you expect to show?

   The idea is that our agent can play against us in a rational way. We are not expecting it to be flawless, since Truco involves a lot of bluffing. But we expect to see intelligent responses.

10. What comparisons will you do?

    There is not a straightforward way of measuring the agent's performance since we don't have an already existing agent to play against, so we plan to do two things: make the agent that uses Expectimax and Minimax play against the one that uses Reinforcement Learning to see which one is better, and then, play ourselves against our agents to see how they perform against a human.

11. Are there risks for not getting all the results?

    We expect that Minimax and Expectimax work. We cannot assure that the agent will always win or make the best decisions. But it should play the game. In case of RL, the risk is that if Expectimax or Minimax are playing poorly, then the trained agent will not learn to play properly.

12. If so, what will you do about it?

    If the results are not good because the agent never wins against a human that plays well, we won't consider it a failure, rather a result that shows that it is not an easy task to learn this game and perhaps a more complicated model such as [4] proposes is needed.

References [1] - https://en.wikipedia.org/wiki/Truco [2] - https://github.com/willyandan/truco.ai [3] - https://github.com/mayleone1994/TRUCO-GAME-WITH-AI- [4] - https://www.sbgames.org/proceedings2020/Computacac