CS 4710 Final Project: Youngster Joey AI

Jaden Kyler-Wank, Drew Goldman, Peter Shin, Alex Shen

December 9th, 2021

1 Introduction

Pokémon is the highest-grossing franchise in the world, known for its simple yet captivating video games in which players enter the world of Pokémon to train and battle against Pokémon. The main attraction of the Pokémon games, especially to computer scientists, is the turn-based battle format in which two (or four) players battle each other until one trainer (or team) has no Pokémon remaining. Due to the increasing complexity of the Pokémon games over time, we decided to focus on the battling mechanics of the first generation (Generation I), which featured 151 Pokémon. Since one Pokémon battle has such an unbelievably large number of possible outcomes (even if the moves are played deterministically!), we were determined to craft an artificial intelligent agent — Trainer Youngster Joey — through both experience in strategy and extensive testing, to triumph against other human trainers.

2 Problem Description

Our project attempts to find a semi-optimal battle strategy for Generation I Pokémon battles. From our knowledge and given the stochastic nature of the game, we believe Pokémon is not, and may never be, a solved game. There exist various strategies for artificial intelligent bots, but thus far none has proved to be optimal. Note the meaning of optimal and unbeatable differ here, as an optimal AI can play and still lose to a non-optimal player due to the stochastic processes of the game. Formally, we shall define the two opponents in a battle as Player 1 and Player 2. Unless otherwise stated, we shall be referring to Generation I Random Battles, in which case we assume each trainer to begin the battle with a team of six "random" Pokémon. Each Pokémon will have a level [1, 100]. Further, each Pokémon species has four base stats in each of attack, defense, special, and speed [5, 255], which determine the Pokémon species' abilities. In addition, each individual Pokémon has IVs (Individual Values) [0,15] and EVs (Effort Values) [0,255] that modify these base stats, which are unique for each Pokémon based on its genetics and its past training respectively. Thus, in the generation of random teams, a team of six Pokémon that appears to be identical to another team of six Pokémon will probabilistically be different in some way, which introduces vast complexity to the space of all possible outcomes. Each Pokémon has up to four unique "moves," which can be damaging moves or status moves. As the goal of the game is to faint all opponent Pokémon with attacks, an artificial intelligence agent must use primarily damaging moves in the battle; however, in newer Pokémon generations it is possible to win battles using a larger proportion of status moves, as the dynamics have become more complex. Damaging moves have a base power and a base accuracy, and when combined with the following variables in Figure 1, a random damage value is calculated:

$$Damage = (\frac{(\frac{2*Level}{5}+2)*Power*A/D}{50} + 2) * random * STAB * Type$$

Figure 1: Damage Calculation

Where critical hits cause the level variable to be doubled, A is attacking stat, D is defending stat, and the random variable is a random integer [217,255]/255. Status moves can boost or lower stats, inflict status conditions, heal HP, and a whole host of other effects.

Recall that a policy is a mapping from a state S to an action A, while a strategy is a selection of a policy. We shall define our strategy YoungsterJoeyAI as an ordered sequence of actions to be executed by the trainer. The set of all possible actions includes $\{move1, move2, move3, move4, switch2, switch3, switch4, switch5, switch6\}$. More concretely, the goal is to create an optimal strategy — one in which the actions performed at each timestep (the future is independent from the past given the present) maximize the probability of winning the battle. Necessarily, a strategy may have a higher probability of winning by sacrificing one's own Pokémon in order to preserve the health of certain other comparatively advantageous Pokémon. As such, the precise definition of probability of winning a battle is ambiguous to say the least.

3 Solution And Potential Improvements

There were multiple implementations that we initially determined may be options for our AI. One possible implementation was a minimax approach. However, the game will rapidly become too complex for the game tree to expand at a reasonable speed. This could potentially be optimized using alpha-beta pruning, but since there isn't a guarantee that the adversary takes optimal moves, this approach would require constantly reconstructing the game tree anyways or else the agent might act unexpectedly.

Another potential implementation was using expectimax. We thought that this would be an appropriate solution given some of the limitations of a minimax, such as the fact that our agent is most likely playing against a non-optimal adversary. However, expectimax is not guaranteed to play optimally if it plays against a non-optimal adversary, which is true in our case. It could potentially lose if a choice is made upon a probability that does not lead to a higher utility. Also, we run into the same issue regarding the game tree becoming much too large for any of our computers to handle.

Based on these considerations, we believed that a greedy policy with a possible look-ahead would be the most feasible agent that would maximize the probability of winning. In our greedy policy, we calculated a score for each possible action, utilizing a heuristic which considers many aspects of the game. Once every choice is calculated, the AI selects the action with the highest score, and this is repeated until a winner is determined (i.e. the state in which all of a player's Pokémon are fainted). More formally, our heuristic takes the following factors into account: Damage, Move

Accuracy, Primary Effect, Secondary Effect, Critical Hit Probability, Type Advantage/Disadvantage, Move Priority and Switching Pokémon. Damage is calculated by the method denoted above in Figure 1. Move Accuracy is the probability that a certain move will hit the opponent. Primary Effect refers to any move whose primary purpose is to inflict a status effect on a Pokémon (paralysis, burn, sleep, etc). Secondary Effect refers to damaging moves which have a specific probability to inflict a status effect. Critical Hit Probability is the likelihood that the attack will be a "critical hit" that does extra damage, and is largely determined based on a Pokémon's base speed stat. A Type Advantage is a situation where the AI's Pokémon's typing is favorable against the opponent's Pokémon such that it will resist the moves of the opponent's Pokémon's type (for example, Water-type Pokémon take less damage from Fire-type attacks). Switching Pokémon is actually factored in as a discount to the heuristic, with the discount increasing the lower the AI's active Pokémon's HP is. These factors are used in the AI's heuristic with the following formula:

$$Heuristic =$$
$$[(Accuracy/100) * (expectedDamage + (expectedDamage * critProbaility) +$$
$$secondaryEffect + primaryEffect] * SwitchDiscount * Typing * Priority$$

Figure 2: Heuristic Calculation

With a look-ahead feature, the runtime of our calculations would increase exponentially per step. Our code did not implement a look-ahead feature, but it is important to note that it could allow the agent to make more informed decisions and could have been a way to optimize the agent in a way where we define optimality, whether it is to finish the game in least moves, finish the game with most help, etc.

4 Results

While we were unable to pit Youngster Joey AI against any of Pokémon's top players, we collected statistics about the AI based on our fights against the bot. Each of us played 25 test games (100 games total) against Youngster Joey, with the AI achieving a win rate of 62%. Although we are by no means experts at playing Pokémon, we would consider ourselves near the level of the average player based on our ratings on one of the most popular Pokémon battling website, Pokémon Showdown. Of these 62 games won, about 73% of them were won while the AI had a team with favorable typings against the player. That is, the AI's Pokémon had more Pokémon that were super-effective and/or resistant against the player's Pokémon than the player had against the AI. These results make sense based on our greedy policy and heuristic calculation. Furthermore, we found that Youngster Joey would act more optimally in situations where the AI was already in an advantaged state, likely due to a more obvious action choice. We also found that the AI was very adept at switching its Pokémon to a better one when faced with an unfavorable type matchup, even after the calculated discount the heuristic would receive based on switching.