Pokemon Showdown Alpha-Beta Minimax AI bot

Kyle Peterson
CS4804

Problem statement and Analysis

Pokemon showdown is an online browser game created in 2012 and it allows players to simulate pokemon battles online for every generation of the game series with a myriad of different game modes. This project focused on the specific gamemode of generation eight random single battles. The original scope of this project was for generation nine random single battles; however, due to continuous updates of this format, the data was constantly changing which posed a serious problem for making a reliable project. Because of this, it was decided that switching to a gamemode that is not changing anymore would be better. In generation eight random single battles, each player is given six random pokemon with varying movesets, items, and abilities.

The rules of the game are as follows. Each player has six pokemon, the game is won if the player causes all six of the opponent's pokemon to faint or the opponent forfeits. For every turn of the game each player chooses an action, those actions can be make a move, dynamax (can only be used on one pokemon, active for three turns, can only be used once per game per player), or switch to another pokemon on their team if there are any available.

In the game mode chosen, each player is assigned 6 pokemon with no repeats from a pool of 458 possible pokemon and each pokemon has anywhere from two to eight variations of movesets, items, and abilities. This mean each player has over 1 billion possible team combinations. When the game starts each player makes a move at the same time, each player has up to 13 potential actions, 4 moves + 4 dynamax moves (if active) + 5 possible switches. This means each turn has up to 169 possible successor states. Besides moves a bot must be able to take into account type matchups, stat distributions, field effects, status conditions, abilities, and items. These do not increase the amount of gamestates but rather impact the favorability of successor game states.

Due to the large number of successor game states and factors to consider for each game state, the approach I felt was the most appropriate was the use of a minimax search algorithm utilizing alpha-beta pruning. The algorithm will assume the opponent plays optimally and will choose the best action based on that move the opponent is assumed to make.

Use Case Scenarios

The use cases of this project are as follows, the bot can be used as a reference of players to learn optimal strategies and actions in situations they may not have considered. It can also be used as an opponent against other players or yourself. The algorithm and surrounding can be applied for other online games. Finally, the algorithm being used is one of many possible implementations, meaning this bot can be used to compare performance of other algorithms in the situation of online pokemon battles.

Literature Review

I didn't reference any literature for this project, rather I found similar projects from github and youtube and utilized the information I found from them in order to determine the direction I wanted to go for this project. Listed below are some of the similar projects I found the most influential or useful.

- rameshvarun/showdownbot: A Pokemon Showdown ...GitHubhttps://github.com › rameshvarun › showdownbot
- https://github.com/RemptonGames/Pokemon-Showdown-Agent
- https://www.youtube.com/watch?v=rhvj7CmTRkg&t=0s

Algorithm

The algorithm the bot utilizes is the minimax algorithm with the use of alpha-beta pruning. The algorithm will search up to a depth of 3 and use a game engine that does all the damage calculations for each possible outcome as well as factoring in type matchups, stat distributions, field effects, and status conditions for both the player and the opponent. Each of these possible outcomes is assigned a score and the bot will choose the action with the best possible action based on what it assumes the opponent will choose for the next three turns. The bot will assume that the opponent is playing optimally against the player and will choose actions that will maximize the score for the player and minimize the score for the opponent. As the bot searches the tree is will prune off branches that look unpromising for the bot.

Results and Demonstration

To measure the performance of the bot, I had the bot play 100 games. I then recorded the win loss record over those games. After the bot finished I checked the player ratings of the bot provided by Pokemon Showdown. Pokemon Showdown tracks three ratings for each player, those be ELO (this is the main rating), GXE, and Glicko 1.

ELO is the earned rating of the player, each player starts win 1000 ELO and will earn more if they win and lose ELO if they lose. The amount of ELO gained or lost is dependent on the ELO of the opponent. If the opponent has a higher ELO than the player, the player will gain more if they win and lose less if they lose and vice verse if the opponent has a lower ELO.

GXE is a rating that estimates the chance the player would have of winning a game against the average player on the competitive player. If the chance is over 50% you are considered above average and if the change is below 50% you are considered below average.

The Final rating is Glicko 1, this rating is an estimate of what your rating should be based on the actions of each game. The scale is the same as ELO but it is an estimate of what your ranking should be not what you have earned. The Glicko rating gives you an ELO rating as well as a plus or minus value. This plus or minus value is stating the player should be within plus or minus the estimated ELO rating.

After running the bot for 100 battles I retrieved these metrics for the bot and as well compared those ratings to those of the top ranked played and the 500th ranked player. The reason I chose to compare to those is because the only publicly available ratings are those of the top 500 players for each format. The win loss ratio of those player unfortunately were not available.

Rating	MiniMax AI	Rank 500	Rank 1 player
W/L Ratio	57/43	N/A	N/A
ELO	1302	1499	1705
GXE	59.3%	N/A (set to private)	86.7%
Glicko 1	*1571 ± 33*	1737 ± 127	1854 ± 28

For further evaluation I recorded qualitative observations of the bots decision making process. The bot was effective in choosing good moves when trying to damage the opposing pokemon, it generally chose optimal moves. The bot also used sophisticated strategies when switching pokemon. It would often bait the opponent into using moves that the incoming pokemon was immune to or would negligible damage. Another strategy it used consistently was switching to pokemon that would negate field effects the opponent set up, an example of this would be switching to a pokemon with the "drought" ability which would set up the sunny weather effect and get rid of any other weather conditions on the field. Another positive for the bot was it was ability to effectively keep track of or predict the items and abilities of the opposing pokemon which helped lessen the chances of making blunders in the decision making process.

However, the bot was not perfect. Due to the search depth of the algorithm being three, it had difficulty formulating and detecting long term strategies unless the pokemon it currently had out was capable of winning the game by itself. The bot also had trouble making strategies to deal with status conditions and some field effects, the biggest example of this being that it would neglect the opponent using substitute or when the opponent set up reflect and light screens. This allowed for the opponent to have a series of turns where the they had the advantage. Finally the bot would often neglect strategic threats or set up strategies in favor of switching pokemon or increasing its own stats, this would often lead to significantly unfavorable game states often leading to losses.

Demo - Video Demo of the bot in action
https://drive.google.com/file/d/1hLLESXnuW8u7DrhtFhQfsg2XOm5h1JXa/view?usp=share_link

Code and Documentation

Github - https://github.com/kpp91302/CS4808Project

Lessons Learned

I learned two main lessons from this project, the first being that creating logic for AI agents isn't particularly difficult. The agent just needs the proper information in order to perform effectively. When I first starting this project I had two options for this bot, I could either create the logic for the agent and hand feed it the information I see while playing or I could utilize the existing showdown API in order to relay the information to the bot. I ended up choosing using the showdown API to relay the information to the bot which was significantly more effective but it was significantly harder to implement.

The second lesson I learned from this project is that there is never a correct approach when it comes to AI. There were a multitude of implementations for similar projects and I found that some may performed better than others in the ratings, the other implementations still performed better in other ways that may not be as noticeable. This tought me that when approaching a project like this, it is more beneficial to use more than one approach and use the information I gain from each to make a more effective bot.

Future Work

Currently the bot is insufficient to reach the original goal of playing Gen 9 random battles, this is due to the fact that this is the active generation of pokemon has received multiple updates in the game as well as the distribution of moves, items, etc has been updated for many of the pokemon so it was unrealistic to provide the bot with accurate, up to date data. Once these continuous updates slow down I would like to adjust the bot to able to play gen 9 games reliably

The current bot currently uses the Alpha-Beta pruning miniMax algorithm. I would like to create alternate bots that use other algorithms or methods and compare the performance of these implementations to the current bot. Other implementation strategies I would like to look into are reinforcement learning, monte carlo search trees, and neural networks. There are few examples of these strategies already being implemented with some success and I believe if I can compare these implementations I will be able to gain some insight into what strategies are effective in random battles on pokemon showdown