Proposal: Pokémon Battling with Heuristic Search

Danny McClanahan

Vanderbilt University

January 26, 2017

1 Abstract

Pokémon battling is a popular form of competitive gaming. Heuristic approaches have been applied to play Pokémon, but have not been very successful due to a large search space and stochastic effects. Monte Carlo Tree Search is known to be more effective on such problems. We will apply Monte Carlo Tree Search to play Pokémon and evaluate its success.

2 Background & Significance

Pokémon battles have become a popular form of competitive gaming worldwide. The games expose a two-player turn-based battle system. Both players start with a team of Pokémon, each with some amount of health. Each turn, each player chooses an attack on an opposing Pokémon, and play ends when one player has no healthy Pokémon left.

Choosing the best action on every turn requires knowledge about the opponent's Pokémon and their likely choice of action on that turn. This is very difficult to analyze for both human and computer players due to the large search space and stochastic events that can occur during a turn. The opponent draws a team from 802 different Pokémon and can manipulate their strength and other variables. Each Pokémon has 4 choices of attack chosen from up to 719 total attacks. In addition to inflicting damage, attacks may have various other effects over a turn, some stochastic. It is obvious that the large search space and stochasticity of a Pokémon battle disqualifies any form of exhaustive search for a win condition.

However, expert human players apply many heuristics to make choosing the best action more tractable. Drawing from personal experience, we know Pokémon has a canon of generic tactics, similar to chess. However, unlike chess, the Pokémon developers have specifically designed some Pokémon and attacks

to exhibit unusual effectiveness for one or more of these tactics, and others to be unusably weak. As a result, a much smaller subset of Pokémon and attacks are likely to be used, and those outside of that set are usually less effective. A strong human player can choose actions to satisfy the goals of some abstract tactic, instead of searching directly for the win condition. This makes play more tractable.

While there are previous well-documented attempts to develop a computer player, they have not been well-tested or play poorly. Most rely on manually-created heuristics [1, 2], and while an expert player may rely on some heuristics for success, they may be hard to codify correctly or enumerate completely. Additionally, the problem of automated team selection has been ignored or reduced to blindly copying selections of teams seen before instead of using gameplay experience to generate better teams [1].

3 Research Plan

We will develop a program to play Pokémon which uses Monte Carlo Tree Search to choose the best move at each turn when given a team in advance. If time allows, we will then attempt to use the results of training to generate better teams.

3.1 Hypotheses

1. Monte Carlo Tree Search is a good approach for Pokémon battles.
2. Pokémon teams generated with Monte Carlo Tree Search are better than human ones.

3.2 Rationale

Pokémon battles have a large state space and human experts reliably win by accomplishing complex and abstract objectives. Team creation is even more abstract, and difficult for the same reasons. Monte Carlo Tree Search does not use an explicit evaluation function for individual game states, which we expect equips it well to cope with the state space and stochastic mechanics of Pokémon battles.

3.3 Tools & Data

The program will be written in Python, and Monte Carlo Tree Search will be implemented with a neural network using the TensorFlow library developed by Google [3]. There are code samples publicly available implementing this type of model [4], as well as prior work with examples of more elaborate extensions we can make [5].

Encoding game state as input to the search algorithm may be nontrivial due to the number of different game mechanics, and will likely require multiple

iterations. We will draw on previous work which accomplishes this task. We may evaluate the use of heuristics in successive iterations, but for the reasons stated above we will probably avoid this.

Most competitive play occurs on the open-source simulator Pokémon Showdown [6]. The program will battle by interfacing with the simulator, so we will not have to implement all game mechanics. The simulator allows each player to request any number of battles at any time, and the most popular servers have thousands of battles playing at once. The program will accumulate thousands of complete battles in less than a day. We may advise the program to choose less than optimal actions and see if that makes training more effective.

We will also play the program against itself, which will allow training much more quickly than waiting for a human player. Even though the program will only have a finite set of teams to choose from compared to human play, we expect the stochasticity and complexity of effects attacks may have in Pokémon battles to produce useful variation. Once we implement team generation, we expect self-training to become more useful.

Skilled Pokémon players very often "retire" successful teams and release them to the public in a format that can be fed directly into the simulator. We plan to do web scraping or just manually obtain these teams to use in training before we implement team generation.

3.4 Evaluation & Iteration

We will evaluate the accuracy of the search by comparing expected win probabilities to actual wins, a typical method of judging Monte Carlo Tree Search [5]. In addition, we will also use the program's simulator rating (similar to ELO) against human players as a proxy for the success of the approach.

We will log data about the opponent and game state over every battle. If we score poorly by our evaluation metrics, we will analyze these logs as well as manually inspect a small number of games. If any patterns arise, they can hopefully be used to improve the state encoding or to help prune the search tree more quickly.

3.5 Timeline

Phase 1: Implement MCTS and Game Mechanics *February*

A program will be created to implement Monte Carlo Tree Search and interface with the battle simulator. This phase is complete when our program can choose from any of a number of teams and train itself against human players automatically.

Phase 2: Iterate, Implement Team Generation (?) *March*

We will evaluate the program's performance and analyze logging to identify failures in our model and iterate. If time allows, we will develop an approach to adapt our program to search for a team of Pokémon. This

phase is complete when we have evaluated performance over a sufficiently large dataset for every iteration of the program.

Phase 3: Iterate, Write Up *April*

Write up results, making a quantitative statement on our hypotheses. If successful or interesting, announce to Pokémon community.

4 References

[1] "Technical machine: A pokemon ai," http://doublewise.net/pokemon/, accessed: 2017-01-25.

[2] "Pokemon showdown ai bot," http://www.smogon.com/forums/threads/pokemon-showdown-ai-bot.3547689/, accessed: 2017-01-25.

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[4] "Gocnn," https://github.com/jmgilmer/GoCNN, accessed: 2017-01-25.

[5] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[6] "Pokemon showdown," http://pokemonshowdown.com/, accessed: 2017-01-25.