Playing Pokemon with deep RL

Noah Feinberg

Problem

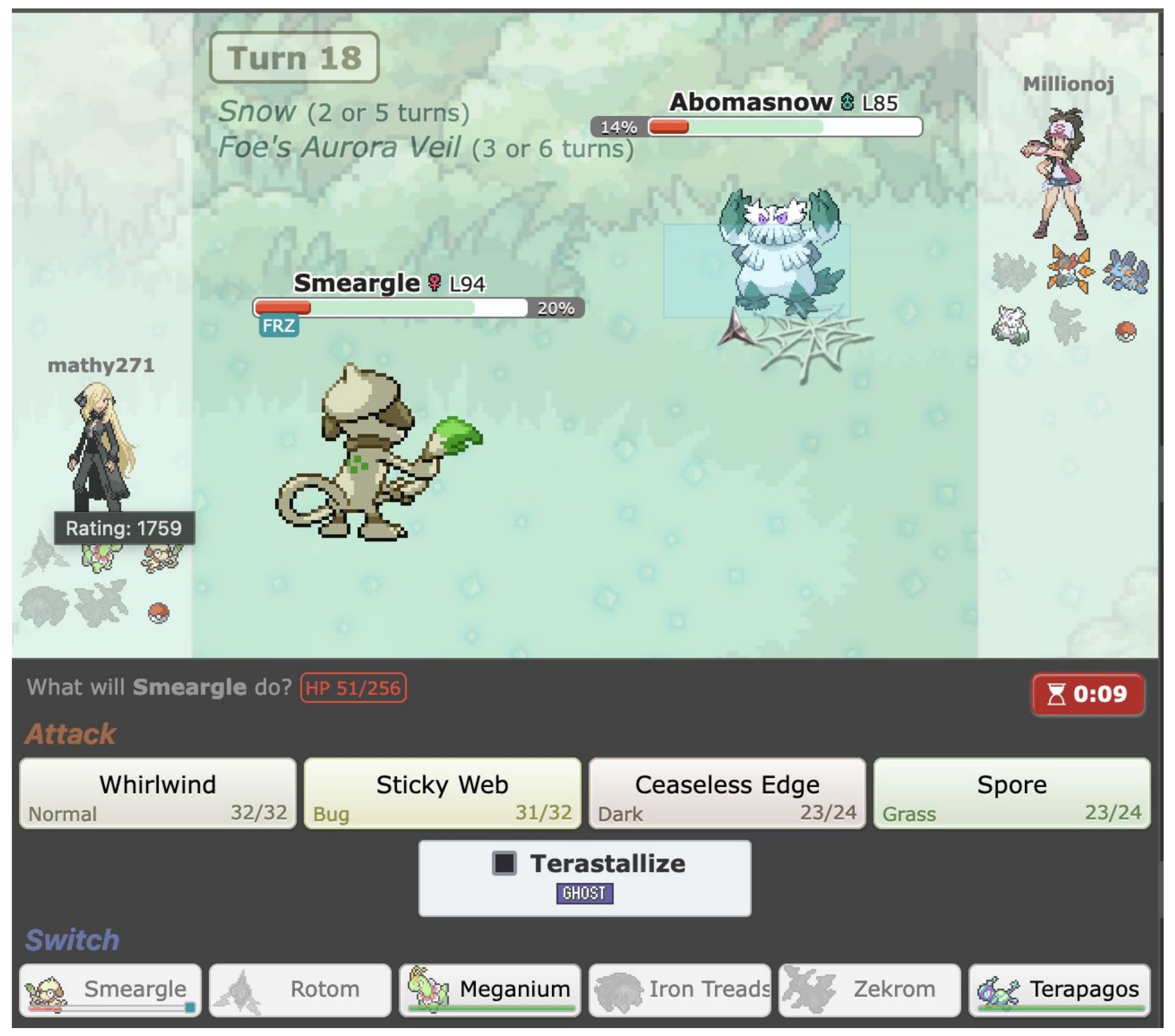
Pokemon battles are a complex and interesting domain that hasn't been well explored with RL. Notable difficulties include

- Simultaneous actions
- Dynamic action space
- Large and complex action space (934 moves, 1025 pokemon as of generation IX)
- Imperfect information, extends to opponents possible actions
- Large, partially observed state space (movesets, EVS, IVS, abilities, items, nature, and more)
- Highly stochastic transitions
- Sparse rewards, win/loss at end of episode
- Team building

Goals

Make progress towards an RL powered agent that plays generation IIX random battles (the library I used currently does not support generation IX). Random battles simplify the problem in several ways

- Reduces variances over initial states significantly
- No EVS, reduced movesets for all pokemon
- Reduces total number of possible pokemon and moves to ~450 and ~300 respectively
- No team building required



Example state. Our possible actions are the moves Whirlwind, Sticky Web, Ceaseless Edge, and Spore, or switching to Meganium or Terapagos. We cannot switch to Rotom, Iron Treads, or Zekrom since they have fainted. Both sides have one unrevealed pokemon indicated by the pokeball. We can also see other components of the state such as weather (Snow), side conditions Aurora Veil, Spikes, and Stick Web, and statuses (Frozen)

Methodology

The python library *poke-env* [4] was used to provide a gym like API for sending instructions to a *Pokemon Showdown* [5]. The architecture primarily followed R2D2 [1] with modifications, most notably a modified Bellman update for Markov Games [2] [3]. This meant the algorithm used an RNN function approximator as a Q-function to try and represent memory.

State and Action Embeddings

Moves were embedded in a 69 dimensional vector including the power, type, and various flags about the behaviour.

Pokemon were embedded as a 438 dimensional vector including moves, type, stats, boosts, and status.

Actions then could be embedded as a 517 dimensional vector.

States were embedded into a 4183 dimensional vector that included information about field information, reserve pokemon, turn, and number of fainted pokemon. This was fed into a 2 layer, fully connected network to reduce the size before the RNN

Action Masking

To deal with dynamic action sets, the agent maintained a list of the revealed pokemon and moves, adding an additional UNKNOWN action for when not all moves or pokemon had been revealed. Then, each pair of actions could be fed into state and advantage networks [6]. From there a nash equilibria was computed to determine a distribution to pick an action from.

Future Steps

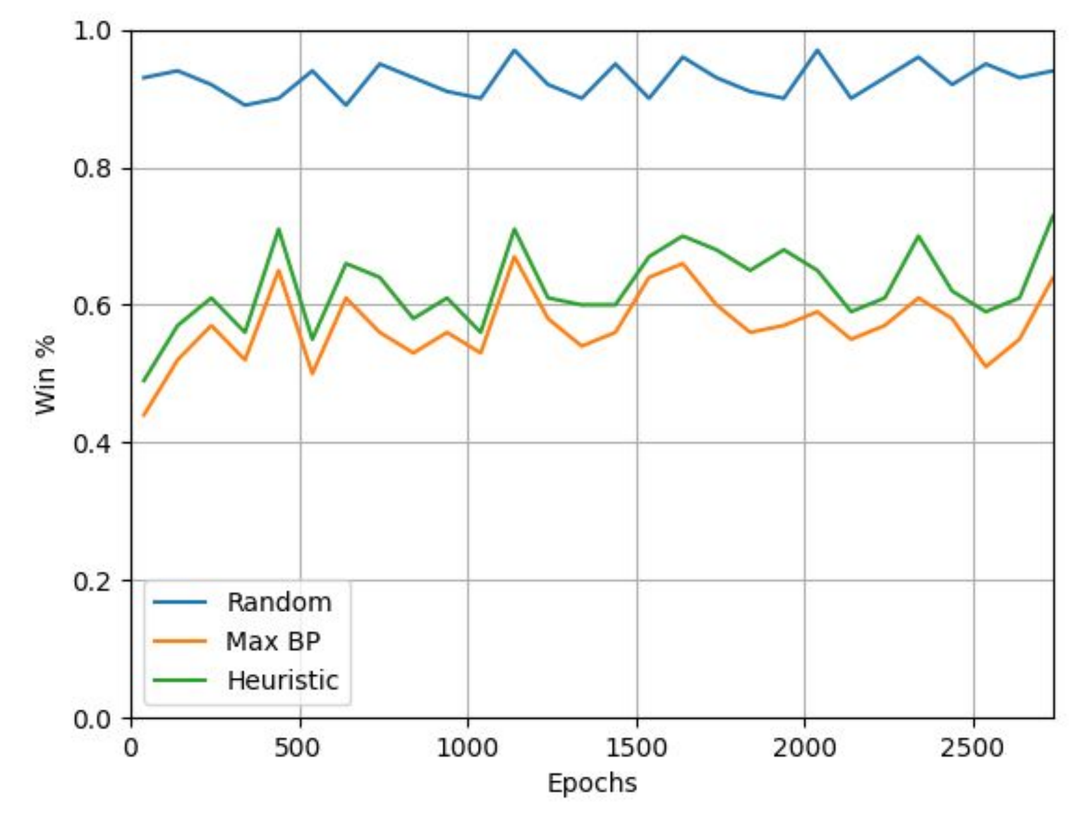
- Throw more compute power. We were only able to do about ~4000 training steps due to single threaded code. An obvious improvement is to use the distributed architecture of R2D2
- Learning better encodings with some kind of self-supervised reinforcement learning. The current encodings left our information about abilities and iems.
- Use strategies for better exploration, i.e. *Never Give Up* [7]

Results

To be honest the results are a bit… disappointing. Performance was benchmarked against several simple agents

- Random Player
- Max BP player
- GOFAI player with handwritten eval

While we were able to beat all of these, the margins were embarrassingly small, and performance plateaued very quickly. My suspicion is the state embedding and model complexity are holding it back.



References

- [1] Steven Kapturowski et al. "Recurrent experience replay in distributed reinforcement learning". In: I*nternational conference on learning representations*. 2018.
- [2] Littman, Michael L.. "Markov Games as a Framework for Multi-Agent Reinforcement Learning." *International Conference on Machine Learning*. 1994
- [3] Diddigi, Raghuram Bharadwaj, Chandramouli Kamanchi, and Shalabh Bhatnagar. "A generalized minimax Q-learning algorithm for two-player zero-sum stochastic games." *IEEE Transactions on Automatic Control* 67.9. 2022: 4816-4823.
- [4] Sahovic, H. (n.d.). *Poke-env: pokemon AI in python*. Retrieved from https://github.com/hsahovic/poke-env
- [5] Luo, Guancong et al. *Pokemon Showdown.* Retrieved from https://github.com/smogon/pokemon-showdown
- [6] Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., & de Freitas, N. (2016). Dueling Network Architectures for Deep Reinforcement Learning. *arXiv [Cs.LG]*. Retrieved from http://arxiv.org/abs/1511.06581
- [7] Badia, A. P., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., … Blundell, C. (2020). Never Give Up: Learning Directed Exploration Strategies. *arXiv [Cs.LG]*. Retrieved from http://arxiv.org/abs/2002.06038