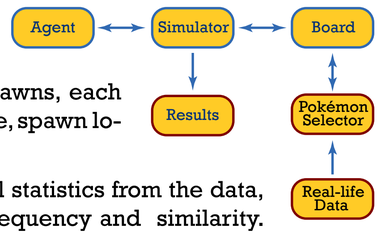
Catch'em all: A Pokémon Go Playing AI

Yue Li yulelee@stanford.edu

Motivation

Pokémon Go is a location-based augmented reality game launched this summer. In the game, players could use a mobile device to capture and battle Pokémons in the real world[1]. The Pokémons would appear on the screen only if they were in the same real-world location as the player. Currently, there is no effective way for the players to locate the *exact* positions of Pokémons[2]. While it's really exciting to explore the outside (randomly), it's won't hurt to learn and accumulate experience and try to find Pokémons more effectively. In this project, I developed an AI agent that does exactly this, using the same information as a human player, and try to catch as many Pokémons as it can.

Simulation



The simulation of this game has the following components:

Data contains 300k individual spawns, each spawn includes the Pokémon name, spawn location and spawn time.

Pokémon Selector extracts useful statistics from the data, construct a model of Pokémon frequency and similarity. When being asked for Pokémons, it chooses an initial spawn according to the frequency and (potentially) some neighboring spawns based on the similarity model.

Board: The game is simulated by a *n* * *n* board, which keeps track of the positions of Pokémons and the agent, the agent can catch any Pokémon if they're at the same location[3]. The board is also responsible for generating the radar information for the agent, just like in the real game, we'll let the agent know which Pokémons are nearby (without revealing the exact location, or even the distance). Each Pokémon can only exist on the board for a random amount of time, once the time is up, the board should delete it.

Agent knows its won position and the radar information provided by the board (a list of Pokémon names), and decides where to go next. One step towards one direction at a time.

Simulator is like the middle man between the board and agent, passing messages in between, and at the same time, recording all of the useful statistics needed for the evaluation.

Input / Output

No special **input** is needed, the game could just start (agent being put to the center of the board). The **output** is the percentage of Pokémons being caught by the agent, compared with the total amount of Pokémons being spawned onto the board.

Agents

Several agents utilizing different techniques have been developed:

Oracle Agent:
The oracle agent knows the exact locations of all the Pokémons. It constructs a graph connecting them. An edge from one Pokémon to another exist if it is guaranteed that the agent can catch the first Pokémons and then go to the next before it disappears. The graph is acyclic so the longest path can be found by dynamic programming.

Baseline Agent:
Here we view the world of Pokémons as a MDP with these definitions:

- State: a tuple containing the current agent position and a list of Pokémon names on the radar. Start state is the middle point on the board and an empty radar list.
- Actions: we have 4 actions: 'LEFT', 'RIGHT', 'UP' and 'DOWN'.
- Reward: issued by the board when the agent catches any Pokémon.
- Transitions: the transition for the agent position is deterministic, just follows the action, while the radar list is updated by the board to reflect the situation around the agent, which is random.

Here we utilize the Q-learning algorithm with function approximation. The feature extractor constructs indicator features for the current agent position and the Pokémons in the radar.

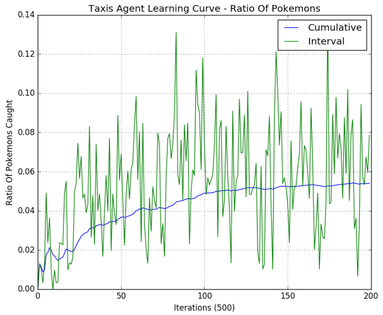
Improved Agent:
The improved agent is mostly the same as the baseline agent, except that it keeps track of the last radar list it has seen, and compare it with the current one, if it managed to keep some Pokémons within the list (tracking...), it can issue some reward to itself. By doing this, we are encouraging seemingly-good moves.

Reflect Agent:
Instead of viewing the world as an MDP, the reflect agent tries to find the Pokémons based on the memory. It keeps a list of 'centers' for each Pokémon, which is set to the average location of all the positions where that Pokémon has been caught. When a Pokémon appears on the radar, the agent heads to the corresponding center.

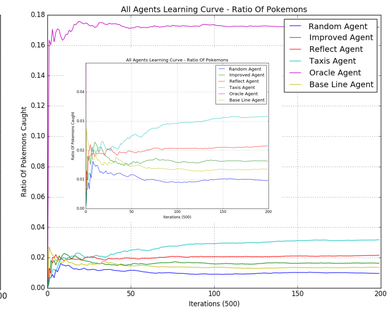
Taxis Agent[4]:
The taxis agent tries to estimate the probability of catching each Pokémon at every position. The intuition is that when we see a Pokémons on the radar, though we don't know the exact location, we can still increase the probability of that Pokémon exists at all of the nearby locations. Similarly, if we have actually caught a Pokémon at some location, the probability of that Pokémon appears again at here, and all of the nearby places, has increased. When making decisions, for each position, the agent sums the probability of all the Pokémons in the radar, renormalize among all the positions, and then randomly sample a destination using the normalized probabilities. The direction could then be determined by that destination.

Results

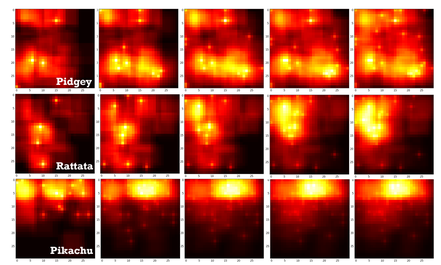
Currently the taxis agent has the best performance among the others, here are some of the results:



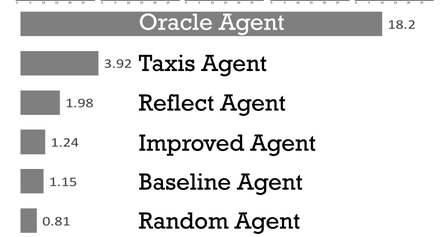
(above) The learning curve for the taxis agent. Data are collected for each interval (500 iterations), the cumulative catching rate and the catching rate within the interval are presented.



(above) The learning curve for all of the agents for one simulation. The small figure in the middle is a zoomed-in version of the same plot for all of the agents except the oracle.



(left) The probability heatmap for 3 Pokémons plotted by the taxis agent within the same simulation. Each spot represent one position on the board, the lighter the color, the higher the probability. The plots are generated for each 10000 iterations.



Catching rate (%) for all the agents. Generated by 10 independent simulations each with 100000 iterations and then take the average for each agent.

Next

There are still a few problems for the taxis agent. Firstly, it's too slow: there is too much computation for each iteration. Second, modeling each point seperately is also not so scalable. Those things could be the focus of the next few weeks.

[1] https://en.wikipedia.org/wiki/Pokémon_Go
[2] Well... this is complicated, here we ignore the SF tracker currently provided by the game and all of the other 3rd-party trackers.
[3] Let's assume we have an infinite amount of Pokéballs and Pokémons don't run away.
[4] Taxis is the movement of an organism in response to a stimulus such as light or food[5].
[5] https://en.wikipedia.org/wiki/Taxis