Evolution strategies cho cuộc thi GoldMiner

Cuộc thi: RLCOMP - https://rlcomp.codelearn.io/

Đội: BeerBot

Xin chào mọi người, mình là Hoàng Hồng Quân, thành viên đội BeerBot. Mình là sinh viên kinh tế, sau quãng thời gian đi làm sales thấy không hợp thì 2 năm trước mình có vào TP Hồ Chí Minh học master tại viện JVN (John von Neumann). Tuy model chưa đủ tốt để có thể tiến sâu tại cuộc thi, mình cũng xin phép chia sẻ lại giải pháp của team mình, hy vọng mọi người có thể tìm thấy một vài điều có ích có thể dùng được cho những cuộc thi RL sau.

1. Hướng tiếp cận chung của team mình:

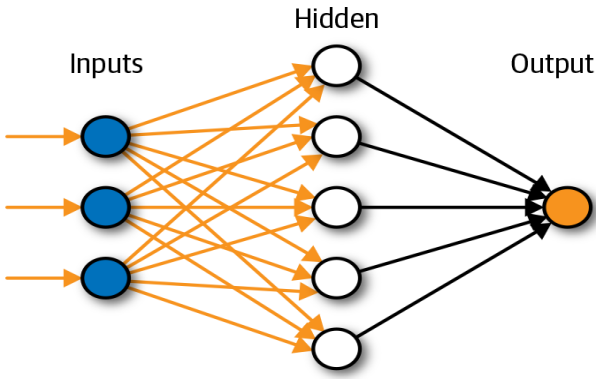
- Dùng thuật toán để xác định đường đi ngắn nhất và tốn ít năng lượng nhất từ điểm A đến B trên map. Giải thuật cũng khá đơn giản, các bạn có thể tham khảo 1 bài tương tự trên Leetcode (https://leetcode.com/problems/minimum-path-sum)

- Tại mỗi step, mình sẽ dùng ES (Evolution strategies) để xác định mỏ vàng cần đến, sau đó dùng thuật toán ở trên để đi đến mỏ vàng.

2. Thuật toán Evolution strategies cho cuộc thi:

Evolution strategies là 1 thuật toán optimization có thể thay thế các thuật toán RL trong 1 số trường hợp. Nó có thể là 1 lựa chon tốt khi số lượng timesteps trong 1 episode là nhiều, khi mà actions tạo ra "longlasting effects", hoặc khi value functions khó có thể estimate. (https://openai.com/blog/evolution-strategies/)

Để xác định mỏ vàng tốt nhất tại mỗi step, team mình xây dựng 1 features vector có length = 9 tương ứng cho từng mỏ vàng.



Features vector này là input cho 1 mạng neuron network đơn giản có 1 lớp hidden layer với số neurons là 18 và 1 output.

Các output của các mỏ vàng sẽ được so sánh, mỏ vàng có output lớn nhất sẽ được chọn.

Việc học weights của mạng NN sẽ do ES xử lí. Tóm tắt lại quá trình như sau:

B0: Ở iteration 0: Thuật toán ES sẽ tạo ra 1 population gồm n (team mình chọn n = 100) mạng NN

B1: Tại step 0 của 1 ván đấu: random weights cho 100 mạng NN.

B2: Tại mỗi step, cho từng features vector của các mỏ vàng làm input cho mạng NN, chọn ra mỏ vàng có output lớn nhất để đi đến.

B3: Sau 100 steps, tính reward = số vàng đào được của agent

B4: ES sẽ chọn trong 100 mạng NN những mạng NN có reward tốt nhất để tiến hành tạo ra population cho các iteration tiếp theo, các mạng NN sẽ được update weights. Sau đó lặp lại bước 2, 3, 4 ở các iter sau.

Các bạn có thể tham khảo thêm về ES tại https://lilianweng.github.io/lil-log/2019/09/05/evolution-strategies.html

3. Featues vector cho từng mỏ vàng:

Độ dài vector là 9:

4 khoảng cách từ mỏ vàng tới 4 agents (khoảng cách được estimate = số steps cần để agent đi đến mỏ, dựa trên khoảng cách manhattan và tổng số energy của quãng đường)

1 khoảng cách trung bình từ mỏ vàng đến các mỏ vàng khác

1 = mean của (vàng tại các mỏ khác / khoảng cách từ mỏ đến các mỏ khác)

1: số vàng của mỏ

1: vàng của mỏ/ tổng vàng hiện có

1: số mỏ vàng hiện có

4. Một vài ý kiến đánh giá về phương pháp:

Về bots để train: team mình có tạo ra 15 con bots (1 số dùng rules, 1 số dùng ES với features đơn giản). Tại mỗi iter mình lấy ngẫu nhiên 3 bots để train.

Về reward: Như ở trên thì reward chính là số vàng đào được sau 100 steps, tuy nhiên reward này khá biến động gây khó khăn cho thuật toán ES, nên team mình sử dụng reward là số vàng trung bình sau 12 ván chơi.

Về thuật toán: Mình có sử dụng khá nhiều thuật toán ES khác nhau (NES, CMAES, …) tuy nhiên thuật toán ES trong paper của openai cho kết quả tốt nhất trong bài toán này (https://openai.com/blog/evolution-strategies/)

ES có nhược điểm là cần rất nhiều data để train, và cần 1 CPU có nhiều cores để có thể scale, nếu không thời gian train sẽ rất lâu. Để khắc phục vấn đề đó nên team mình mới xây dựng features vectors cho từng mỏ vàng như ở trên mà không đưa toàn bộ state vào và dùng CNN như cách thông thường. Việc làm này có lợi là sẽ giúp giảm thời gian train, model cũng đỡ bị overfit và có thể dùng cho map bất kì. Tuy nhiên nhược điểm là việc mất mát 1 số thông tin về vị trí tương ứng giữa các mỏ vàng. Có lẽ hoàn

hảo nhất là có thể biểu diễn các mỏ vàng dưới dạng graph để làm input, tuy nhiên mình chưa có đủ khả năng để xử lí.

Thời gian train: Mình train = CPU của Colab (Xeon 2 cores) mất khoảng 4 – 5 tiếng là model hội tụ. Do đặc tính dễ scaleup của ES, nếu bạn có 1 CPU nhiều cores 1 chút (16 cores chẳng hạn) thì hoàn toàn có thể train model < 60 phút. Mình có dùng chùa CPU của Colab =)) mở nhiều tabs và train ra khoảng 150 cái weights khác nhau, sau đó ensemble (dùng voting đơn giản) để chọn mỏ vàng tốt nhất.

Theo đánh giá cá nhân thì với vị trí xuất phát ở các góc của maps thì bot xử lí khác tốt, tuy nhiên nếu xuất phát tại vị trí trung tâm thì bot chưa được tốt lắm. Khá đáng tiếc là mình đã không chia vị trí xuất phát (ở góc hoặc ở trung tâm) để train riêng. Về phía features có lẽ thêm yếu tố vị trí của các bots tại 1 số steps trước có thể cải thiện thêm model.