Sprawozdanie

Zastosowanie uczenia maszynowego do gry wyścigowej

Mateusz Burczaniuk,
Patryk Fijałkowski

07.12.2020

Spis treści

1 Wprowadzenie

1.1 Problem

Omawiana gra wyścigowa symuluje rywalizację na torze pomiędzy kilkoma samochodami wyścigowymi. Tor, na którym odbywa się rywalizacja jest zamkniętą pętlą otoczoną niezbyt wysokimi ścianami. Poza tym na torze występują także przeszkody. Samochód może poruszać się do przodu, do tyłu, skręcać w lewo, prawo lub wykorzystać nitro. Zderzenie samochodu gracza ze ścianą lub przeszkodą powoduje natychmiastową śmierć gracza i koniec gry. Zderzenie dwóch graczy ze sobą powoduje analogiczne skutki (śmierć i usunięcie z rozgrywki) dla obu graczy.

Gracz ma zadanie przejechać zadaną trasę jak najszybciej, wyprzedzając swoich rywali. Podczas rywalizacji gracz może znacznie zwiększyć swoją prędkość przy użyciu nitro. Jednak maksymalna dostępna w danej chwili ilość nitro jest ograniczona, więc gracz nie może używać go bez przerwy. Zużyte nitro stopniowo odnawia się i po pewnym czasie wraca do pierwotnego poziomu. Może ono odnawiać się nieskończoną liczbę razy.

Wszystkie rozważane pojazdy mają taki same parametry techniczne i osiągi, w związku z czym wygrana lub przegrana w wyścigu zależy wyłącznie od obranego przez nie sposobu przemieszczania się po torze.

Zatem postawiony problem składa się z kilku zagadnień - należy zoptymalizować politykę doboru kierunku poruszania się gracza oraz używania nitro tak, aby przejechać całą trasę bez żadnego zderzenia, a jednocześnie jak najszybciej, pozostawiając rywali za sobą. Zadania te w pewnym stopniu konfliktują ze sobą, gdyż optymalną strategią w celu uniknięcia zderzeń jest stanie w miejscu, natomiast dążenie do uzyskania optymalnego czasu przejazdu powoduje wzrost ryzyka zderzenia z przeszkodą lub innym pojazdem.

Problem próbowano rozwiązać za pomocą dwóch metod: PPO oraz SAC.

1.2 PPO

Proximal Policy Optimization, czyli PPO to nowa metoda, wprowadzona w 2017 roku. Wykorzystuje ona metodę aktor- krytyk, posługując się dwoma sieciami neuronowymi. Aktor jest reprezentowany przez sieć neuronową, która wykonuje zadanie klasyfikacji, wybierając, jaką akcję podjąć przy obserwowanym stanie środowiska. Krytyk to druga sieć, o podobnej strukturze, która uczy się oceniać, czy poszczególne akcje poprawiają, czy pogarszają stan. Krytyk zwraca Q-wartość akcji wybranej w poprzednim stanie. Na tej podstawie aktor modyfikuje swoją politykę, dbając, by nie przeprowadzić

zbyt dużych zmian.

W metodzie PPO aktualna polityka wykorzystywana jest do przeprowadzenia pewnej liczby doświadczeń (wielkość tę określa się mianem rozmiaru batcha) wymagających interakcji ze środowiskiem, które wykorzystywane są do poprawy polityki. Jest to uczenie on-policy, jako, że zebrane dane są wykorzystywane tylko raz.

Algorytm korzysta z mechanizmu Uogólnionej Estymacji Przewagi (ang. *Generalized Average Estimation*), aby ustalić, które akcje przyniosły, jakie skutki. Liczona jest ona od ostatnich wykonywanych akcji. Obliczane są kolejno wartości:

$$\delta = r_t + \gamma V(s_{t+1}) m_t - V(s_t) \tag{1}$$

$$gae_t = \delta + \gamma \lambda m_t gae_{t+1} \tag{2}$$

$$R_t(s_t, a_t) = gae_t + V(s_t) \tag{3}$$

gdzie:

- t - numer kroku analizowanego testu,
- s_t - stan w chwili t,
- a_t - akcja podejmowana w chwili t,
- $V(s)$ - wartość stanu s,
- r_t - nagroda w chwili t.
- m_t - maska w chwili t, jeśli w stan t jest końcowy, to 0, w przeciwnym wypadku 1.

Wystąpiły tutaj dwa hiperparametry algorytmu:

- γ - parametr dyskontowy, standardowo 0.99,
- λ - parametr wygładzający, standardowo 0.95.

Wyliczane jest ratio, które mówi, jak bardzo polityka uległa zmianie.

$$ratio = \pi_{new} / \pi_{old} \tag{4}$$

Polityka jest oceniana przez aktora i krytyka. Ocena krytyka to błąd średniokwadratowy pomiędzy wartością zwracaną a oszacowaniem krytyka.

$$critic\_loss = (R - V(s))^2 \tag{5}$$

Natomiast błąd aktora wyraża się wzorem:

$$actor\_loss = min(ratio * advantage, clip(ratio, 1-\varepsilon, 1+\varepsilon) * advantage) \tag{6}$$

ε to parametr obcinania, który zapewnia, że jednorazowo zmienimy politykę o co najwyżej ε%. Domyślna wartość tego parametru wynosi 0.2.

Ostatecznie ocena polityki wyraża się wzorem:

$$total\_loss = critic\_loss * critic\_discount + actor\_loss - entropy \quad (7)$$

critic_discount to współczynnik mający na celu zrównoważenie wpływu na wynik oceny aktora i oceny krytyka. Entropia to natomiast hiperparametr metody, który domyślnie przyjmuje wartość 0.005.

Na podstawie powyższych funkcji oceny używane sieci uczone są przy użyciu metody stochastycznego przyrostu gradientu. Sieć jest uczona na danych z pojedynczego batcha przez pewną liczbę epok, będącą hiperparametrem metody.

Opis działania metody PPO został oparty na artykułach C. Trivediego [1],[2].

1.3 SAC

W przypadku obu algorytmów zbadane zostaną rozmiar batcha, parametr tempa uczenia sieci neuronowych oraz liczba ich warstw.

2 Eksperymenty

3 Wnioski

Literatura

[1] Trivedi C., *Proximal Policy Optimization Tutorial (Part 1/2: Actor-Critic Method)*, https://towardsdatascience.com/proximal-policy-optimization-tutorial-part-1-actor-critic-method-d53f9afffbf6, 2019.

[2] Trivedi C., *Proximal Policy Optimization Tutorial (Part 2/2: GAE and PPO loss)*, https://towardsdatascience.com/proximal-policy-optimization-tutorial-part-2-2-gae-and-ppo-loss-fe1b3c5549e8, 2019.