Trudności w sprzątaniu - zastosowanie algorytmu DDPG

Filip Morawiec [1] Wojciech Siostrzonek [1] Witold Beluch [2]

[1]III LO im. A. Mickiewicza w Katowicach [2](opiekun) Politechnika Śląska

Wstęp

Sprzątając pokój, lub organizując przesyłki w magazynie, możemy przyjąć różne strategie. Pierwszym podejściem mogłoby być zgromadzenie wszystkich przedmiotów w jakimś centralnym punkcie. Z drugiej strony możnaby ustalić kilka takich punktów, jeżeli układ ścian nie wskazuje jednoznacznie na żadne konkretne środkowe miejsce.

Celem tej pracy było rozstrzygnąć czy istnieje reguła wyznaczająca optymalne rozłożenie takich punktów - w kontekście sprzątania, który został tutaj przyjęty, będziemy nazywać je kopcami. Rozmiar pokoju wynosi 35x35 kratek.

Rodzaj problemu

Na samym początku ustaliliśmy dział uczenia maszynowego w którym będziemy pracować. Reinforcement learning *(ang. uczenie przez wzmocnienie)* obejmuje ogół problemów, w których biorą udział dwie jednostki - agent oraz środowisko.

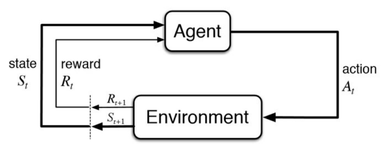

Figure 1. Schemat reinforcement learning.

Agent, na podstawie swoich poprzednich doświadczeń oraz obecnego stanu środowiska S_t działa na nie czynnością A_t, by dostać informację zwrotną o nagrodzie jaką dostał R_{t+1} j nastepujący stan S_{t+1}. Indeks dolny odnosi się do momentu w czasie.

Schemat projektu

Ostateczny program składa się z dwóch modułów:

- **Środowiska**, które m.in. na podstawie współrzędnych początkowych i końcowych ruchu miotłą - zatem $A_t = (x_1, y_1, x_2, y_2)$ - zwraca S_{t+1} oraz ilość brudu który wpadł do kopca R_{t+1}.
- **Agenta sprzątania**, który uczy się optymalnie sprzątać działając w środowisku.

Najpierw zostało zaimplementowane środowisko, następnie agenci sprzątania w 3 wersjach odpowiadających trzem różnym algorytmom uczenia przez wzmocnienie.

Repozytorium projektu widnieje na github.com/gournge/cleaning-optimization. Warto zaznaczyć że w pliku konfiguracyjnym `config.cfg` są ustalone ważniejsze parametry symulacji, takie jak szerokość miotły, czy ilość kopców. Oprócz tego wyjaśnione zostały ważniejsze kwestie implementacji.

Środowisko

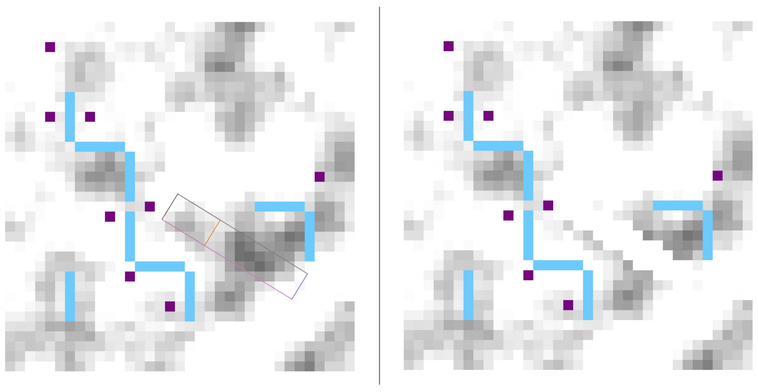

Figure 2. Ruch miotłą bez punktu P. Losowo ustalone fioletowe kopce, niebieskie ściany.

W momencie, w którym środowisko otrzymuje czynność A_t, wykonywane są następujące czynności:

1. utworzony jest prostokąt reprezentujący obszar działania miotły,
2. w prostokącie wyznaczamy punkt P gdzie podczas ruchu miotłą zostanie przekroczona jej pojemność,
3. losowo rozrzucamy brud znajdujący się w pozostałej części głównego prostokąta:
   1. jeżeli P istnieje, rozrzucamy brud na boki i do przodu,
   2. w przeciwnym razie - skoro brud się nie wysypał - przemieszczamy go tylko do przodu.
4. zwracamy ile brudu wpadło do kopców i czy ruch miotłą nie kolidował ze ścianą.

Wypróbowane algorytmy

Z poniższych jedynie ostatni okazał się być właściwy - wiąże się to z rodzajem zwracanych przez agenta czynności. W przypadku A2C i DQL czynność A_t jest wybierana z dyskretnej przestrzeni (np. przyciski konsoli Atari), wówczas DDPG deterministycznie wybiera ją z przestrzeni ciągłej (np. 4 liczby w przedziale $(0, 35)$).

Actor-critic (A2C) i Deep Q-learning (DQL)

Auto-critic składa się z dwóch głównych komponentów: aktora i krytyka. Aktor próbuje wybrać najlepsze akcje na podstawie obserwowanego środowiska, podczas gdy krytyk ocenia te akcje i dostarcza informację zwrotną o ich jakości. Deep Q-learning to głęboko-neuronowa wersja popularnego algorytmu, który opiera się na znalezieniu najlepszego zaokraglenia funkcji Q - funkcji, która zwraca ewaluację możliwych czynności A_{t+1}.

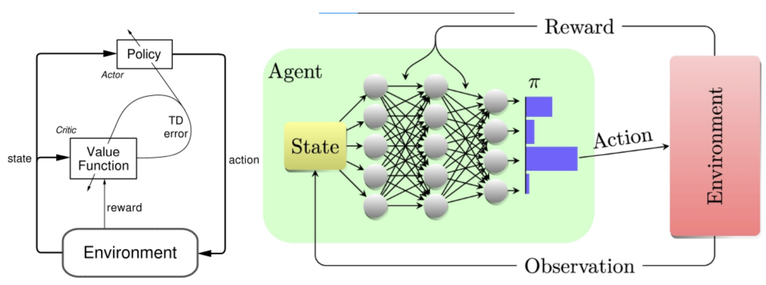

Figure 3. Schemat A2C i DQL.

Deep Deterministic Policy Gradient (DDPG)

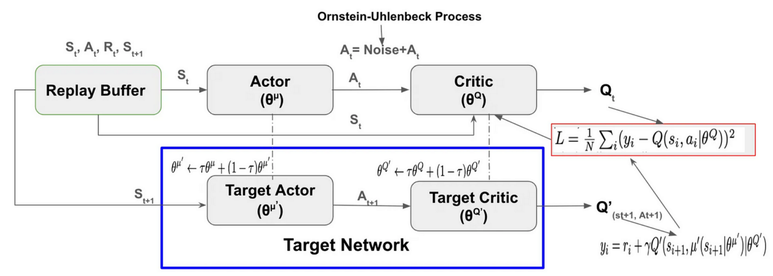

Figure 4. Schemat DDPG.

- **memory replay**; w trakcie treningu losowo zapisywane i wybierane są elementy, by zróżnicować proces aktualizacji wag sieci neuronowych,
- **target networks**; osobne kopie sieci aktora i krytyka - bez nich cel, do którego zmierzają sieci, zmienia się z każdą próbą zbliżenia do niego tych sieci,
-

Architektura sieci

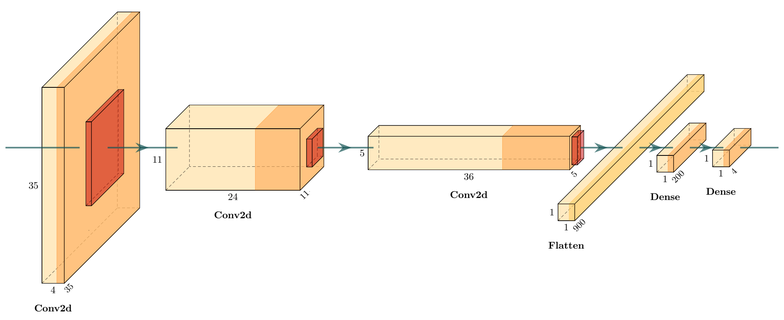

Figure 5. Architektura sieci (kształt opiera się na [1]) aktora. Rozmiar wejścia 3x35x35, rozmiar wyjścia 4.

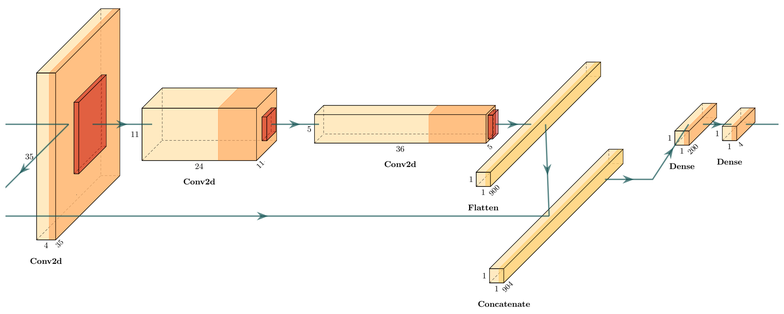

Figure 6. Architektura sieci krytyka. Rozmiar wejścia 3x35x35 i 4, rozmiar wyjścia 4.

Aktor podejmując decyzję na podstawie obserwacji przyjmuje środowisko jako tensor o wymiarach 3x35x35 - są to 3 warstwy kwadratowego pokoju, z czego:

- pierwsza odpowiada za układ ścian,
- druga za układ kopców,
- trzecia za gęstość brudu.

Krytyk oprócz tego bierze na argument czynność zwróconą przez agenta - 4 liczby (stąd rozmiar danych 1x1x904 w przedostatniej warstwie). Obydwa rodzaje tych sieci zwracają również 4 liczby - aproksymowane przez nich optymalne A_t.

Warto nadmienić że wybór przedostatniej warstwy jako wejścia dla czynności w sieci krytyka jest uzasadniony trudnością połączenia kształtu czynności z poprzednimi warstwami konwolucyjnymi.=

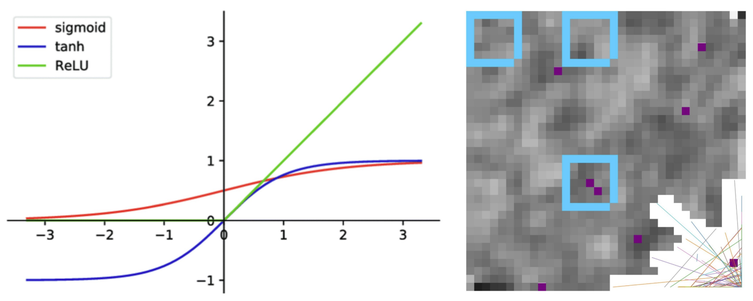

Figure 7. (lewo) wykresy badanych funkcji aktywacyjnych (prawo) ruchy zwracane przez wytrenowanego agenta z funkcją ReLU

Podejrzewając niesymetryczny (zatem potencjalnie faworyzujący prawy dolny róg) kształt wykresu funkcji ReLU eksperymenty zostały przeprowadzone jedynie z funkcjami sigmoid oraz tanh. Ta druga została przetestowana najwięcej razy, ze względu na to, że tanh jest funkcją nieparzystą, czyli nieco skupioną na środku pokoju.

Pierwsza faza treningu

Pierwotnie, by uniknąć nawyku uderzania miotłą w ścianę, ustaliliśmy, że gdy ruch miotłą będzie niewłaściwy, środowisko zwróci $R_t = K < 0$. Zazwyczaj dla losowych ruchów sumaryczna nagroda (całkowita ilość sprzątniętego brudu) wynosiła ≈ 1.5. Biorąc $K = -5$, otrzymaliśmy następujący wykres:

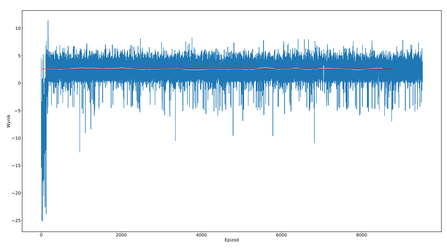

Figure 8. zależność między wynikiem agenta sprzątania od ilości epizodów (seria 40 ruchów w jednym pokoju)

Anomalia

Próbując dostosować parametry treningu i środowiska zostały przeprowadzone różne eksperymenty. Są one opisane w pliku `experiments.txt`. Żaden z nich niestety nie doprowadził do dostatecznie dobrego poziomu wytrenowania agenta. Istotnym okazał się eksperyment 13: w trakcie treningu wariancja wyników drastycznie się zwiększyła:

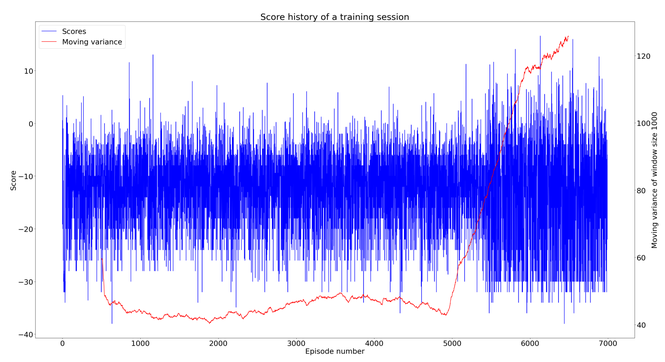

Figure 9. Wyniki agenta (niebieski) i ich wariancja obliczana z 1000 najbliższych wyników (czerwony). Parametry: $K = -2$, `actor-learning-rate = 0.0001`, `replay-memory-size = 1024`, `noise = 0.3`

Wizualizacja treningu

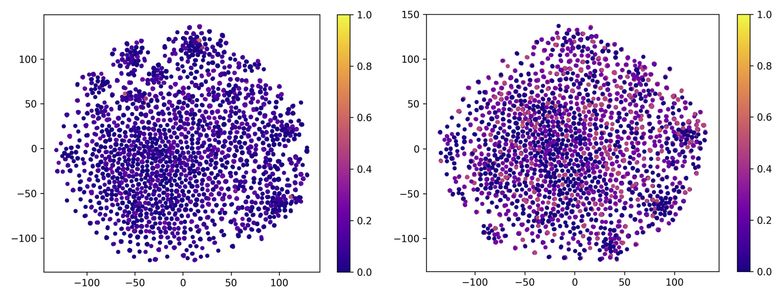

Figure 10. Każdy punkt to środowisko po przejściu przez pierwsze 4 warstwy sieci agenta niewytrenowanego (lewo) i wytrenowanego (prawo), pokolorowany zgodnie z jego nagrodą jaką otrzymał agent w tej sytuacji

Na wykresach widać jedynie różnice w wariancji kolorów - wytrenowany agent osiąga nieco wyższe wyniki. Występowanie podobnie pokolorowanych grup punktów wynika ze zbliżonych wyników w przypadkach środowisk, które głęboko w swoim *mózgu* kojarzy ze sobą w podobny sposób. Świadczy to o pewnej stabilności tego algorytmu, lecz trudno wyciągnąć bardziej złożone wnioski, ponieważ algorytm *t-SNE* [5], który został użyty do reprezentacji wielowymiarowych ($\in \mathbf{R}^{200}$) danych z przedostatniej warstwy na płaszczyźnie, jest do pewnego stopnia losowy: regularności w rozłożeniu punktów mogą nie być miarodajne.

Konkluzja

Chociaż cel pracy - wytrenowanie optymalnych agentów sprzątających, a następnie (potencjalnie) planowanie rozłożenia kopców - nie został osiągnięty, zostały dokonane istotne obserwacje, które oprócz wskazania dalszych kierunków badań, dostarczają innego wglądu do działania algorytmu DDPG. Chociaż mogłoby się wydawać, że obserwacje sugerują nieaplikowalność DDPG do tego konkretnego problemu, naprzeciw stają rezultaty z [2], gdzie po dobraniu sieci konwolucyjnej o odpowiednim kształcie udało wytrenować się wirtualnego kierowcę skutecznie podejmującego decyzje na podstawie wielowymiarowego obrazu. Z tego względu autorzy podejrzewają, że wybrana architektura sieci mogła być problemem. Zamiast dołączać dane o wykonanej czynności do pierwszej gęstej warstwy w architekturze krytyka, można rozważyć dołączenie jej dopiero w przedostatniej warstwie - niewykluczone, że informacje o tej czynności A_t zostają w pewien sposób utracone, kiedy łączymy je z aż 900 wartościami odnoszących się do obserwacji środowiska.

Oprócz tego metoda wykorzystania algorytmu t-SNE do wizualizacji reprezentacji środowiska przez DDPG (zainspirowana [4]) wydaje nie być się do końca odpowiednia. Może to wynikać z niewykształcenia agenta - gdyby faktycznie umiał on sprzątać, wykres przedstawiałby więcej informacji.

Źródła

[1] Zhang J. Humaidi A.J. Alzubaidi, L.
Review of deep learning: concepts, cnn architectures, challenges, applications, future directions.
J Big Data 8, 53, 2021.

[2] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez.
Deep reinforcement learning for autonomous driving: A survey.
IEEE Transactions on Intelligent Transportation Systems, 23(6):4909–4926, 2022.

[3] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra.
Continuous control with deep reinforcement learning.
In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis.
Human-level control through deep reinforcement learning.
Nature, 518(7540):529–533, February 2015.

[5] Laurens van der Maaten and Geoffrey Hinton.
Viualizing data using t-sne.
Journal of Machine Learning Research, 9:2579–2605, 11 2008.