Warsztaty z technik uczenia maszynowego - Halite

Wojciech Drochomirecki
Maciej Ejduk
Aleksandra Gajda

Marzec 2022

1 Temat

Celem projektu będzie stworzenie algorytmu, który będzie znajdował optymalną strategię ruchów dla gry Halite. Opis gry na Kaggle pod linkiem: Kaggle. Opis Halite na oficjalnej stronie twórcy: Halite. Jest to projekt, który dzięki dużemu wachlarzowi problemów które zawiera pozwoli na zastosowanie różnych algorytmów i wykorzystanie jak największej ilości informacji zdobytych na przedmiocie. Jednocześnie dzięki swojej postaci pozwala na stworzenie rozwiązania zarówno banalnie prostego jak i wysoce wysublimowanego dzięki czemu będziemy mogli dostosować poziom zaawansowania do możliwości w trakcie semestru.

2 Zadania w zespole

Sama gra udostępnia dosyć rozbudowany interfejs pozwalający na komunikację z jej środowiskiem, wizualizację rezultatów jak i zbieranie danych z wielu symulacji. Aby efektywnie rozpocząć pracę związaną z implementacją algorytmów ML w naszym rozwiązaniu potrzebujemy dobrze zapoznać się z tym jak on działa oraz poznać jego funkcje. W tym celu każdy wykona swoją wersję podstawowego algorytmu do prowadzenia rozgrywki wykorzystującą proste algorytmiczne podejście do problemu.

Następnie rozwiązania będą połączone, lub wybrane zostanie najlepsze z nich które poóźniej stanie się podstawą do dalszych prac nad projektem.

Następnym krokiem będzie wdrożenie różnych algorytmów ML w konkretnych decyzjach podejmownych przez bota. Gra udostępnia rozmaite przestrzenie, w których mogą one zostać wykorzystane. Z pośród już rozpoznanych przez nas dziedzin można wyróżnić takie jak sterowanie statkami, interakcje z nieprzyjacielem oraz makrostrategie takie jak tworzenie nowych statków czy gospodarowanie zasobami. W każdej z tych dziedzin planujemy przetestować różne podejścia oparte na wstępnym reaserchu dostępnych rozwiązań. Dodatkowo podczas przeprowadzonych eksperymentów, planujemy zbierać przykładowe rozgrywki ilustrujące typowe dla danego problemu i rozwiązania cechy, które pozwolą na końcu projektu wysnuć wnioski na temat poszczególnych algorytmów.

W tym etapie planujemy przydzielić każdemu z członków zespołu jedną z dziedzin w której rozwinie algorytm ML spełniający którąś z wymienionych wyżej funkcji, lub inną która okaże się bardziej interesująca dla wyników eksperymentu. Dzięki temu członowie zespołu otrzymają pewnego rodzaju niezależność tworząc wspólnie ostateczne rozwiązanie.

Dodatkowo wspólnie będziemy tworzyć dokumentacje projektu związaną z odpowiadającymi nam częściami projektu.

3 Narzędzia

Narzędzia których będziemy używać w rozwoju naszej aplikacji będą głównie bibliotekami Pythona. Najprawdopodobniej będą to rozwiązania należące do podstawowego kanonu narzędzi wykorzystywanych w zadaniach z zakresu Data Science, takie jak pyTorch, numPy, matplotlib, tensorflow. Dzięki ich powszechnej dostępności i szerokim zastosowaniom będziemy mieli dostęp do rozbudowanych dokumentacji oraz tutoriali pomagających w zrozumieniu sposobu ich działania jak i metod implementacji konkretnych algorytmów. Są to też doskonałe narzędzie pomagajace w wizualizacji otrzymanych wyników która pomogoże w wyciągnięciu konstruktywnych wniosków z eksperymentu. Do tworzenia wizualnych reprezentacji naszych rozgrywek jak i do interakcji ze światem gry wykorzystamy środowisko udostępnione przez twórców

gry. Dzięki temu zredukujemy ilość pracy niezwiązanej bezpośrednio z przedmiotem polegającej na samodzielnej implementacji środowiska gry.

3.1 Język programowania

Projekt planujemy wykonać przy użyciu języka *Python*. Jest on, obok *R*, narzędziem powszechnie używanym we wszystkich dziedzinach Data Science i najpopularniejszym do zastosowań związanych z ML oraz AI. Sposób w który jest zaprojektowany pozwala na łatwe prototypowanie rozwiązań w celu przeprowadzenia doświadczeń. Dzięki wykorzystaniu narzedzi takich jak Jupyter Notebook pozwala na bardzo dynamiczną pracę z konkretnymi danymi. Dodatkowo posiada wiele bibliotek oraz rozszerzeń zaprojektowanych do wydajnego implementowania algorytmów ML. Ostatecznym argumentem dla użycia tego języka jest jego pełna integracja ze środowiskiem gry Halite, i jest on wyróżniony jako polecany przez twórców gry.

3.2 Przetwarzanie danych

Ze względu na to że zaproponowany przez nas projekt ma dosyć nietypowy charakter, wszystkie przetwarzane przez nas dane będą wygenerowane na podstawie eksperymentów. Algorytmy które będziemy stosowac będą w większości należały do rodziny Unsupervised Learning takich jak Reinforced Learning czy Evolutionary Alghoritms. Dodatkowo dzięki popularności konkursu będziemy mogli przetestować nasze rozwiązania w rywalizacji z botami zaprojektowanymi przez inne zespoły, często profesjonalistów. Pozwoli to na uwypuklenie ich cech w różnych środowiskach.

Dodatkowo wykorzystując dane z poprzednich rozgrywek innych graczy możemy wykorzystać rozwiązania takie jak regresja liniowa do ekstrakcji pewnych zwycięzkich strategii.

3.3 Wstępne interakcje z danymi

Pierwszym krokiem do uzyskania danych potrzebnych w naszym eksperymencie jest podstawowa integracja ze środowiskiem. W tym celu przed pierwszym checkpointem

projektu skupiliśmy się na zapoznaniu ze środowiskiem gry. Rozpoczeliśmy od pobrania i skonfigurowania plików gry na maszynie lokalnej wykorzystując instrukcje twórców konkursu. Następnie uruchomiliśmy podstawowego bota dostarczonego w pakiecie startowym. Zaznajomiliśmy się z API udostępnianym przez środowisko gry, a także poznaliśmy możliwości interakcji z nią poprzez inspekcję plików gry. Ostatecznie podjeliśmy się prostych modyfikacji podstawowego bota, modyfikując sposób sterowania statków, a także próbując zapobiec kolizjom. Dzięki tym operacjom mogliśmy zapoznać się z plikami replay oraz errorlog które są podstawowymi źródłami informacji o przebiegu rozgrywki. Mogliśmy też przetestować różne sposoby produkcji logów, dzięki czemu w przyszłości będziemy mogli szybki zbierać dane z dużej ilości symulacji.

4 Analiza istniejących rozwiązań

Kolejnym krokiem do przygotowanie pod utworzenie sieci neuronowej była analiza istniejących rozwiązań. Najlepiej działające rozwiązania wykorzystywały algorytmy i drzewa warunków w celu wybrania najlepszego rozwiązania. Rozwiązania wykorzystujące sieci istniały, jednak nie osiągały tak dobrych wyników. Popularną metodą przy wykorzystaniu sieci neuronowych było podejście hybrydowe. Algorytm generował kilka początkowych ruchów za pomocą algorytmów, po czym przechodził do sieci neuronowej i na koniec rozgryki wracał do algorytmu w celu wykonania optymalnych kroków, które powtarzały się w innych rozwiązaniach. Rozwiązania miały różne pomysły u swoich podstaw. Nietóre oczyszczały okolicę bazy w celu minimalizacji czasu powrotu inne wyszukiwały miejsca o największym zgromadzeniu surowca, w celu maksymalizacji zysków. Większość rozwiązań nie starała się unikać przeciwnika, a jedynie minimalizować ilość kolizji w obrębie własnej floty. Rozwiązania zbierające surowce w okolicy bazy nie wykorzystywały dodatkowych punktów dostawczych, podczas gdy rozwiązania o szerszym polu działania rozstawiały punkty bardziej losowo, często przekształcając pojazdy, które nie były się w stanie dalej przemieścić w celu redukcji zużycia surowców na nowe, specjalnie przeznaczone do tego statki. Większość algorytmów, niezależnie od wcześniejszego sposobu działania, w końcowych turach rozgrywki rozbijała własne statki w punktach dostawczych. Na

pierwszy rzut oka wydawać się to może dziwne, jednak statki trafiające w ten sposób do baz oddawały zebrane surowce, a następnie przy zderzeniu oddawały część z surowców zużytych przy budowie statku. Większość z 50 najlepiej działających algorytmów działa w ten sposób. Różnice polegały najczęściej na określeniu czasu wykonania tej części. Część algorytmów uzależniała to od długości gry, inni od odległości statków od bazy. To podejście wymagało również, by statki tworzone były w miarę możliwości parami, w celu usunięcia wszystkich statków na koniec gry. Rozwiązania towrzące nowe punkty dostaw posiadały problem, gdy statki trafiały do różnych punktów dostaw i nie rozbijały się ze względu na to. Rozwiązania agresywne osiągały często lepsze wyniki, jednak zdarzały się sytuacje, w których wszystkie statki rozbijały się o statki gracza defensywnego uniemożliwiając kontynuajcę gry. Rozwiązania defensywne wykonywały obliczenia znacząco szybciej niż ich agresywne odpowiedniki. Przy tworzeniu sieci neuronowych należało będzie zwrócić uwagę na ilość danych wejściowych. Agresywne rozwiązania algorytmicze często dla każdego statku przetwarzały wszystkie dane z istniejącej planszy, podczas gdy defensywne działały na znacząco mniejszej ilości danych wejściowych.