Porównanie metod nauczanie przez wzmacnianie na podstawie gry wideo

Kacper Majkowski

24 grudnia 2025

Spis treści

1 Wstęp

1.1 Motywacja

W dzisiejszych czasach nieiwele jest dziedzin nauki które rozwijają się tak szybko jak sztuczna inteligencja. Technologie, które jeszcze dekadę czy dwie temu byłyby uznane za science fiction, używane są dzisiaj na porządku dziennym. Rozwój takich narzędzi jak ChatGPT czy Copilot całkowicie odmieniły sposób w jaki ludzie rozwiązują problemy, czy to w pracy czy poza nią. Jedną z dziedzin nauczania maszynowego jest Nauczanie przez Wzmacznianie (ang. Reinforcement Learning, RL)

1.2 Nawiązanie do natury

Wspomniane było wcześniej że nauczanie przez wzmacnianie jest powiązane z tym jak zwierzęta uczą się na podstawie doświadczeń. Możemy to również zaobserwować u ludzi. Jeżeli dziecko dotknie raz gorącej kuchenki i się oparzy, to najprawdopodobniej więcej już tego nie zrobi. Innymi słowy, podjęło ono pewną akcję, za którą zostało "ukarane", więc w przyszłości będzie tego unikać. Z drugiej strony jeżeli dziecko znajdzie szafkę ze słodyczami, to będzie chodzić tam częściej, wiedząc że za każdym razem czekają tam na nie słodycze. Mówiąc inaczej, wykonało ono pewną akcję i dostało za to "nagrodę", więc w przyszłości będzie próbowało powtórzyć tę akcję.

Analogicznie możemy spojrzeć na to jak zachowują się zwierzeta. Wyobraźmy sobię mysz która wkrada się do domu, szukając pożywienia. Jeżeli natrafi ona na groźnego kota, to od tej pory będzie już wiedzieć aby do tego domu nie chodzić. Ale czy na pewno? Może kot strzeże wartościowej spiżarni? Może nagroda byłaby wystarczająca aby mysz podjęła się tego ryzyka?

Są to życiowe, uproszczone przykłady, ale dobrze pokazują sposób działania nauczania przez wzmacnianie oraz problemy jakimi się ono zajmuje.

1.3 Intuicja

Możemy więc zdefiniować pewną intuicję, która pozwoli zrozumieć nam na czym polega nauczanie ze wzmacnieniem. Podmiot (nazywany później "agentem") znajduje się w pewnej sytuacji (stanie) i ma do podjęcia decyzję (akcję). Za tę akcję dostaje następnie nagrodę lub karę i ją zapamiętuje. Na podstawie tego może w przyszłości podjąć decyzję, kiedy znajdzie się w tym stanie ponownie w przyszłości to czy powinien podjąć tę samą decyzję. W późniejszych rozdziałach przedstawiona zostanie bardziej dokładna

i rygorystyczna definicja, jednakże ta intuicja pozwoli nam lepiej zrozumieć esencję nauczania ze wzmacnianiem, zanim przejdziemy do bardziej matematycznych, statystycznych i programistycznych rozważań.

2 Historia nauczania przez wzmacnianie

2.1 Początki

Współczesne nauczanie przez wzmacnianie jest wynikiem połączenia kilku dziedzin nauki i ich aplikacji dla programów komputerowych. Jedną z tch dziedzin jest psychologia, a konkretnie badania mające ustalić czy zwierzęta potrawią się uczyć, a jeżeli tak to na podstawie jakiego mechanizmu. Już na początku XX wieku naukowcy znajdowali dowody na to że zwierzęta faktycznie uczą się na podstawie doświadczeń

"Z kilku odpowiedzi na tę samą sytuację te, które są połączone z satysfakcją zwierzęcia lub następują bezpośrednio po niej, będą, przy innych równych warunkach, silniej powiązane z sytuacją, tak że gdy sytuacja się powtórzy, będą się częściej powtarzać; te, które są połączone z dyskomfortem zwierzęcia lub następują bezpośrednio po nim, będą, przy innych równych warunkach, słabiej powiązane z tą sytuacją, tak że gdy sytuacja się powtórzy, będą się rzadziej powtarzać. Im większa satysfakcja lub dyskomfort, tym silniejsze lub słabsze jest połączenie." Thorndike (2017)

Kolejną dziedziną nauki która przyczyniła się do rozwoju nauczania przez wzmacnianie jest Teoria Sterowania Optymalnego (eng. Optimal Control Theory). Jednym z naukowców zajmujących się tym tematem w latach 50-tch XX wieku był Richard Bellman. Jego wkład przyczynił się do rozwoju programowania dynamicznego, zdefiniowania Równania Bellmana oraz w końcu na ich podstawie do rozwoju nauczania przez wzmacnianie.

Dziedzina ta z początku nie cieszyła się zbytnią popularnością, jednak w końcu ktoś postanowił dogłębniej zbadać nauczanie przez wzmacnianie i odkryć jego prawdziwy potencjał.

2.2 Odrodzenie

W 1979 roku, Richard S. Sutton i Andrew G. Barto pracujący w University of Massachusetts zaczęli pracę nad projektem opartym o teorię A. Harry Klopfa "heterostatic theory of adaptive systems". Zdali sobie sprawę że nauczanie przez wzmacnianie, lub jak oni to nazywali "nauczanie hedonistyczne", kryje w sobie ogromne możliwości. W poprzednich latach wiekszość naukowców pracujących nad sztuczną inteligencją

najczęściej ignorowało nauczanie przez wzmacnianie i szybko przechodzili do innych dziedzin, takich jak klasyfikacja, nauczanie nadzorowane lub dziedzin całkiem niezwiązanych ze sztuczną inteligencją. Z perspektywy czasu wiemy jednak że nauczanie przez wzmacnianie ogromnie pomogło całej dziedzinie sztucznej inteligencji, zwłaszcza po rozwoju komputerów i wzroście możliwości obliczeniowych. Sutton and Barto (2018)

2.3 Historia zastosowań w grach

Od wczesnych lat nauczanie przez wzmacnianie znalazło wiele zastosowań. Jednym z nich była nauka grania w gry. Bardzo łatwo można przełożyć koncept "gracza" wykonującego "ruchy" na "agenta" podejmującego "akcje".

Jednym z pierwszych przykładów takiego zastosowania jest program grający w warcaby stworzony w 1959 roku przez Artura L. Samuela. Mimo że nazwa "nauczania przez wzmacnianie" jeszcze nie istniała, to z perspektywy czasu możemy powiedzieć że działa on poprzez właśnie ten proces. Program Samuela stosuje metodę prób i błędów, zapamiętuje je, uczy się ze swoich poprzednich doświadczeń i stara się polepszać swoje działanie. Samuel (1959)

Od programu Samuela musiało minąć trochę czasu, lecz w końcu zaczęto wykorzystywać nauczanie przez wzmacnianie na większą skalę. W latach 90-tych Gerald Tesauro z IBM Thomas J. Watson Research Center opracował TD-Gammon - program do gry backgammon, który osiągał wyniki zbliżone do ówczesnych mistrzów świata. Tesauro et al. (1995)

Jednym z nowszych przełomów dla nauczania przez wzmacnianie jest gra w szachy. Kiedy Deep Blue pokonał Garri Kasparowa w 1997 roku, nie używał on nauczania przez wzmacnianie, lecz bardziej klasycznego podejścia opierająca się na metodzie minimax połącznoego z ogromną, ręcznie zaprogramowaną funkcją oceny. Hsu (1999) Nauczanie przez wzmacnianie zostało w szachach spopularyzowane dopiero w 2017 poprzez porgram AlphaZero od Google DeepMind. Zahavy, Veeriah, Hou, Waugh, Lai, Leurent, Tomasev, Schut, Hassabis, and Singh (2023) Po 9 godzinach treningu w grach z samym sobą pokonał on uważany wtedy za najsilniejszy silnik szachowy - Stockfish 8. Bratko (2018) Od tamtego czasu, większość silników szachowych, w tym Stockfish, zaczęły używać nauczania przez wzmacnianie.

W tym samym roku AlphaZero pokonał również najlepszy silnik do gry w shogi - Elmo Silver, Hubert, Schrittwieser, Antonoglou, Lai, Guez, Lanctot, Sifre, Kumaran, Graepel, et al. (2017), oraz rok wcześniej pokonał on najlepszego gracza na świecie w grze Go. Granter, Beck, and Papke Jr (2017)

3 Czym jest nauczanie przez wzmacnianie

Czym w sumie jest nauczanie przez wzmacnianie? Wiadomo że jest dziedziną nauczania maszynowego, ale czy możemy przypisać je do jakiegoś działu nauczania maszynowego? W dziedzinie nauczania maszynowego możemy zidentyfikować dwie duże podgrupy - nauczanie nadzorowane oraz nauczanie nienadzorowane. Czy więc nauczanie przez wzmacnianie jest podzbiorem jednej z tych grup?

3.1 Nauczanie nadzorowane

Możemy zauważyć pewne wspólne cechy z nauczaniem nadzorowanym - model uczy się za na podstawie pewnych kryteriów. Klasycznym przykładem nauczania nadzorowanego jest identyfikacja przedmiotów na zdjęciach. Tam, model uczy się przewidując przedmiot na obrazie. Po takiej klasyfikacji porównuje on swoje przewidywania z etykietowanym zbiorem poprawnych odpowiedzi. Na podstawie tego które elementy przewidział poprawnie a które niepoprawnie, "uczy się" i poprawia swój model przewidywania.

Tutaj właśnie leży różnica pomiędzy nauczaniem nadzorowanym a nauczaniem przez wzmacnianie. Mimo że podczas nauczania przez wzmacnianie agent dostaje nagrody i kary, nie mówi mu to który wybór jest "poprawny" albo "niepoprawny". Czasem w pewnym stanie wykonanie ruchu który daje mniejszą natychmiastową nagrodę będzie prowadzić to większej nagrody w przyszłości. Agent nie dostaje listy poprawnych ruchów, tylko na podstawie doświadczeń i brania przyszłych stanów pod uwagę musi sam wywnioskować które ruchy są "poprawne".

3.2 Nauczanie nienadzorowane

Nauczanie nienadzorowane różni się od nadzorowanego tym że nie podajemy modelowi etykiet, lecz model ma sam wywniosować ze struktury danych ukryte wzorce lub grupy. Używane jest ono często na przykład do wykrywania anomalii w dużych zbiorach danych. Model nauczania nienadzorowanego dostaje sam zbiór danych, bez żadnych informacji zwrotnych i sam ma znaleść w nim wzorce.

Widać tutaj wyraźną różnicę W przeciwieństwie do nauczania nienadzorowanego, które nie dostaje informacji zwrotnych w trakcie uczenia, podczas nauczania przez wzmacnianie agent dostaje informacje na temat akcji jakie podejmuje. Nie są to może konkretne odpowiedzi "dobrze" lub "źle", ale nie musi znajwać on ukrytych informacji na podstawie samego środowiska.

3.3 Nauczanie sekwencyjne

Kolejnym dużym elementem wyróżniającym nauczanie przez wzmacnianie od nauczania nadzorowanego lub nienadzorowanego jest fakt iż polega ono na uczeniu sekwencyjnym. Oznacza to że wybór podjęty w jednym momencie wpływa na przyszłe decyzje agenta. Jest to zasadnicza różnica od innych sposobów nauczania. Na przykład model który uczy się rozpoznawania obrazów za pomocą naczuczania nadzorowanego nie uczy się sekwencyjnie - to jak zklasyfikuje jeden obraz nie wpływa na to jak powinien zklasyfikować następny.

Nauczanie przez wzmacnianie powiązuje jednak szeregi decyzji ze sobą. Jeśli agent dostanie nagrodę za daną akcję, to nie jest efekt tylko ostatniej decyzji, ale ciągu decyzji które doprowadziły go do tego punktu. Z drugiej strony nagrody są często opóźnione w czasie. Agent po podjęciu decyzji nie wie czy była ona poprawna czy zła, dowie się on tego dopiero po jakimś czasie kiedy odkryje do jakiego rezultatu ona doprowadziła. Innymi słowy decyzje agenta mają długoterminowe konsekwencje.

Kolejnym elementem który odróżnia nauczania przez wzmacnianie od innych metod jest to że agent zmienia swoje własne środowisko. Przy nauczaniu nadzorowanym lub nienadzorowanym model dostaje szereg danych z których ma się uczyć i nie ma wpływu na to jakie dane dostaje. Podczan nauczania przez wzmacnanie jest inaczej. To jaką decyzję agent podejmie w danym stanie ma bezpośredni wpływ na to w jakim stanie znajdzie się w następnym kroku i przed jakim wyborem zostanie postawiony. Innymi słowy to agent wybiera swoje "dane wejściowe". Jest to właściwość która występuje jednynie w nauczaniu przez wzmacnianie.

Podsumowując, największą różnicą pomiędzy nauczaniem przez wzmacnianie a nauczaniem nadzorowany i nienadzorowanym jest fakt że celem agenta jest maksymalizacja nagrody w czasie, a nie tylko tu i teraz. Agent musi wziąć pod uwagę nie tylko jaką nagrodę dostanie za obecny wybór, ale również do jakich stanów ten wybór doprowadzi oraz jakich nagrów może on się spodziewać. Ten niuans nie występuje przy innych typach nauczania, gdzie jedyne co się liczy to poprawny wybór tu i teraz.

4 Elementy nauczania przez wzmacnianie

Poza agentem i środowiskiem, nauczanie przez wzmacnianie składa się z kilku elementów które należy poznać i zdefiniować aby zrozumieć jego sposób działania. Możemy także zauważyć tutaj porównania do biologi, z której nauczanie przez wzmacnianie bierze dużą inspirację.

4.1 Polityka

Polityka mówi agentowi jak powinien się zachować w danym stanie. Jest to w zasadzie funkcja, która danemu stanowi przypisuje konkretną akcję. Może ona przybierać różne formy, takie jak tablica stan/akcja, lub może być to sieć neuronowa, która na wejściu otrzymuje stan, a na wyjściu zwraca akcję. Czasem jest one deterministyczna, zawsze dająca ten sam wynik dla danego stanu, a czasem może być probabilistyczna. Jeśli polityka jest probabilistyczna to nazywana jest "polityką stochastyczną". Polityka leży w sercu nauczania przez wzmacnianie. Celem nauczania jest ciągłe poprawianie i ulepszanie polityki.

W biologii polityce odpowiada zjawisko, które w psychologia nazywa modelem "bodziec-reakcja". Jest to zbiór zasad który mówi organizmowi jak powinien zachować się kiedy napotka dany bodziec. Najczęściej te rekacje są podświadome. Jeżeli dotkinemy gorącej kuchenki to nie musimy przeanalizować tego faktu w głowie i świadomie odsunąć ręki, nasz mózg ma zapamiętaną poprawną reakcję i sam to za nas zrobi zanim jeszcze zdamy sobie sprawę co się stało. Sutton and Barto (2018)

4.2 Nagrody

System nagród definiuje cel problemu nauczania przez wzmacnianie. Po każdej akcji agent otrzymuje nagrodę, która jest reprezentowana przez liczbę. Jeżeli agent wykonuje akcję która pomaga mu zrealizować cel to dostaje większą nagrodę niż za akcje które go od celu oddalają. Całym zadaniem agenta jest zmaksymalizowanie wartości nagród które otrzymuje w czasie.

W biologii jako nagrody o kary odpowiadają uczuciom przyjemności oraz bólowi. Jeżeli organizm odczuwa przyjemność to jest dla niego sygnał że wykonał dobrą rzecz. Jeżeli odczuje ból to oznacza że wykonał błąd i powinien w przyszłości tej czynności unikać.

4.3 Funkcja wartości

Funkcja wartości jest używana do oceny wartości akcji w danym stanie. Przewiduje ona spodziewaną długoterminową wartość danej akcji, biorąc pod uwagę nie tylko możliwie natychmastowo największą nagrodę, ale również wszystkie przyszłe potencjalne nagrody. Często agent może znaleść się w sytuacji gdzie mniejsza natychmiastowa nagroda będzie prowadzić do większej nagrody na przestrzeni całego epizodu, zaczynając od obecnego stanu.

Funkcja wartości jest o tyle ważna że to właśnie na jej podstawie agent będzie decydował jaką decyzję podjąć. Będzie wybierał akcje które maksymalizują nie natychmiastową nagrodę, ale długoterminową funkcję wartości. Jednak obliczanie funkcji wartości nie jest takie proste. Gdzie natychmiastowe nagrody za ruch są podawane prosto ze środowiska, to jak obliczyć długoterminową funkcję wartości jest o wiele bardziej skomplikowane. Problem ten leży w sercu nauczania przez wzmacnianie i wydajność różnych metod obliczania funkcji wartości jest jedną z najważniejszyk kwestii w tej dziedzinie.

Nawiązując do biologii, jeśli nagroda jest natychmastowym odczuciem przyjemności lub bólu, to funkcja wartości byłaby ogólną oceną jak organizm jest zadowolony z styuacji w której się znajduje, czy obecna sytuacja może porwadzić to większej ilości przyjemności i mniejszej ilości bólu.

4.4 Model środowiska

Ostatnim elementem nauczania przez wzmacnianie jest model środowiska. Model środowiska jest rzeczą która odwzorowuje zachowanie prawdziwego środowiska i pozwala agentowi na przewidywanie przyszłych stanów i potencjalnych nagród. Dla przykładu, jeżeli agent znjaduje się w danym stanie to może przewidzieć w jakim stanie się znajdzie jeśli wykona pewną akcję. Pomaga to agentowi lepiej planować swoje akcje.

Nauczanie przez wzmacnianie nie musi jednak zawierać modelu środowiska. Jeśli metoda nauczania zawiera model to jest nazywana "modelową". Jeżeli jednak metoda nie zawiera modelu środowiska i bazuje wyłącznie na metodzie prób i błędów, nazywana jest "bezmodelową".

5 Przykład nauczania przez wzmacnianie

Aby lepiej zrozumieć jak te elementy ze sobą współpracują, najlepiej spojrzeć jak odpowiadają one przykładom nauczania przez wzmacnianie z prawdziwego życia:

5.1 Gra w szachy

Środowisko: Szachownica i bierki

Agent: Gracz

Stan: Obecna pozycja bierek na szachownicy

Akcja: Legalny ruch w danym stanie

Nagrody: Pozytywne za dobre ruchy (zbicie bierek, wygranie partii), negatywne za złe ruchy (strata bierek, przegranie partii)

5.2 Robot sprzątający

Środowisko: Mieszkanie

Agent: Robot

Stan: Pozycja robota, wykryty brud, przeszkody, poziom baterii

Akcja: Jazda prosto, skręt, zacząć odkurzać, przestać odkurzać, zatrzymać się

Nagrody: Pozytywne za sprzątanie brudu, negatywne uderzenie w przeszkody i marnowanie energii

5.3 System rekomendacji (np. Youtube, Netflix)

Środowisko: Użytkownik i jego zachowania

Agent: Algorytm rekomendujący treści

Stan: Historia oglądania, preferencje użytkownika, bieżący kontekst

Akcja: Wyświetlenie konkretnej treści

Nagrody: Pozytywne jeżeli użytkownik włączy polecane wideo i obejrzy je do końca, negatywne jeśli je pominie lub przerwie

6 Proces decyzyjny Markowa

Do tej pory definiowaliśmy elementy nauczania przez wzmacnianie w sposób raczej nieformalny. Warto więc zapoznać się z formalną definicją nauczania przez wzmacnianie, a konkretnie to problemu który próbuje ono rozwiązać

6.1 Elementy MDP

Proces decyzyjny Markowa (ang. Markov decision process - MDP) jest formalnym matematycznym modelem podejmowania decyzji w czasie. MDP składa się z kilku elementów, które poznaliśmy przy nauczaniu przez wzmacnianie:

S: Zbiór wszystkich możliwych stanów

A: Zbiór możliwych akcji

P(s' | s, a): Funkcja przejścia. Mówi ona jak zmieni się środowisko po wykonanej akcji. Jeżeli w stanie **s** wykonana zostanie akcja **a** to stan środowiska zmieni się na **s'**

R(s, a): Funkcja nagrody. Za wykonanie akcji **a** w stanie **s** przyznawana jest nagroda **R(s, a)**

6.2 Własność Markowa

To co umożliwia nam stosowanie modelu nauczania przez wzmacnianie w ten sposób tzw. Własność Markowa. Mówi ona, że przyszłość modelu zależy tylko i wyłącznie od obecnego stanu i podjętej akcji, nie od przeszłości. Innymi słowy, decyzje podjęte w przeszłości nie wpływają na obecny wybór. Pozwala nam to w Funkcji Wartości brać pod uwagę jedynie obecny stan i możliwe akcje.

W praktyce istnieją techniki nauczania przez wzmacnianie również dla modeli które nie spełniają własności Markowa. W takich sytuacjach stosuje się różne rozwiązania, na przykład stany które zawierają sobie dane o przeszłych akcjach. Jednak w tej pracy nie będziemy się takimi przypadkami zajmować. Wszystkie problemy które będziemy rozwiązywać będą spełniać własność Markowa.

7 Polityka - Podejście Tabularyczne

W rozdziale o polityce wspomniane zostało że może ona przybierać różne formy. Dla przypomnienia, polityka to mechanizm w jaki agent wybiera akcje które wykonuje. Innymi słowy jest to funkcja (deterministyczna lub probabilistyczna), która określia sposób wyboru akcji agenta na podstawie stanu w którym się znajduje. Przyjrzymy się jednej z najprostrzych implementacji polityki - Q-Tabela.

7.1 Q-Tabela

Q-Tablela jest prostym sposobem implementacji polityki agenta. Przypisuje ona po prostu dla każdej możliwego stanu i każdej akcji wartość tej akcji. Ważne jest tutaj do zaznaczenia że nie jest to nagroda za tę akcję, ale funkcja wartości o której mówiliśmy wcześniej.

stan/akcja	akcja 1	akcja 2	akcja 3	akcja 4
stan 1	4	5	-2	-10
stan 2	-3	-5	1	10
stan 3	7	2	0	0
stan 4	-5	-6	8	-4
stan 5	1	1	-2	3

Tabela 1: Przykładowa Q-Tabela

Powyżej została przedstawiona przykładowa Q-Tabela dla pewnego środowiska które ma 5 możliwych stanów i w każdym stanie agent ma do wyboru 4 akcje. Tabela zawiera wyliczone funkcje wartości dla każdej pary stan-akcja. Celem nauczania przez wzmacnianie jest ciągłe aktualizowanie wartości w tej tabeli.

7.2 Aktualizowanie wartości Q-Tabeli

Q-Tabela z początku nie będzie oczywiście optymalna, częstym podejściem jest zaczynanie z Q-Tabelą zawierającą same zera. Jak więc możemy poprawiać jej wartości na podstawie nowych doświadczeń? Zacznijmy najpierw od najprostszego pomysłu i stopniowo będziemy go ulepszać.

7.2.1 Natychmiastowa nagroda

Weźmy najprostszy pomysł, gdzie zastępujemy wartość właśnie zdobytą nagrodą:

$$Q(s,a) \leftarrow r(s,a)$$

$Q(s,a)$ - Obecna wartość dla stanu s i akcji a

$r(s,a)$ - Nagroda za wykonanie akcji a w stanie s

7.2.2 Przyszłe nagrody

Jest to początek, ale podczas uczenia powinniśmy wziąć więcej aspektów pod uwagę niż tylko natychmastowa nagroda. Jak na razie nie uwzględniamy przyszłych stanów, do których wybór może nas dobrowadzić. Dodajmy więc wyraz, który reprezentuje wartość najlepszej akcji w następnym stanie, czyli w stanie s'

$$Q(s,a) \leftarrow r(s,a) + \max_{a'} Q(s',a')$$

s' - Stan po wykonaniu akcji a w stanie s

Skąd jednak mamy znać najlepszy ruch w stanie s'? Tutaj z pomocą przychodzi nam idea znana z programowania dynamicznego. Zamiast stosować tę formułę rekurencyjnie to wartość $\max_{a'} Q(s', a')$ podejrzymy prosto z Q-Tabeli.

Możemy również rozwinąć ten wzór aby lepiej zrozumieć co tak naprawdę tutaj obliczamy. Jeśli jest to formuła dla $Q(s, a)$, to możemy ją również zastosować jako podstawienie za $Q(s', a')$. Otrzymujemy wtedy:

$$Q(s, a) \leftarrow r(s, a) + r(s', a') + \max_{a''} Q(s'', a'')$$

Warto zaznaczyć że $r(s', a')$ oznacza tutaj nagrodę jaką dostaniemy, jeśli w stanie s' wykonamy akcję o największej wartości, niekoniecznie największej natychmiastowej nagrodzie.

Możemy kontynuować ten proces, aż w końcu dostaniemy takie rozwinięcie:

$$Q(s, a) \leftarrow r(s, a) + r(s', a') + r(s'', a'') + r(s''', a''') + r(s'''', a'''') + ...$$

Możemy więc zauważyć, że ten wzór tak naprawdę wylicza sumę nagród jakie dostaniemy wykonując w każdym stanie akcje o największej wartości. Innymi słowy udało nam się uwzględnić przyszłe stany i nagrody w obliczeniu wartości pary stan-akcja.

7.2.3 Waga przyszłych nagród - Współczynnik dyskontowania

Pojawia się tutaj pewien problem. Czy agent powinien wartościować nagrody odległe w czasie tak samo jak te które może otrzymać natychmiast. Możliwe że istnieją sytuacje, kiedy takie zachowanie jest pożądane, ale najczęściej chcemy aby agent preferował otrzymać nagrodę szybciej niż później. Aby uzyskać taki efekt, dodamy do naszego wzoru nowy parametr - współczynnik dyskontowania γ.

$$Q(s, a) \leftarrow r(s, a) + \gamma * \max_{a'} Q(s', a')$$

W jaki sposób zmniejsza to wpływ nagród bardziej oddalonych w czasie? Po rozwinięciu formuły tak jak wcześniej otrzymujemy:

$$Q(s,a) \leftarrow r(s,a) + \gamma * r(s',a') + \gamma^2 * r(s'',a'') + \gamma^3 * r(s''',a''') + \gamma^4 * r(s'''',a'''') + ...$$

Możemy zauważyć, że kolejne nagrody są ważone przez kolejne potęgi współczynnika dyskontowania. Dla przykładu, jeżeli $\gamma = \frac{1}{2}$, to podczas obliczania wartości akcji, nagroda odległa o 2 ruchy będzie warta $\frac{1}{2}$ nagrody natychmiastowej, odległa o 3 ruchy będzie warta $\frac{1}{4}$ itd.

Można również sprawdzić co dzieje się dla przypadków granicznych. Dla $\gamma = 1$ agent przykłada taką samą wagę do każdej nagrody, nie ważne jak bardzo oddalonej w czasie. Dla $\gamma = 0$ znów nie bierze przyszłych nagród pod uwagę i rozpatruje jedynie natychmiastową nagrodę.

7.2.4 Współczynnik uczenia

Kolejnym elementem który dodamy do wzoru to współczynnik uczenia α. Obecnie agent całkowicie zapomina o starej wartości w Q-Tabeli i zastępuje ją nową wartością. Niesie to za sobą kilka problemów. Po pierwsze nowa wartość byłaby tylko jedną, najnowszą próbką, która może być obarczona różnymi błędami. W nauczaniu przez wzmacnianie checmy stopniowo ulepszać politykę, tak aby stabilnie zbliżała się do optymalnej polityki (często oznaczanej jako Q^*). Zastępowanie starej wartości nową całkowicie eliminuje tę stabilność i prowadzi do chaotyczności. Zostało również udowodnione, że polityka Q zbiega do polityki optymalnej Q^* jedynie gdy α maleje lub jest małe. Watkins and Dayan (1992)

Zmodyfikujmy więc wzór tak aby wziąć pod uwagę poprzednią wartość w Q-Tabeli

$$Q(s,a) \leftarrow (1-\alpha) * Q(s,a) + \alpha * (r(s,a) + \gamma * \max_{a'} Q(s',a'))$$

Modyfikując współczynnik α możemy wybrać wagę nowej wartości w porównaniu do starej, gdzie przykładowo $\alpha = 0.1$ oznacza że agent jako nową wartość bierze sumę starej wartości z wagą 0.9 i nowej wartości z wagą 0.1.

Możemy znów zbadać przypadki graniczne. $\alpha = 0$ oznacza że agent nie bierze nowej w ogóle pod uwagę (czyli nic się nie uczy), $\alpha = 1$ oznacza że agent zamienia całkowicie starą wartość na nową.

7.2.5 Równanie Bellmana

W ten sposób otrzymaliśmy równanie Bellmana, jeden z najważniejszych konceptów nauczania przez wzmacnianie. Pozwala on agentowi na ulepszanie funkcji wartości na podstawie nowych doświadczeń. Podczas uczenia, po każdym kroku agent będzie obliczał nową wartość właśnie za pomocą tego wzoru.

Często można spotkać się również z równaniem Bellmana w takiej formie:

$$Q(s,a) \leftarrow Q(s,a) + \alpha * (r(s,a) + \gamma * Q(s',a') - Q(s,a))$$

Jednak można łatwo pokazać że jest ono równoważne z równaniem które wyprowadziliśmy wcześniej.

$$Q(s,a) \leftarrow Q(s,a) + \alpha * (r(s,a) + \gamma * \max_{a'} Q(s',a') - Q(s,a))$$
$$Q(s,a) \leftarrow Q(s,a) + \alpha * r(s,a) + \alpha * \gamma * \max_{a'} Q(s',a') - \alpha * Q(s,a)$$
$$Q(s,a) \leftarrow Q(s,a) - \alpha * Q(s,a) + \alpha * r(s,a) + \alpha * \gamma * \max_{a'} Q(s',a')$$
$$Q(s,a) \leftarrow (1-\alpha) * Q(s,a) + \alpha * (r(s,a) + \gamma * \max_{a'} Q(s',a'))$$

8 Eksploracja, Eksploatacja

Ważnym aspektem nauczania przez wzmacnianie jest mechanizm przez który agent decyduje kiedy podczas uczenia ma on wybrać najlepszy ruch. Błędem byłoby, żeby agent podążał ciągle najlepszą według siebie ścieżką, gdzyż nie odkrył by nigdy potencjalnie lepszych ścieżek. Nie powinien on również ciągle chodzić losowo, chcemy aby wykorzystywał on wiedzę jaką zdobył i na jej podstawie uczył się dalej. Musimy więc znaleść balans eksploracji, czyli wybierania losowych ścieżek, a eksploatacji, czyli wybierania najlepszej akcji. Rozwiązaniem tej sytuacji współczynnik eksploracji ϵ.

8.1 Współczynnik eksploracji

Współczynnik eksploracji przyjmuje wartości od 0 do 1. Mówi on agentowi jaką część ruchów ma wykonywać losowo, a jaką część ma wykonywać zgodnie z najlepszą wartością zgodnie ze swoją polityką. Im wyższy współczynnik, tym częściej agent będzie wykonywał akcje losowo. Dokładniej, ϵ oznacza procent akcji które agent ma wykonywać losowo.

8.2 Zmiana współczynnika eksploracji podczas uczenia

Współczynnik eksploracji powinien zmieniać się w trakcie uczenia. Uczenia zaczyna się ze współczynnikiem eksporacji bliskim 1 - agent jeszcze nic nie wie o środowisku, więc powinien odkrywać jak nawięcej może. Z czasem, jak agent nabiera wiedzy współczynnik powinien się zmniejszać, tak aby agent używał wiedzy którą zgromadził.

Musimy więc znaleść sposób na zmniejszanie współczynnika eksploracji w trakcie uczenia. Istnieje wiele podejść, których użyteczność zależy od konkretnego przypadku. Dla przykładu najprostrzym sposobem jest liniowe zmniejszanie współczynnika.

$$\epsilon(n) = 1 - \frac{n}{N}$$

n - Obecny krok

N - Całkowita liczba kroków podczas uczenia

Kolejnym możliwym sposobem jest zmniejszanie współczynnika ϵ, tak aby agent spędził więcej czasu na eksploatację, innymi słowy aby częściej podążał najlepszą według siebie ścieżką. Dla takich przypadków nada się funkcja która szybciej maleje.

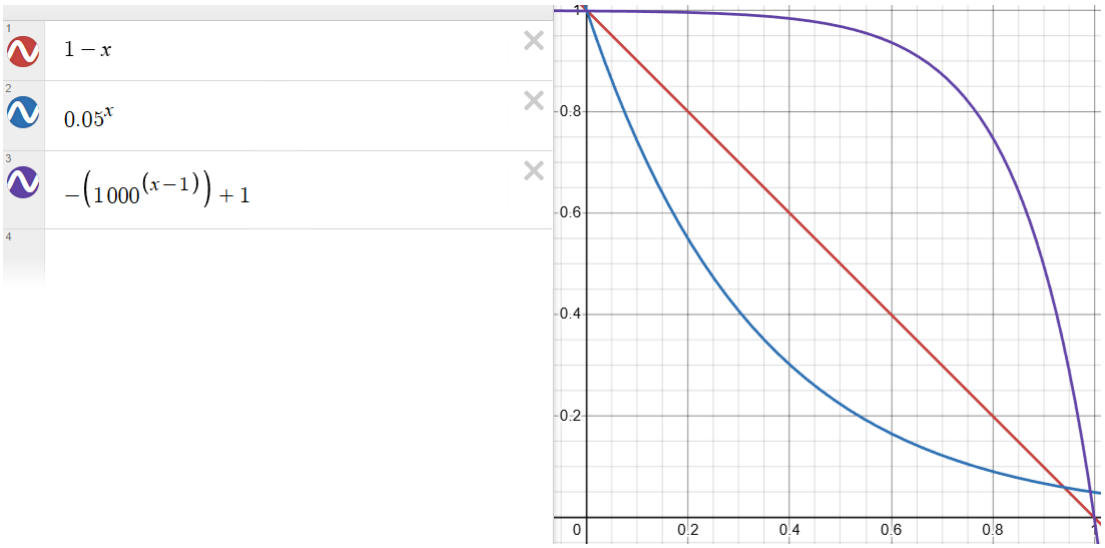
$$\epsilon(n) = t^{\frac{n}{N}}$$

t - Docelowy współczynnik ϵ w kroku N

Z drugiej strony jeżeli checmy aby agent spędził więcej czasu odkrywając środowisko powinniśmy użyć funkcji która dłużej utrzymuje wyższą wartość i maleje w późniejszym etapie nauczania.

$$\epsilon(n) = -(b^{(n/N-1)}) + 1$$

b - Współczynnik definiujący kształt funkcji, im większy tym dłużej ϵ utrzyma się na wysokim poziomie



Rysunek 1: Wykresy funkcji wygenerowane za pomocą programu desmos. W tym przypadku przyjęte zostało N=1 aby lepiej przedstawić kształt krzywych.

9 Polityka jako sieć neuronowa - Deep Reinforcement Learning

W poprzedich rodziałach omawialiśmy czym jest polityka oraz wspomniane było że może ona przyjmować różne formy. Do tej pory omówiliśmy jedną z nich - Q-Tabelę. W tym rodziale przedstawione zostanie inne podejście - wykorzystanie sieci neuronowych.

9.1 Deep Q-Network

Głębokie nauczanie przez wzmacnianie (and. Deep Reinforcement Learning - DRL) jest w wielu aspektach identyczne do nauczania które do tej pory omówiliśmy. Zasadniczą różnicą jest polityka. Dalej ma ona tę samą ideę - w trakcie uczenia jest doskonalona i na podstawie doświaczeń dla danego stanu zwraca akcję. Jednak zamiast sprawdzania wartości w Q-Tabeli, stan środowiska zostaje przepuszczony przez sieć neuronową, zwaną Deep Q-Network, która na wyjściu zwraca najlepszy według siebie stan.

9.2 Dlaczego sieć neuronowa?

Mimo że Q-Tabela sprawdza się bardzo dobrze, ma ona kilka sporych ograniczenie. Jednym z największych jest fakt, że przetrzymuje ona jedynie dyskretne stany. Nie jest to problemem jeżeli nasze środowisko jest dyskretne (np. gra w warcaby), ale problem się pojawia dla środowisk ze stanami ciągłymi. W prawdzimym życiu najczęściej spotkamy

się ze środowiskami które zawierają elementy przyjmujące wartości z pewnego zakresu (np. dla robota sprzątającego jego pozycja, poziom baterii itd.).

Rozwiązaniem tego problemu jest wykorzystanie sieci neuronowej. Metoda DQL (Deep Q-Learning), powstała aby pokonać ograniczenia Q-Tabeli.

9.2.1 Środowiska ciągłe

Pierwszym z nich jest problem który już omówiliśmy - środowiska ze stanami ciągłymi. Istnieją sposoby aby zastosować Q-Tabelę do takich środowisk, ale niosą one za sobą pewne koszty. Jednym ze sposobów jest dyskretyzacja - podzielenie ciągłego stanu na dyskretne części (np. podzielenie dwuwymiarowej powierchni na dyskretne kwadraty i traktowanie każdego jako osobną pozycję agenta). Problemem jest tutaj utrata dokładności informacji o stanie. Można dzielić stan na coraz to mniejsze części, ale powoduje to szybki wzrost możliwej liczby stanów, coś z czym Q-Tabela ma duże problemy.

Dla nauczania głębokiego z użyciem sieci neuronowej środowiska ciągłe nie są żadnym problemem, a w zasadzie są jego mocną stroną.

9.2.2 Przewidywanie dla podobnych stanów

Kolejną dużą zaletą nauczania głębokiego jest fakt, że sieć neuronowa potrafi rozpoznać kiedy stany są do siebie podobne. Dla Q-Tabeli, jeżeli stany różnią się w jakikolwiek sposób, nawet najmniejszy i nieistotny (np. robot sprzątający który ma 90% baterii a 89%), to są to dla niej kompletnie różne stany dla których musi się osobno uczyć jak się zachowywać.

Sieć neuronowa ma tę zaletę, że potrafi rozpoznać podobne stany wejściowe. Jeśli widziała już jakiś stan w przeszłości, to może często na tej podstawie "przewidzieć" najlepsze akcje dla stanów podobnych, mimo że ich nigdy nie widziała. Oczywiście, nie zawsze takie przewidywania będą poprawne, ale jest to o wiele lepsze niż uczenie się ich od zera.

9.2.3 Środowiska z dużą liczbą stanów

Jak wspomniano wcześniej, Q-Tabela nie radzi sobie dobrze ze środowiskami z dużą liczbą stanów. Ze względu na fakt, że musi ona przechowywać każdy możliwy stan osobno, jej rozmiar potrafi urosnąć bardzo szybko. Dokładniej, rozmiar Q-Tabeli dla środowiska zawierającego zbiór możliwych stanów S i zbiór możliwych akcji A wynosi $|S| * |A|$. Dla wielu środowisk, nawet dyskretnych, jest to całkowicie niepraktyczne.

Sieć neuronowa nie ma takiego problemu. Nie musi przechowywać wszystkich możliwych stanów, lecz tylko wagi wyliczone podczas uczenia. Jej rozmiar zależy jednynie od jej budowy, nie od liczby stanów i akcji.

9.3 Wady nauczania głębokiego

Nauczanie głębokie nie jest jednak objektywnie "lepszą wersją" nauczania używającego Q-Tabeli. Zostało ono stworzone, aby można było zastosować nauczania przez wzmacnianie do innych typów środowisk, lecz pomimo wielu zalet, ma ono również swoje wady.

9.3.1 Złożoność obliczeniowa

Mimo że jest bardzo wydajne pamięciowo, potrafi ono być bardzo wymagające pod względem mocy obliczeniowej. Używając Q-Tabeli, jeżeli chcemy odczytać wartość dla pary stan-akcja, to musimy po prostu odczytać pole z tabeli, co jest bardzo szybkie. Jeżeli chcemy jednak odczytać wartość dla pary stan-akcja przy użyciu DQN to musimy wykonać wiele obliczeń, co przy tysiącach iteracji znacząco zwiększa obciążenie i wydłuża proces uczenia.

9.3.2 Przejrzystość

Naturanlym problemem pracy z sieciami neuronowymi jest to że są to "czerne skrzynki" - dostają dane na wejściu i zwracają na ich podstawie wynik, ale najczęściej nie wiemy jak to robią. Możemy zobaczyć informację o wagach, neuronach itd, ale trudno zinterpretować który element odpowiada za co. Z tego powodu, w przypadku problemów z głębokim nauczaniem często trudo jest zdiagnozować w czym leży problem.

9.3.3 Stabilność

Jednym z największych problemów nauczania głębokiego jest ogólna niestabilność uczenia w porównaniu do metody Q-Tabeli. Po pierwsze sieć neuronowa działa na aproksymacjach, a nie na dokładnych wartościach. Dodatkowo jedna aktualizacja parametrów wpływa na wyniki wielu możliwych stanów, nawet tych których agent jeszcze nie widział. Ponadto, w przeciwieństwie do Q-Tabeli, nie ma dowodu na zbieżność nauczania DQN, co oznacza że model może oscylować lub całkowicie rozbiegać w trakcie nauczania.

9.4 Double DQN

Popularnym sposobem na zwiększenie stabilności nauczania głebokiego jest użycie dwóch sieci neuronowych - jednej bieżącej (ang. online network) i jednej docelowej (ang. target network). Pozwala to rozdzielić ocenę akcji od wyboru akcji. W klasycznym nauczaniu głębokim agent używa tej samej sieci do oceny akcji jak i do jej wyboru. Potrafi to prowadzić do oscylacji polityki, przeszacowywania wartości Q przez funkcję "max" oraz generalnej niestabilności. Można porównać to do psa który goni swój własny ogon.

Użycie dwóch sieci neuronowych pozwala rozwiązać ten problem. Do wyboru akcji będziemy używać sieci bieżącej, którą będziemy za każdym razem aktualizować. Z drugiej strony, do oceny akcji użyjemy sieci docelowej. Z początku jest ona kopią sieci bieżącej, lecz ona jest aktualizowana tylko co jakiś czas, na przykład po danej liczbie iteracji. Staje się ona wtedy znów kopią sieci bieżącej. W ten sposób mamy stabilny "cel", względem którego uczymy sieć.

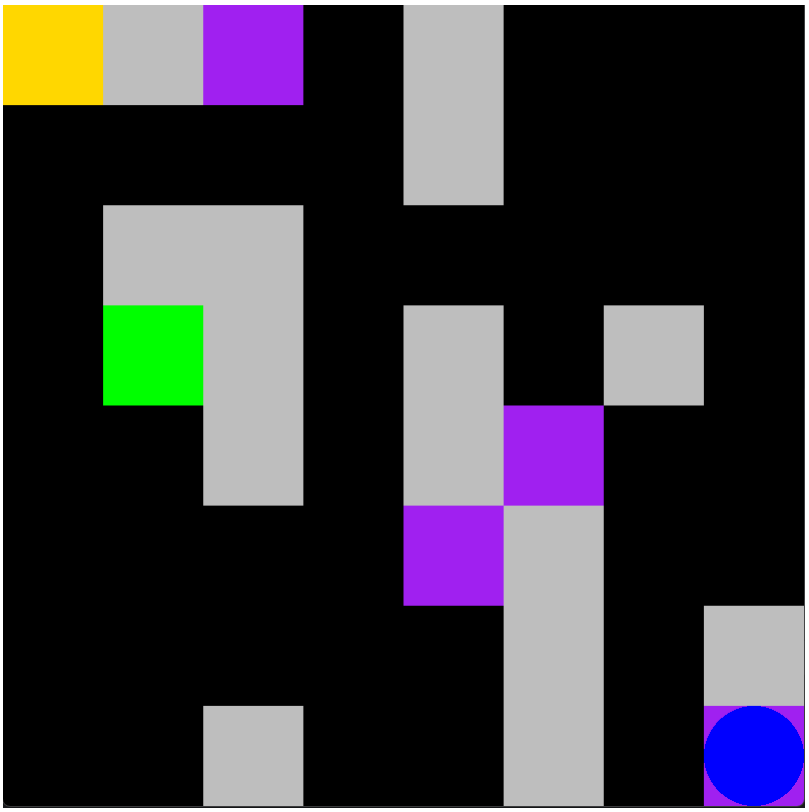
10 Gra Taxi

Do porównania tych metod nauczania wykorzystamy specjalnie zaimplementowaną grę "Taxi", opartą na popularnym problemie nauczania maszynowego.

10.1 Cel gry

Rozgrywka odbywa się na prostokątnej planszy. Na planszy wyznaczone są specjalne pola gdzie może pojawić się pasażer oraz jego cel. Celem taksówki jest jak najszybsze przemieszczenie się na pozycję pasażera, zabranie go, przemieszczenie się na pozycję celu i odstawienie pasażera. Jeżeli to się uda to pasażer i cel zmieniają pozycję na inne losowe pola specjalne. Na planszy występują również ściany, które taksówka musi omijać.

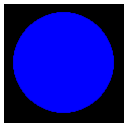
10.2 Przykładowa plansza



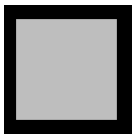
Rysunek 2: Przykładowy stan gry taxi. Elementy gry opisane poniżej

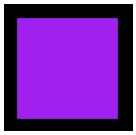
10.3 Elementy gry

 - Taksówka

 - Pasażer

 - Cel

 - Ściana

 - Możliwe punkty pojawienia się pasażera i celu

Literatura

Ivan Bratko. Alphazero–what's missing? *Informatica*, 42(1), 2018.

Scott R Granter, Andrew H Beck, and David J Papke Jr. Alphago, deep learning, and the future of the human microscopist. *Archives of pathology & laboratory medicine*, 141 (5):619–621, 2017.

Feng-hsiung Hsu. Ibm's deep blue chess grandmaster chips. *IEEE micro*, 19(2):70–81, 1999.

Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Richard S. Sutton and Andrew G. Barto. *Uczenie przez wzmacnianie. Wprowadzenie*. Wydawnictwo Naukowe PWN, Warszawa, 2018. Polskie tłumaczenie: Reinforcement Learning: An Introduction.

Gerald Tesauro et al. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

Edward Thorndike. *Animal intelligence: Experimental studies*. Routledge, 2017.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

Tom Zahavy, Vivek Veeriah, Shaobo Hou, Kevin Waugh, Matthew Lai, Edouard Leurent, Nenad Tomasev, Lisa Schut, Demis Hassabis, and Satinder Singh. Diversifying ai: Towards creative chess with alphazero. *arXiv preprint arXiv:2308.09175*, 2023.