ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORMATIKY

DIPLOMOVÁ PRÁCA

BC. MILOŠ FOLTÁN

Aplikácia strojového učenia založená na učení posilňovaním pre vybraný problém

Vedúci práce: Ing. Lukáš Falát PhD.

Registračné číslo: 967/2019

Žilina, 2020

ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORMATIKY

DIPLOMOVÁ PRÁCA

ŠTUDIJNÝ PROGRAM: INFORMAČNÉ SYSTÉMY

BC. MILOŠ FOLTÁN

Aplikácia strojového učenia založená na učení posilňovaním pre vybraný problém

Žilinská univerzita v Žiline

Fakulta riadenia a informatiky

Školiace pracovisko: Katedra makro a mikroekonomiky

Žilina, 2020

ŽILINSKÁ UNIVERZITA V ŽILINE, FAKULTA RIADENIA A INFORMATIKY.

ZADANIE TÉMY DIPLOMOVEJ PRÁCE.

Študijný program: Informačné systémy

Zameranie: Podniková informatika

Meno a priezvisko: Miloš Foltán

Osobné číslo: 558187

Názov práce v slovenskom aj anglickom jazyku

Aplikácia strojového učenia založená na učení posilňovaním pre vybraný problém

Application of Reinforcement Learning for Selected Problem

Zadanie úlohy, ciele, pokyny pre vypracovanie (Ak je málo miesta, použite opačnú stranu)

Cieľ diplomovej práce:
Na základe teoretických poznatkov a praktických zručností z oblasti strojového učenia vytvoriť aplikáciu pre vybraný problém s využitím strojového učenia založenom na učení posilňovaním (reinforcement learning).

Obsah:
1. Analýza súčasného stavu problematiky
2. Teoretické východiská vybraných metód
3. Návrh a implementácia modelu a užívateľskej aplikácie
4. Vyhodnotenie, výsledky, diskusia

Pri analýzach a vypracovaní práce je potrebné postupovať podľa interných smerníc fakulty.

Meno a pracovisko vedúceho DP: Ing. Lukáš Falát, PhD., KMME, ŽU

Meno a pracovisko tútora DP:

11 FEB. 2020

vedúci katedry
(dátum a podpis

Zadanie zaregistrované dňa 06.12.2019 pod číslom 967/2019 podpis

Čestné vyhlásenie

Čestne vyhlasujem, že som diplomovú prácu Aplikácia strojového učenia založená na princípe učenia posilňovaním pre vybraný problém vypracoval pod vedením vedúceho práce Ing. Lukáša Faláta PhD, na základe poznatkov získaných počas štúdia a informácií z dostupnej literatúry z uvedených zdrojov.

Poďakovanie

V prvom rade by som chcel poďakovať svojej pani manželke za podporu pri písaní tejto práce.

Ďalej by som chcel poďakovať vedúcemu mojej bakalárskej práce Ing. Lukášovi Falátovi Phd. za odbornú pomoc a nápady pri tvorbe tejto práce, ktoré mi pomohli pri tvorbe aplikácie a celkovom vypracovaní práce.

ABSTRAKT V ŠTÁTNOM JAZYKU

FOLTÁN, Miloš. *Aplikácia strojového učenia založená na učení posilňovaním pre vybraný problém. Diplomová práca.* – Žilinská univerzita v Žiline. Fakulta riadenia a informatiky; Katedra makro a mikroekonomiky. – Vedúci: Ing. Lukáš Falát Phd. - Stupeň odbornej kvalifikácie: inžinier. – Žilina: FRI UNIZA, 2020. Počet strán 81 s

Oblasť umelej inteligencie (AI) nie je neznámym pojmom v oblasti informatiky modernej doby. Výhody využitia AI si uvedomujú firmy po celom svete. V súčasnosti sa využíva vo viacerých podobách, napríklad vo forme učenia s učiteľom, bez učiteľa alebo čiastočnom učení s učiteľom.

Učenie s posilňovaním (RL) je ďalšou formou AI. Je odklonom od štandardných metód, nakoľko sa pri učení neriadi chybovým ukazovateľom, ale získanou odmenou. Koncept je založený na vzájomnej interakcii s riešeným problémom. Priebeh učenia je teda ľahšie interpretovateľný na základe skúmania akcií a reakcií.

Algoritmy založené na RL dokážu nájsť uplatnenie v rôznych simuláciách alebo hrách. V tejto práci sa budeme zaoberať problémom hier a hľadania vhodných stratégií na ich riešenie. Predstavíme viacero vhodných metodík na riešenie problémov, avšak hlavne metodiku aktér-kritik a jej rozšírenia.

Hry, ktoré musia algoritmy riešiť, sú jednoduchšie, ale aj zložitejšie. Pri hraní týchto hier bude skúmaný vplyv nastavenia parametrov na výsledky riešenia. Pomocou modulárnych vylepšení sa pokúsime o zlepšenie týchto výsledkov robením experimentov.

Aplikácia obsahuje viacero naimplementovaných agentov a prostredí. Je možné rozšírenie o nové typy agentov, respektíve o riešené problémy alebo simulácie. V aplikácii je možné trénovať týchto agentov, nastavovať im parametre, meniť prostredia alebo prezentovať ich dosiahnuté výsledky.

Na základe experimentov a empirických testovaní bol vytvorený vylepšený model DDPG. Tento model je otestovaný oproti štandardným modelom a je aplikovaný na prostredí s veľkou dimenziou akcií a stavov.

Kľúčové slová: Učenie posilňovaním, Umelá inteligencia, Neurónové siete, DDPG

ABSTRAKT V CUDZOM JAZYKU

Artificial intelligence (AI) is not an unknown term in the field of informatics. Companies around the world are aware of benefits of using AI. Currently, companies using AI in several forms, for example in the form of supervised learning, unsupervised learning or semi supervised learning.

Reinforcement learning (RL) it other form of AI. It is a bit different from other methods, because learning is not based on error indicators, but on the reward. Concept RL is based on interaction with problem. Therefore the learning process is easier to interpret by follow-up actions and reactions.

RL algorithms can be used in different types of simulation or games. In this paper, we are going deal with the problem of games and finding properly strategies to solve them. We introduce several problem-solving methodologies, but mainly the actor-critic methodology and its extensions.

Games presented in application are two types – simple and more complex. Algorithms goal is to solve them. During playing games, we examine the impact of parameters settings on the solutions result. We try to improve agents results by taking experiments on modular enhancements connected to them.

Application contains several agents and environments. Application is possible to be extended by new agents or environments, respectively by other problems or simulations. In application it is possible to train these agents, change its parameters, switch environments or present trained agents results in environments.

Based on results of experiments, enhanced model DDPG was created. The model is tested against standard models and it is applied on high – dimensional environment.

Key words: Reinforcement learning, Artificial intelligence, Neural Network, DDPG

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam skratiek

RL (angl. Reinforcement learning) Učenie posilňovaním

NN (angl. Neural network) Neurónová sieť

AI (angl. Artificial Intelligence) Umelá inteligencia

MDP (angl. Markov Decision proces) Markov rozhodovací problém

HW (angl. Hardware) Hardvér

(D) Diskrétny agent alebo prostredie

(C) Spojitý agent alebo prostredie

G Agent

E (angl. Enviroment) Prostredie

s Stav

S Množina stavov

a Akcia

A Množina akcií

1 Úvod

Umelá inteligencia je v dnešnej dobe veľmi populárnou oblasťou. Zaujíma sa o ňu čoraz viacej ľudí, a to nie len z informatickej praxe. Čo sa vlastne skrýva pod týmto pojmom? Inteligenciu si môžeme prestaviť ako určitú mentálnu schopnosť riešiť komplexné úlohy. Základným predpokladom pre získanie inteligencie je schopnosť učiť sa.

Umelosť inteligencie dodáva oblasť teórie počítačových systémov. Systémy, ktoré sú schopné naučiť sa vykonávať istú činnosť, ktorá si vyžaduje ľudskú inteligenciu, označujeme ako inteligentné. Spôsob, akým stroj získava vedomosti je kľúčový. Tak ako aj u človeka, inteligencia sa nedá nadobudnúť narodením, ale musí sa získať učením. Práve podľa princípov učenia rozlišujeme v AI viacero oblastí.

Hlavným delením AI je rozdelenie na učenie s učiteľom a bez učiteľa. Pri učení s učiteľom je prítomný učiteľ, ktorého úlohou je určiť, či je odpoveď stroja na daný vstup dobrá alebo zlá. Tento spôsob sa využíva v oblastiach s veľkou databázou vzorov, ktoré sú označené správnymi výsledkami. Naopak, pri učení bez učiteľa nie sú prítomné žiadne výsledky k vstupom. Stroj teda musí nájsť vzťahy v dátach bez učiteľa. Využíva sa v oblastiach, kde neexistuje jednoznačné priradenie výsledku vzorom alebo toto priradenie by bolo nákladné a zdĺhavé vytvoriť.

Spojením týchto dvoch foriem vnikla oblasť čiastočného učenia s učiteľom. Vychádza z predpokladu, že človek pri určovaní správnych výsledkov môže prehliadnuť určité vzťahy v dátach. Preto sa pri učení využije časť označených vzorov a časť neoznačených vzorov. Stroj dokáže v niektorých prípadoch nájsť aj skryté prepojenia v dátach.

Všetky vyššie spomenuté oblasti potrebujú pred učením získať základňu dát, na ktorých sa následne môže stroj učiť. Tieto dáta musia byť dopredu mnohokrát zložito spracované alebo označované. Ďalším problémom je nárast komplexnosti problémov, čo rapídne sťažuje a predlžuje proces učenia. Nie len tieto problémy podmienili vytvorenie ďalšieho samostatného smeru – učenia s posilňovaním – ktorý je predmetom tejto práce.

Učenie s posilňovaním (RL) nepotrebuje mať učiteľa, dokonca ani len dáta (pred spustením učenia). Učenie prebieha na základe odmeny alebo trestu za vykonanú akciu. Spôsob učenia je interaktívny a preto sa využíva najmä v oblastiach simulácie alebo počítačových hrách.

1.1 Ciele práce

Cieľom práce je navrhnutie aplikácie a implementovanie viacerých agentov, ktorí budú schopní sa pomocou učenia s posilňovaní naučiť vlastnú stratégiu pre hranie hier. V aplikácii bude možné nastaviť agentom parametre, pomocou ktorých môže užívateľ riešiť viacero dostupných prostredí. Agenti budú otestovaní na niektorých prostrediach a ich naučené stratégie budú porovnané s inými porovnávacími modelmi podľa určených kritérií a hraných hier. Tento cieľ sa dá rozložiť do štyroch častí.

Prvou časťou je navrhnutie a implementácia samotných agentov, ktorý dokážu využiť rôzne metodiky prístupu k učeniu s posilňovaním. Pre spojité aj diskrétne prostredie bude navrhnutých viacero agentov.

Druhou časťou je testovanie agenta na viacerých prostrediach, v ktorých bude prezentovať svoje schopnosti. Agentove výsledky sa porovnajú s dosiahnutými výsledkami iných agentov v týchto hrách.

Treťou časťou je implementácia aplikačného rozhrania pre výber agentov a prostredí s možnosťou upravenia hyper-parametrov agentov. Pre demonštrovanie výsledkov bude aplikácia obsahovať porovnávacie kritéria a zobrazovať sa budú vo forme grafov.

Nakoniec budú na agentoch vykonané experimenty a modifikácie pre zlepšenie ich učiacich schopností. Zmenené budú rôzne parametre, vnútorné nastavenia a upravená metodika učenia sa. Pomocou týchto nastavení sa pokúsime o zrýchlenie procesu učenia a zlepšenie dosiahnutých výsledkov. Na základe výsledkov experimentov bude vytvorený vylepšený model, ktorý bude otestovaný na prostrediach oproti pôvodným modelom. Nakoniec bude aplikovaný na väčšie spojité prostredie, ktoré je z hľadiska nájdenia riešenia časovo aj výpočtovo zložitejšie.

2 Analýza histórie a súčasného stavu

2.1 História

Vývoj umelej inteligencie ako takej siaha až do roku 1949, v ktorom kanadský psychológ Donald Hebb popísal svojou teóriu učenia ako proces, v ktorom pri učení vznikajú alebo zanikajú, respektíve sa zosilňujú spojenia medzi neurónmi[1]. Neskôr na jeho teóriu nadviazal v roku 1957 Frank Rosenblatt popísaním umelého perceptónu [2].

Po týchto kľúčových objavoch sa v oblasti umelej inteligencie sa začali formovať viaceré smery. V nasledujúcich rokoch sa vyformovali do samotných oblastí, podľa toho, aký spôsob učenia využili alebo aký problém riešili. Jednalo sa hlavne o problémy spracovania jazyka, dolovania dát, robotiky ale aj forma predikcií z dát, napríklad na burze.

Popri týchto hlavných smeroch sa postupom času v snahe reflektovať a spracovávať impulzy v reálnom čase vyvíjal koncept RL. V počiatkoch sa v tejto oblasti vyformovali tri smery. Prvým smerom bolo učenie pokus – omyl. Podklady pre tento smer sa čerpali v psychológii. Ani nie však v tej ľudskej, ako zvieracej. Podľa amerického psychológa R. S. Woodwortha myšlienka metódy pokus – omyl siaha do roku 1894, kedy psychológ C.L. Morgan použil tento termín na popísanie správania zvierat. Najstručnejšie vyjadril tento termín vo svojom článku v roku 1911 Thorndik. Hovorí v ňom, že pokiaľ je nesprávne rozhodnutie zvieraťa nasledované trestom, väzba k týmto rozhodnutiam slabne, a naopak, keď je správne rozhodnutie nasledované odmenou, väzba silnie. Tým pádom u zvieraťa sa zmenšuje pravdepodobnosť výberu zlých rozhodnutí. Jeho vyjadrenie je známe ako zákon účinku (angl. Law of Effect) [101].

Ďalším smerom bol smer optimálnej kontroly. Kľúčovým míľnikom bolo rozšírenie Hamilton – Jacobi teórie z 19. storočia R. Bellmanom v roku 1950. Bellman použil koncept dynamického systémového stavu a funkcii ohodnotenia (angl. Value function) pre definovanie rovnice známej ako Bellamova rovnica. Metódy, ktoré riešia túto rovnicu sú známe ako dynamické programovanie. Ďalej v roku 1957 prestavil diskrétnu a stochastickú verziu problému optimálnej kontroly známu ako Markov rozhodovací proces (MDP), z ktorej odvodil R. Howard iteračnú metódu optimalizácie politiky. Tieto teórie boli základnými podkladmi moderného RL [101].

Treťou oblasťou je oblasť učenia pomocou časového oneskorenia (angl. Temporal difference learning). Pôvod taktiež siaha do psychológie zvieraťa, avšak sa nezaoberá len primárnym oceňovaním (jedlo, bolesť), ale aj sekundárnym oceňovaním. V roku 1959 použil túto metódu v jeho programe na hranie dámy [101].

Na pár rokov sa celá oblasť RL odmlčala. Spomenuté oblasti sa približne do roku 1980 spojili a vytvorili tak modernú formu RL. Po týchto rokoch sa stala oblasť RL opäť atraktívnou. Zaslúžili sa o to najmä Sutton a Barto. V roku 1982 rozvinuli Klopfovu teóriu lokálnej odmeny z roku 1972 a popísali psychologický model založený na časovom oneskorení. Neskôr v roku 1996 D. Bertsekas a J. Tsitsiklis vytvorili termín neurodynamické programovanie, ktoré označovalo spojenie dynamického programovania a neurónových sietí.

2.2 Súčasný stav

V súčasnosti, najmä po myšlienke spojenia oblasti RL s neurónovými sieťami, dovoľuje AI riešiť problémy, ktoré dovtedy nebolo možné riešiť pre ich komplexnosť a rozmery. Neurónové siete v RL zohrávajú úlohu aproximátora funkcií, prípadne úlohu rozpoznávania obrazového vstupu. Spojenie so spracovaním obrazového vstupu umožňuje aplikovanie RL do širokej škály problémov. Keďže je RL založené na odmeňovaní, jeho hlavný ukazovateľ je získaná odmena. Preto je najvhodnejšie prezentovanie RL algoritmov pomocou hry, či už počítačovej alebo inej, prípadne simulácie.

Prelom nastal v roku 1997, kedy algoritmus IBM Deep blue porazil v šachu majstra sveta Garry Kasparova. Kasparov bol porazený vo viacerých šachových partiách. Celá udalosť bola aj sfilmovaná s titulkom „The Man vs. The Machine“. Tento triumf odštartoval viacero tímov plných nadšencov o ďalšie prekonanie ľudských rekordov[17].

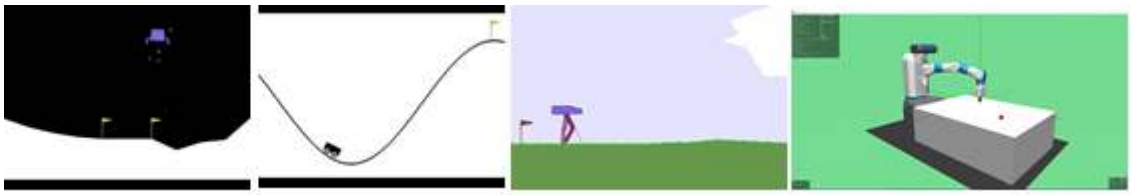
Ďalšie výrazné úspechy v prekonaní človeka prišli o niekoľko rokov neskôr. V roku 2015 zvíťazili algoritmus AlphaGo, ktorý je zložený z neurónových sietí kombináciou učenia s učiteľom a RL. Algoritmus porazil svetového šampióna v hre Go Lee Sedola[18]. O rok neskôr porazil počítač IBM Watson v hre riskuj! (angl. Jeopardy!) [19] najlepších hráčov v tejto hre – Ken Jenningsa a Brad Ruttera. Počítač používal robustné moduly pre predspracovanie dát a princípy RL v rozhodovaní [20].

Obrázok 1 AplhaGo vs majster sveta -Lee Sedol[18], IBM Watson v hre Jeopardy![19], IMB Deep blue vs majster sveta Gari Kasparov[17]

Tím IBM Blue v týchto rokoch aplikoval RL algoritmus na učenie sa hier Atari 2600. Algoritmom na princípe agenta DQN sa podarilo dosiahnuť nadľudský level v hraní týchto hier. Učenie prebiehalo na základe spracovania vysoko dimenzionálneho vstupu vo forme RGB obrázku a odmeny [20]. Tieto hry sa rýchlo stali populárnymi pre všetkých, čo sa zaujímali o oblasť RL. Sada obsahuje 526 hier (prispôsobených dnešným PC), ktoré sú ovládané pomocou hernej konzoly. Prostredie sprostredkúva Arcade Learning Enviroment [21] a je možné v ňom trénovať svojich agentov.

Obrázok 2 Niekoľko príkladov Atari 2600 hier (Breakout, Space invaders, Seaquest, Bream rider) [7]

Približne v roku 2017 bola vytvorená platforma nových problémov. Tieto problémy sú dostupné v lepšom rozlíšení ako hry Atari a aj ich ovládanie je zložitejšie. Platforma sa volá OpenAi Gym a poskytuje viacero hier, od algoritmických po grafické riešenie diskrétnych, ako aj zložitých spojitých problémov. Pomocou tejto platformy budú trénovaní a testovaní agenti v tejto práci.



Obrázok 3 Príklady hier OpenAi (Lunar Lander, Mountain Hill, Bipedal Walker, Robotics)[22]

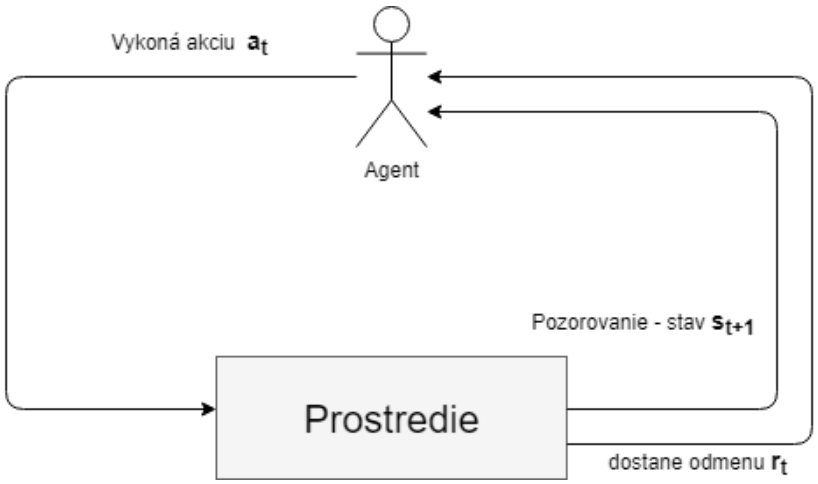
3 Metodológia učenia s posilňovaním

Učenie s posilňovaním je jeden z možných spôsobom učenia pri umelej inteligencii. Tak ako aj neurónové siete ako také boli konštruované na odpozorovaní fungovania ľudského mozgu, aj RL je spätý so živými tvormi. Ako už bolo spomenuté, ani nie tak s človekom, ako skôr so zvieraťom.

Je to pre to, že v tomto smere nájdeme aj psychologické podklady, ktoré sú najmä z oblasti „behavioralizmu“. Táto oblasť neuznáva vedomie ako predmet výskumu, ale zaujíma sa o rozličné formy správania, ktoré sa prejavujú ako reakcie organizmu na vonkajšie impulzy [8]. Celá myšlienka sa dá najlepšie demonštrovať na príklade výcviku psa. Pri trénovaní povelu „sadni“ psa oceníme po správnom vykonaní príkazu a naopak, pri zlom vykonaní povelu ho potrestáme. Ak je oceňovanie účelné, psovi sa zosilní väzba ku správnemu vykonávaniu príkazu a na druhej strane zoslabne, k nesprávnemu konaniu na príkaz. Po čase sa pes povel naučí [101].

3.1 Model učenia posilňovaním

Podľa spomenutého príkladu funguje aj učenie posilňovaním. Kľúčovú rolu v ňom zohráva ***agent***, ktorý je umiestnený do prostredia. Agent interaguje s prostredím na základe vykonania akcie ***a***. Akcia môže, ale nemusí zmeniť stav prostredia. Agent si vyberá, ktorú akciu vykoná na základe zvolenej politiky $\boldsymbol{\pi}$. Po každom vykonaní akcie je agent odmeňovaný alebo trestaný skalárnou veličinou ***r*** na základe toho, či sa rozhodol z hľadiska daného prostredia správne alebo nesprávne. Celý proces popisuje obrázok nižšie.



Obrázok 4 Konceptuálny model RL

Cieľom celého procesu je vďaka kladným a záporným odmenám zo spätných väzieb prostredia naučiť agenta stratégiu, ktorá maximalizuje množstvo získanej odmeny. Do učenia nezasahuje chyba ako ukazovateľ, pretože dosiahnutá odmena je hlavným ukazovateľom kvality riešenia.

3.2 Markovov rozhodovací proces

Celý vyššie spomenutý konceptuálny model RL môžeme zapísať ako Markov rozhodovací problém (MPD). V modely má prostredie Markovovskú vlastnosť, čo znamená, že reakcia prostredia v čase $t + 1$ je závislá len na stave s_t. Markovov rozhodovací proces je popísaný nasledujúcimi podmienkami [10]:

- $\boldsymbol{S}$ je konečná množina stavov, kde stav $\boldsymbol{s_0} \in \boldsymbol{S}$ je počiatočný stav
- Pre každý stav $\boldsymbol{s} \in \boldsymbol{S}$ existuje konečný počet akcií $\boldsymbol{a} \in \boldsymbol{A_s}$, ktoré môže agent vykonať
- $\boldsymbol{R(s, a)}$ je ohodnocujúca funkcia, ktorá ohodnocuje agentove rozhodnutie v stave *s* očakávanou hodnotou ohodnotenia pri výbere akcie $\boldsymbol{a}$
- $\boldsymbol{P(s_t, a_t s_{t+1})}$ je prechodová funkcia, ktorá určuje, s akou pravdepodobnosťou <0,1> je možné prejsť zo stavu $\boldsymbol{s_t}$ do stavu $\boldsymbol{s_{t+1}}$

Hlavným cieľom MDP je nájdenie vhodnej stratégie pre agenta. Táto stratégia je funkcia, ktorá sa označuje $\boldsymbol{\pi(s)}$ a vyjadruje, ktorú akciu zvolí agent v stave $\boldsymbol{s}$. Výsledné aplikovanie stratégie na postupnosť stavov vytvorí Markovovský reťazec. Celá stratégia podlieha kritériu maximalizácie kumulatívneho ohodnotenia [10].

3.3 Diskrétne a spojité rozhodovanie

Prístup k riešeniu sa môže líšiť podľa toho, aký typ problému máme k dispozícii, respektíve akým spôsobom ho implementuje prostredie. Z pohľadu konania agenta v prostredí rozlišujeme prostredia, ktoré poskytujú diskrétny alebo spojitý priestor pre akcie, ktoré môže agent vykonať.

Zoberme si ako príklad hru hadíka. V tejto hre hráč ovláda hadíka, s ktorým sa snaží zjesť všetkú potravu, ktorá sa mu v levely zobrazí. Musí tak urobiť bez toho, aby narazil do steny alebo seba samého. Možné pohyby hadíka sú dopredu, doľava a doprava.

Na jednej strane sa môžeme na problém pozerať tak, že priestor akcií je diskrétny, a teda hadík sa môže pohnúť len spomínanými smermi. To znamená, že existujú len tri možnosti výberu akcie, zapísaných vektorovo [0, 1, 0], [1, 0, 0], [0, 0, 1] (dopredu, vľavo, vpravo).

Na druhej strane môžeme uvažovať o spojitom priestore akcií. Napríklad, akcie budú ohodnotené reálnym číslom, reprezentujúcim silu prechodu do daného stavu. Oproti predošlému prípadu sa tak priestor akcií zväčší na nekonečne veľa možností prechodov. Príkladom akcie môže byť [0.7, 0.5, 0.3] čo môže byť (po odčítaní protichodných hodnôt) šikmý posun o 0.7 dopredu a 0.2 smerom doľava.

Pri spojitom rozhodovaní však treba uvažovať aj o tom, že konečná množina stavov prostredia sa môže rozšíriť na nekonečnú. Je to práve preto, že kvôli spojitému priestoru akcií sa agent môže dostať do takých stavov, ktoré nie sú v diskrétnom prostredí pri použití diskrétnych akcií dosiahnuteľné.

3.4 Stav

Stav predstavuje aktuálnu situáciu agenta v prostredí. Môže mať viacero reprezentácií, ktoré závisia od definovania problému a možností agenta spracovávať tieto informácie. Označuje sa písmenom s.

Špecifickou definíciou stavu je vektorová reprezentácia. V týchto prípadoch musí byť agent informovaný hlavne o veľkosti stavového vektora, ktorý dokonca môže byť aj viacrozmerný. Pri výbere tejto možnosti je nutná identifikácia atribútov, ktoré majú najväčšiu vypovedaciu hodnotu ohľadom stavu v danom prostredí. Napríklad v spomenutej hre hadík by boli kľúčovými atribútmi pozícia hada, pozícia potravy a pozícia nebezpečenstva v podobe vlastného tela alebo prekážky.

Univerzálnou reprezentáciou stavu je snímkovanie prostredia. To znamená, že stav bude reprezentovaný fotkou v určenom rozmere a farebnej škále. Táto reprezentácia najviac pripomína ľudského hráča, nakoľko umelá inteligencia dostáva vstupy také, aké vidí aj človek. V tomto prípade je na agenta nutnou požiadavkou naučiť sa v prvom rade extrahovať potrebné informácie z obrázkových vstupov.

3.5 Akcia

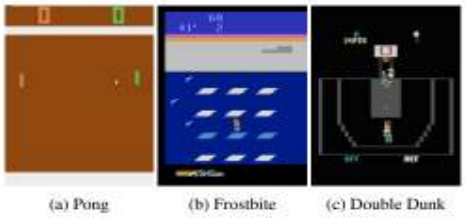
Akcia je komunikačným predmetom interakcie agenta a prostredia. Na pojem akcia sa viaže pojem akčný priestor. Je to konečná množina všetkých akcií, ktoré môže agent vykonať. Akcia sa označuje písmenom **a**, celý priestor akcií **A.** Pri výbere akcií v diskrétnom prostredí sa vyberá jedna akcia, napríklad pohyb hore. V spojitom prostredí agent určuje reálnu hodnotu každej zložky akcie (0.3 dopredu, 0.2 vľavo). Typ akcií závisí od zvoleného prostredia.

3.6 Prostredie

Prostredie predstavuje simulačný priestor. Niekedy je označované aj ako simulačný model. Kvalita jeho implementácie je kľúčová, nakoľko od nej sa odvíja kvalita dosiahnutých výsledkov agenta. Príkladom prostredia môže byť simulácia (model mestských semaforov alebo vyťaženia križovatiek), časový rad alebo ukazovatele finančného portfólia (obchodovanie na burze), určité hry (AlphaGo, šach) alebo najčastejšie počítačové hry.

Prostredie môže byť dvojaké. Môže byť plne preskúmané, a teda agent vidí všetky veličiny prostredia. Prostredie je agentovi úplne známe a predávaný stav agentovi obsahuje všetky skutočnosti o prostredí.

Častejšie sa však stretávame s prostredím, ktoré nie je úplne preskúmané. Takým prostredím môže byť najjednoduchším príkladom hra, v ktorej sa vyskytujú strely alebo lopta, a stav nie je prezentovaný vektorom, ale obrázkom. V týchto prípadoch z obrázku nevieme (ani len ľudia) vyčítať smer lopty alebo strely, a teda v reprezentácii stavu nám chýba informácia o aktuálnom smere lopty, ktorú sa dozvieme až z ďalšej snímky podľa zmeny pozície lopty.



Obrázok 5 Prostredia, ktoré nie sú plne preskúmané (využívajúce snímkovanie stavu)

3.7 Agent

Agent je základnou akčnou jednotkou RL. Vopred má určený alebo zadaný cieľ, napríklad maximalizácia odmeny. Jeho vstupnou časťou, komunikačným modulom, je senzor. Ten musí byť prispôsobený forme stavov, ktorými bude komunikovať s prostredím (vektor, obrázok).

Agent nemusí byť v prostredí sám. V prostredí môže figurovať viacero agentov, pri čom každý z nich môže byť zodpovedný za jednu časť alebo môžu spoločne vysielať akcie prostrediu na vykonanie [12]. Príklad využitia môže byť v učení tímových taktík, napríklad pri simuláciách kolektívnych športov.

3.8 Odmena

V RL je spôsob odmeňovania agenta veľmi dôležitý. Správne zvolenie taktiky odmien a trestov môže mať rázny dopad na vývoj tréningu agenta. Ocenenie je skalárna veličina, ktorú získava agent za svoje rozhodnutia. Označuje sa písmenom **r.** Hodnota odmeny býva kladná alebo záporná, kde veľkosť odmeny závisí od kvality rozhodnutia agenta.

Agent sa snaží o maximalizovanie kumulatívnej odmeny, a teda odmena alebo trest je prostriedok, ktorým sa ovplyvňuje spôsob a rýchlosť agentovho učenia. Preto je nutné uvážiť, či treba agenta oceňovať po každom jeho rozhodnutí, alebo použiť iný systém oceňovania. Totižto, ak je agent oceňovaný kladne za každý jeden krok, ktorý urobí, agent sa môže uprieť na istú (no síce malú) odmenu každý krokom. Vtedy agent už ďalej nebude skúmať prostredie a skúšať vylepšovať svoju stratégiu, ale bude v istých cykloch zbierať len tieto malé odmeny okolo stavu, kde sa nachádza. Agent túto stratégiu vyhodnotí ako najvhodnejšiu a tým pádom sa vyhne možnosti vstúpiť do nepreskúmanej oblasti, aby riskoval veľkú stratu odmeny alebo prípadné získanie vysokej odmeny.

Všeobecne vhodnejšou stratégiou môže byť postupné a veľmi malé trestanie agenta za nesplnenie globálneho cieľa, napríklad prejdenie daného levelu. Vtedy agent bude nútený zvoliť stratégiu tak, aby dosiahol stanovený cieľ a vyrovnal tak škodu na veľkosti odmeny počas procesu dosahovania cieľa.

3.9 Politika

Politika je funkciou, ktorá definuje spôsob správania agenta v danom stave. Môže to byť funkcia deterministická alebo stochastická. Deterministická politika vyberá len jednu akciu podľa zvoleného kritéria, pričom stochastická politika počíta pravdepodobnosť výhodnosti prechodu cez všetky akcie. Najznámejšími kritériami sú chamtivá politika (angl. greedy) politika (výber najvyššej Q - hodnoty akcie). Pri odhadovaní politiky pomocou NN sú za politiku považované vnútorné naučené parametre NN. Využitie NN je všeobecnejšie a lepšie využiteľné, nakoľko dovoľuje aj čiastkovú stratu odmeny pri hľadaní cieľa. Politika sa označuje $\boldsymbol{\pi}$.

Dôležitým pojmom pri politike je aj faktor ovplyvňujúci vplyv budúcich odmien voči súčasným. Označuje sa písmenom gama $\boldsymbol{\gamma} \in (0,1)$ a nazýva sa diskontný faktor (angl. discount factor). Tento faktor ovplyvňuje, aký vplyv v aktuálnom stave bude mať v budúcnosti očakávaná odmena. Agent s menšou hodnotou diskontného faktoru prikladá väčšiu váhu svojim aktuálnym rozhodnutiam, čo môže produkovať odchýlky od priblíženia k cieľu. Následkom môže byť to, že agent síce môže trafiť cieľ, ale môže trafiť aj úplne mimo cieľa [12]. Na druhej strane, vysoká hodnota produkuje menší rozptyl a motivuje agenta vyberať také akcie, ktoré majú najväčšiu možnú budúcu odmenu.

3.9.1 Učenie s ohľadom na politiku alebo bez

Učenie politiky môže byť dvojaké. Učenie s ohľadom na politiku (angl. on-policy) vyjadruje použitie aktuálne zvolene stratégie – politiky, podľa ktorej sa agent rozhoduje na danej trajektórii. Tieto dáta sú zozbierané pod jednou politikou, a teda pod aktuálnymi váhami neurónovej siete odhadujúcej parametre politiky.

Spôsob učenia bez ohľadu na politiku (angl. off-policy) používa pri trénovaní dáta zozbierané aj s minulých politík, čo môže v niektorých algoritmoch spôsobiť problémy. Tieto dáta sa vyznačujú tým, že sú uchovávane v priebehu času, počas ktorého sa politika mení. A teda použitie starších dát z inej politiky môže nesprávnym spôsobom zasahovať do zmeny parametrov aktuálnej politiky.

3.10 Prieskum vs využitie naučených znalostí

Kompromis medzi prieskumom a využitím naučených znalostí (angl. Exploration vs Exploitation trade-off) popisujme agentove správanie pri hľadaní vhodnej stratégie. Agent

je odmeňovaný za správne vykonanie akcií, avšak dopredu mu nie je povedené, ktoré akcie sú správne. Tento kompromis popisuje agentov vnútorný stav pri rozhodnutiach. Pojmom využitie naučených znalostí je označovaný proces, ktorý dovoľuje agentovi otestovať svoju zvolenú stratégiu, ktorú ma naučenú. Keď však agent narazí na dobré akcie, môže túto akciu vložiť do svojej stratégie, čo popisuje proces prieskumu, čím sa zmení aj aktuálna politika.

Hľadanie vhodného pomeru medzi využitím naučených znalostí a prieskumom je označovaný aj ako agentova dilema rozhodovania. Je popísaná ako jav, kedy agent pozná dobrú stratégiu, či bude aj naďalej robiť prieskumy, pri ktorých však nie je zaručené, že nájde lepšiu stratégiu.

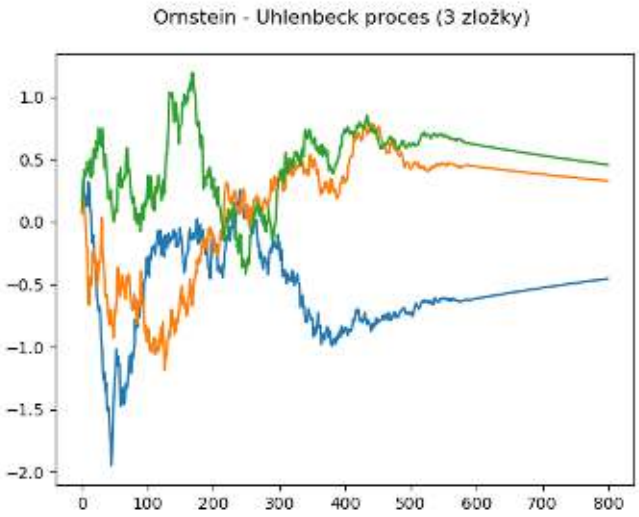
V diskrétnom prostredí je jednoduché vybrať náhodnú akciu. V spojitom prostredí to však nie je možné, pretože je nutné vybrať hodnotu každej zložky akcie. Preto sa používa ako náhodná zložka pre prieskum šum, ktorý generuje hodnoty pre také množstvo zložiek, koľko je potrebných. V tejto práci bol použitý šum generovaný z Ornstein-Uhlenbeck procesu [102]. Proces modeluje rýchlosť Brownovej častice s trením, kde hodnoty sú časovo korelované a vycentrované okolo 0.

$$dx_t = \theta(\mu - x_t)dt + \sigma dW_t \tag{1}$$

V tejto práci boli použité nasledovné parametre procesu.

$$\theta = 0.15, \mu = 0, \sigma = 1, d = 0.01\ W = Norm.rozdelenie \tag{2}$$

Generovaná hodnota závisí od času ***t***, ktorý je rovný počtu epizód. Vygenerované zložky sa pripočítajú k aktuálnemu odhadu akcií podľa stavu po zložkách.



Obrázok 6 Príklad Ornstein – Uhlenbeck procesu s chladením = 600 pre 3 zložky

3.11 Funkcia ohodnotenia vs politika

V RL rozlišujeme dve hlavné typy učenia modelov. Modely založené na funkcii ohodnotenia odhadujú hodnotu funkcie podľa stavu a jeho akcií. Čím vyššia je hodnota, tým lepšia je akcia oproti ostatným. Najznámejším využitím je Q - učenie a jeho rozšírenia. Ich sila spočíva v znovu použití dát pri učení s Q - hodnotou určujúcou, pre ktorú akciu očakávame najväčšiu možnú hodnotu v budúcnosti.

Naopak, metóda založené na politike optimalizujú politiku bez Q - hodnôt. Využitie metódy je vhodné pre spojité problémy alebo pri vysoko dimenzionálnom priestore akcií, pretože rýchlejšie konvergujú k požadovaným výsledkom a sú stabilné. Používajú ju algoritmy založené na gradiente politiky rozhodovania.

3.12 Bellmanova rovnica vs gradient politiky rozhodovania

Metódy s využitím funkcie ohodnotenia podliehajú Bellmanovej rovnici. Myšlienkou rovnice je nasledovné: Hodnota v počiatočnom stave je odmenou, ktorá je očakávaná v tomto stave, plus hodnota ďalšieho stavu, do ktorého sa agent vyberie [16]. Formulácia tejto myšlienky je rozdielna pre metódy s učením s ohľadom na politiku alebo bez.

$$Q^{\pi}(s,a) = r(s,a) + \gamma * Q^{\pi}(s',a') \tag{3}$$

$$Q^{*}(s,a) = r(s,a) + \gamma * max\,(Q^{*}(s',a')) \tag{4}$$

Rozdiel spočíva v pridaní výberu maximálnej Q-hodnoty z akcií. To znamená, že sa agent optimalizuje bez ohľadu na politiku, musí vyberať najvyššiu hodnotu z akcií, aby konal optimálne. V prípade učenia s ohľadom na politiku je jeho výber podmienený politikou, a preto koná v každom stave optimálne podľa politiky $\boldsymbol{\pi}$.

Použitie gradientovej politiky sa viaže na to, že agent sa snaží o maximalizovanie dosiahnutej odmeny. V tomto prípade, teda agent hľadá smer gradientového nárastu v danom stave. Pri využití NN ako aproximátora je gradient použitý v procese spätného šírenia chyby pri učení (angl. back-propagation).

$$\theta_{t+1} = \theta_t + \alpha * \Delta R\,(\theta_t) \tag{5}$$

Kde nové váhy NN $\boldsymbol{\theta_{t+1}}$ sú nastavované pomocou učiaceho koeficientu $\boldsymbol{\alpha}$ a gradientu.

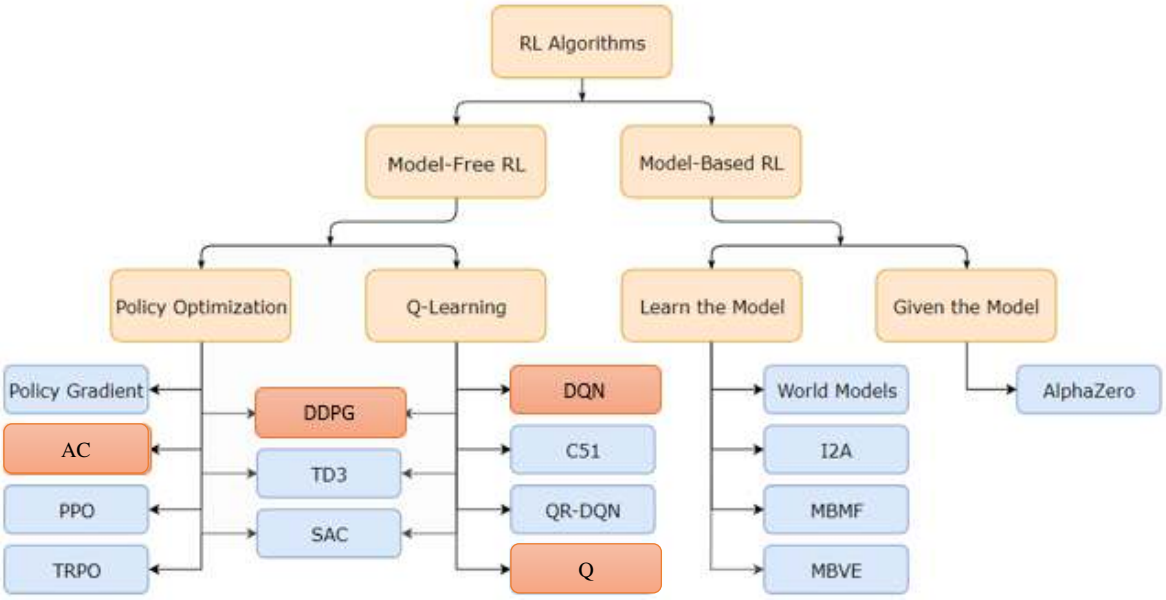
Podľa všeobecného zápisu zmena váh závisí od určenej delty, ktorá v stupuje do procesu úpravy váh NN. Rozlišujeme viacero modifikácii gradientovej politiky pri rôznych algoritmoch. Príkladom konkrétnej implementácie gradientu je nasledovná rovnica.

$$\Delta R\,(\theta_t) = r_t + \gamma * V^{\theta}(s) - V^{\theta}(s') \tag{6}$$

Rovnica popisuje gradient prístupu aktér – kritik, podľa ktorého sa optimalizuje aproximátor politiky na základe diskontovanej hodnoty v aktuálnom stave $\boldsymbol{V^{\theta}(s)}$ zníženú o hodnotu nasledujúceho stavu $\boldsymbol{V^{\theta}(s')}$**.**

4 Metódy riešenia

K riešeniu problému RL existuje viacero prístupov. Obrázok nižšie zobrazuje niektoré z nich v stromovej hierarchii. Hierarchia neobsahuje niekoľko ďalších prístupov, ako napríklad transferové alebo meta učenie. Zobrazuje však určitý spôsob členenia najpoužívanejších prístupov, ktoré od seba odlišuje vo viacerých pohľadoch.



Obrázok 7 Pohľad na metodiky riešenia - upravené (Zdroj: [14])

Kľúčovým členením je delenie podľa toho, či prístup obsahuje alebo má prístup k modelu prostredia. Ako model prostredia si môžeme predstaviť funkciu, ktorá predpovedá prechody medzi stavmi a odmenami prechodov.

V prípade, že agent má dostupný model prostredia, môže rozmýšľať dopredu. Agent vyskúša viacero akcií a na základe explicitného vyhodnotenia ich výhodnosti sa rozhodne, ktorú si vyberie. Tento spôsob predstavuje určité plánovanie agenta. Najznámejším využitím tohto prístupu je agent AlphaZero [14].

Vo väčšine prípadov agent nemá prístup k modelu prostredia. Avšak učenie modelov je častokrát veľmi náročné a nemusí sa vyplatiť. Agenti bez prístupu k modelu prostredia sú jednoduchšie implementovateľný a ľahko vylepšiteľný, vďaka čomu je oblasť bez modelov populárnejšia a viacej rozvinutá.

Ďalším stupňom rozlíšenia v bez modelových prístupoch je delenie, podľa toho, čo je predmetom učenia. Optimalizácia politiky a učenie na základe Q - hodnoty sú metódy, ktoré využívajú agenti bez modelov. Metóda optimalizácie politiky využíva explicitné

vyjadrenie politiky. Optimalizuje parametre funkcie $\boldsymbol{\pi_\theta}(\boldsymbol{a}|\boldsymbol{s})$ priamo výsledkom gradientového nárastu. Pri tejto optimalizácii sa využíva aproximátor $\boldsymbol{V_\Phi}(\boldsymbol{s})$ funkcie $\boldsymbol{V^\pi}(\boldsymbol{s})$, ktorý určuje, ako sa zmení politika. Optimalizácia vo väčšine prípadov býva s ohľadom na politiku, to znamená, že pre každú aktualizáciu sa používajú len dáta zozbierané počas najnovšej verzii politiky. Zjednodušenie po jednej epizóde. Tieto metódy sú stabilné a spoľahlivé.

Typickou črtou učenia na základe Q - hodnoty je parametrizovaný aproximátor $\boldsymbol{Q_\theta}(\boldsymbol{a}|\boldsymbol{s})$, ktorý sa používa na optimalizovanie funkcie $\boldsymbol{Q^*}(\boldsymbol{a}|\boldsymbol{s})$. Tieto metódy využívajú funkciu ohodnotenia založenú na Bellmanovej rovnici spomenutej vyššie. Odhadovanie parametrov aproximátora prebieha bez ohľadu na politiku, čo znamená, že pre aktualizáciu parametrov sa môžu použiť dáta bez ohľadu na to, akou politikou boli získané. Tieto metódy sú však menej stabilné, kvôli nepriamym zmenám agentovho výkonu.

Prístupy založené na modeloch sú ťažšie rozlíšiteľné. Pre ilustráciu boli vybraté dva typy, naučiť sa model a využiť naučený model. V prvom prípade sa jedná o plánovanie. Agent v každom okamihu si určí trajektóriu k riešeniu. Po vykonaní akcie znovu prepočíta celú trajektóriu. Druhým prípadom sú tzv. „Expertné modely“. Využívajú plánovanie, ktoré však vychádza z aktuálnej politiky a preskúmava len kandidátov na akcie v danom stave.

V ďalšej časti tejto práce budú priblížené označené metódy (Obrázok č. 7), ktoré sú využité v tejto práci.

4.1 Q - učenie

Algoritmus je jedným zo základných algoritmov RL. Agent vykonáva akcie prislúchajúce jeho stavu, ktoré majú najväčšiu Q-hodnotu. Dáta sú uložené formou tabuľky, kde sa uchovávajú stavy a Q-hodnoty príslušných akcií. Po vykonaní akcie $\boldsymbol{a}$ si agent uloží stav $\boldsymbol{s}$ do tabuľky a na základe odmeny $\boldsymbol{r}$ podľa predpisu

$$Q^{tab}(s,a) = r + \gamma * Q^{tab}(s,a) \tag{7}$$

aktualizuje Q-hodnotu vybranej akcie diskontovanú faktorom $\boldsymbol{\gamma}$. Pri rozhodnutí agent v prvom rade kontroluje, či daný stav je v jeho tabuľke prítomný. V prípade, že je prítomný, vyberie akciu s maximálnou Q-hodnotou z hodnôt príslušných akcií. Ak nie je prítomný, vtedy si agent uloží stav do svojej tabuľky a rozhodne sa ďalej náhodne. Algoritmus je

náchylný na veľké množstvo stavov v prostredí, pretože je obmedzený veľkosťou tabuľky, a teda HW, ktorý je k dispozícii.

4.2 DQN

4.2.1 Rozšírenie Q - učenia na DQN

Skratka pochádza z anglického názvu Deep Q Neural Network. Je kombináciou hlbokého učenia a RL. Oproti Q-učeniu je hlavná zmena vo forme uloženia dát. V Q-učení sa ukladajú stavy spolu s akciami do tabuľky, kde sa aj aktualizujú Q-hodnoty. DQN agent používa na odhad Q-hodnôt akcií neurónovú sieť.

Pre zvýšenie stability učenia využíva agent skúsenosti. Skúsenosti sú ukladané vo forme stav, akcia, odmena a nasledujúci stav. Tieto údaje sú po ukončení epizódy (alebo väčšieho intervalu) použité na pripomenutie akcií z minulosti, ktorými sa dotrénuje neurónová sieť. Aby sa predišlo korelácii v skúsenostiach alebo rýchlemu zabúdaniu, používajú sa náhodné dávky z dát.

Rozšírenie Q-učenia na DQN síce rieši problém s dimenziou stavov, ale stále je obmedzený na diskrétny a nízko rozmerný priestor akcií. DQN nemôže byť bez úpravy použitý na spojité problémy, pretože hľadá akciu, ktorá má maximálnu Q-hodnotu, čo v spojitom prostredí nie je možné. V spojitom prostredí je nutné vybrať hodnoty pre všetky zložky priestoru akcií.

Učenie prebieha na základe predpisu

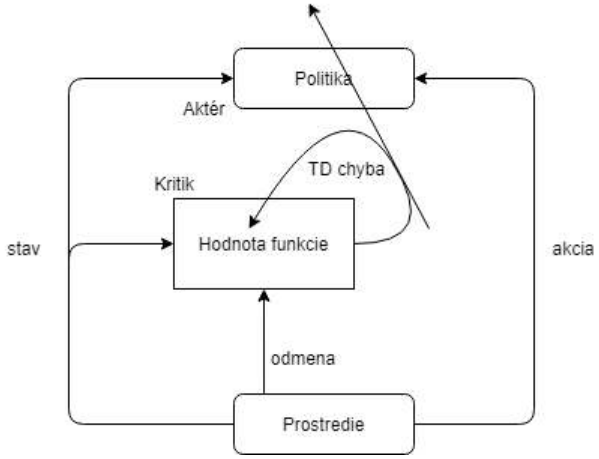
$$Q_\theta(s_t, a) = r + \gamma * \max Q_\theta\,(s_{t+1}, a) \tag{8}$$

kde je výsledná Q – hodnota stavu $\boldsymbol{s_t}$ určená odmenou $\boldsymbol{r}$ v danom stave v súčte s diskontovanou Q – hodnotou nasledujúceho stavu $\boldsymbol{s_{t+1}}$ faktorom $\boldsymbol{\gamma}$. V každom učiacom cykle je použitá dávka skúseností, na ktorú sa aplikuje spomenutý predpis. Podľa spomenutej Bellmanovej rovnice, tým že agent vyberá maximálnu Q – hodnotu z nasledujúcich možných akcií, koná optimálne bez ohľadu na politiku.

4.3 Aktér – kritik

Hlavná myšlienka tejto metodiky pozostáva zo spojenia dvoch hlavných postupov v RL, ktoré sú založené na funkcii ohodnotenia alebo funkcie politiky. Spojenie je kľúčové

pre oblasť moderného učenia s posilňovaním. Došlo k nemu tým, že sa hlavný model agenta rozdelil do dvoch komponentov. Základný koncept učenia zobrazuje nasledovný obrázok.



Obrázok 8 Aktér kritik koncept [101]

Prvým komponentom je aktér. Jeho úlohou je vyberať akcie v danom stave. Predstavuje agentove myslenie, respektíve má kontrolu nad jeho správaním pomocou funkcie politiky. Aktér vyberá najlepšiu akciu zo stavu, v ktorom sa nachádza. Učenie prebieha smerom, ktorý určí kritik, podľa

$$\theta = \theta + \alpha * V_{\omega}(s') * \ln \pi_{\theta}(a \mid s) \tag{9}$$

kde sa synaptické váhy aktéra $\boldsymbol{\theta}$ prispôsobujú učiacim cyklom NN pomocou hodnoty nasledujúceho stavu $\boldsymbol{V_{\omega}(s')}$ a logaritmovanou hodnotou akcie $\boldsymbol{a}$ v aktuálnom stave $\boldsymbol{s}$.

Druhý komponent testuje aktérom zvolené akcie a počíta funkciu ohodnotenia. Označuje sa ako kritik. Vypočítava hodnotu v funkcie ohodnotenia v čase, pomocou ktorej optimalizuje parametre politiky výberu akcií aktéra. Učenie prebieha podľa predpisu

$$\omega_{*} = \omega + \beta * \delta_t * V_{\omega}(s) \tag{10}$$

kde sú synaptické váhy kritika $\boldsymbol{\omega}$ prispôsobované časovo oneskorením $\boldsymbol{\delta_t}$ násobeným hodnotou aktuálneho stavu $\boldsymbol{V_{\omega}(s)}$.

Oba komponenty môžu byť prezentované NN modelom, ktoré môžu zdieľať parametre. Ich učenie prebieha podľa časového oneskorenia (angl. Temporal difference)[101] počas vykonávania akcií v epizóde podľa predpisu

$$\delta_t = r_t + \gamma * V_{\omega}(s') - V_{\omega}(s) \tag{11}$$

kde hodnota časového oneskorenia $\boldsymbol{\delta_t}$ v čase $\boldsymbol{t}$ je určená odmenou $\boldsymbol{r_t}$ a diskontovanou hodnotou nasledujúceho stavu $\boldsymbol{V_\omega(s')}$ pomocou $\boldsymbol{\gamma}$ zníženou o hodnotu predošlého stavu $\boldsymbol{V_\omega(s)}$.

4.4 DDPG

Rozšírením DQN na spojitý priestor vznikol koncept DDPG. Skratka pochádza z anglického názvu Deep Deterministic Policy Gradient, čo v preklade znamená hlboké učenie deterministického gradientu politiky. Je založený na princípoch DQN, avšak je použiteľný len na spojitý priestor. V tejto práci je implementovaný pomocou rozšírenia aktér – kritik s využitím vlastností DQN.

Aktér aj kritik sú aproximovaný NN. Ich úloha ostáva rovnaká, ako pri základnom princípe aktér - kritik. Učenie musí byť tiež modifikované. Agent totižto používa pamäť skúseností, ktorá sa nevyskytuje v koncepte aktér – kritik. Modifikovanie učenia je nasledovné. Kritik sa učí z dávky skúseností pre každú skúsenosť z dávky podľa

$$\omega_* = \omega + \beta * \gamma * Q_\omega(s, a) \tag{12}$$

kde $\boldsymbol{\omega_*}$ sú nové váhy NN nastavené podľa hodnoty starých váh NN zvýšených (alebo znížených) o diskontovanú Q - hodnotu v aktuálnom stave $\boldsymbol{Q_\omega(s, a)}$, s minimalizovaním štvorcovej chyby

$$\sum_{i=1}^{B} (y_i - Q_\omega(s, a))^2 \tag{13}$$

Kde B je veľkosť dávky a $\boldsymbol{y_i}$ je

$$y_i = r_i + \gamma * Q_\omega(s_i', a_i') \tag{14}$$

Aktér je učený gradientom kritika, podľa časovo oneskorenia všetky zložky dávky stavov a akcií (s hodnotami akcií)

$$grad = \sum_{i=1}^{B} Q_\omega(s_i', a_i') \mid a = \pi_\theta(s) \tag{15}$$

Gradient pre učenie je stanovený na základe hodnôt nasledujúcich stavov $\boldsymbol{Q_\omega(s_i', a_i')}$ (podľa hodnoty kritika) a hodnoty akcií $\boldsymbol{a}$ zvolených v danom stave $\boldsymbol{s}$ na základe aktuálne zvolenej politiky $\boldsymbol{\pi_\theta}$.

5 Návrh a implementácia

5.1 Programovací jazyk

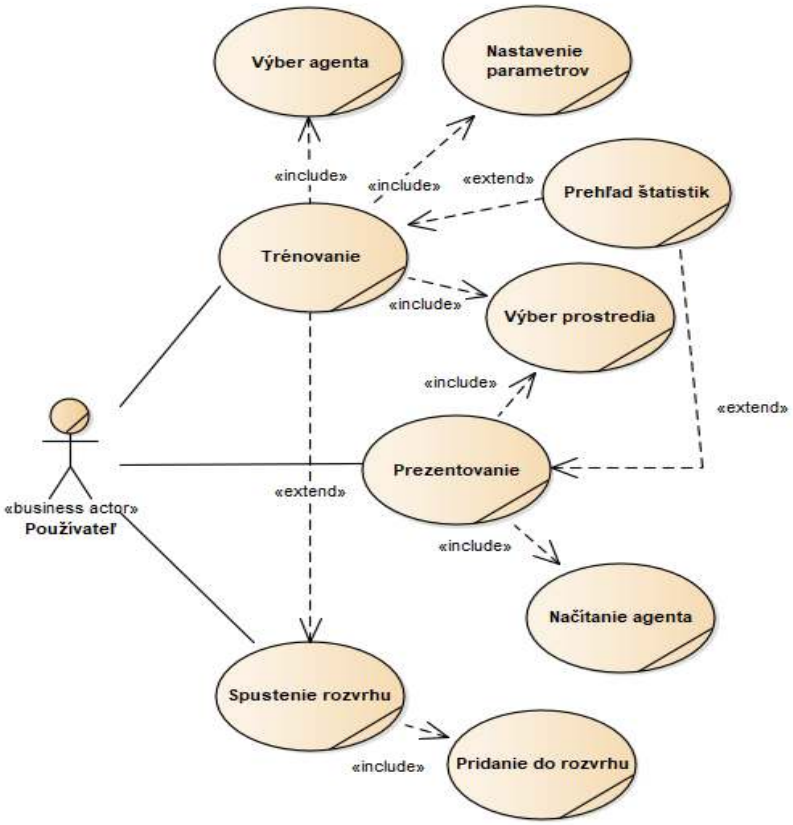
Programovací jazyk, ktorý bol zvolený pre implementáciu potrieb tejto práce je jazyk Python. Zvolený bol preto, že tento jazyk obsahuje vhodnú podporu viac rozmerných polí. Ďalšou vhodnou vlastnosťou je jeho podpora paralelného výpočtu na grafickej karte, čo výrazne zrýchľuje výpočty pri trénovaní.

Nakoľko sa v tejto práci venujeme RL, v Python je k dispozícii knižnica BOX2D, ktorá implementuje niekoľko zaujímavých prostredí na riešenie pre agentov RL. Ku každému prostrediu existuje tabuľka dosiahnutých výsledkov, s ktorými je možné porovnať výsledky agenta.

Ďalšou výhodou tohto jazyka je podpora knižníc Keras a Tensorflow. Keras je aplikačnou nadstavbou knižnice tensorflow, ktorý dovoľuje rýchlu tvorbu neurónových sietí z najvyššieho pohľadu, ktoré sa pri RL používajú ako univerzálne prostriedky na odhad hodnôt funkcií.

5.2 Prípady použitia

Z pohľadu používateľa boli identifikované nasledovné prípady použitia.



Obrázok 9 Diagram biznis prípadov použitia

Podľa diagramu bola vytvorená aplikácia. Aplikácia umožňuje používateľovi vybrať si z troch hlavných prípadov použitia. Prvým je trénovanie, kde užívateľ môže po vytvorení modelu, nastavení jeho parametrov a výbere prostredia spustiť trénovaciu fázu.

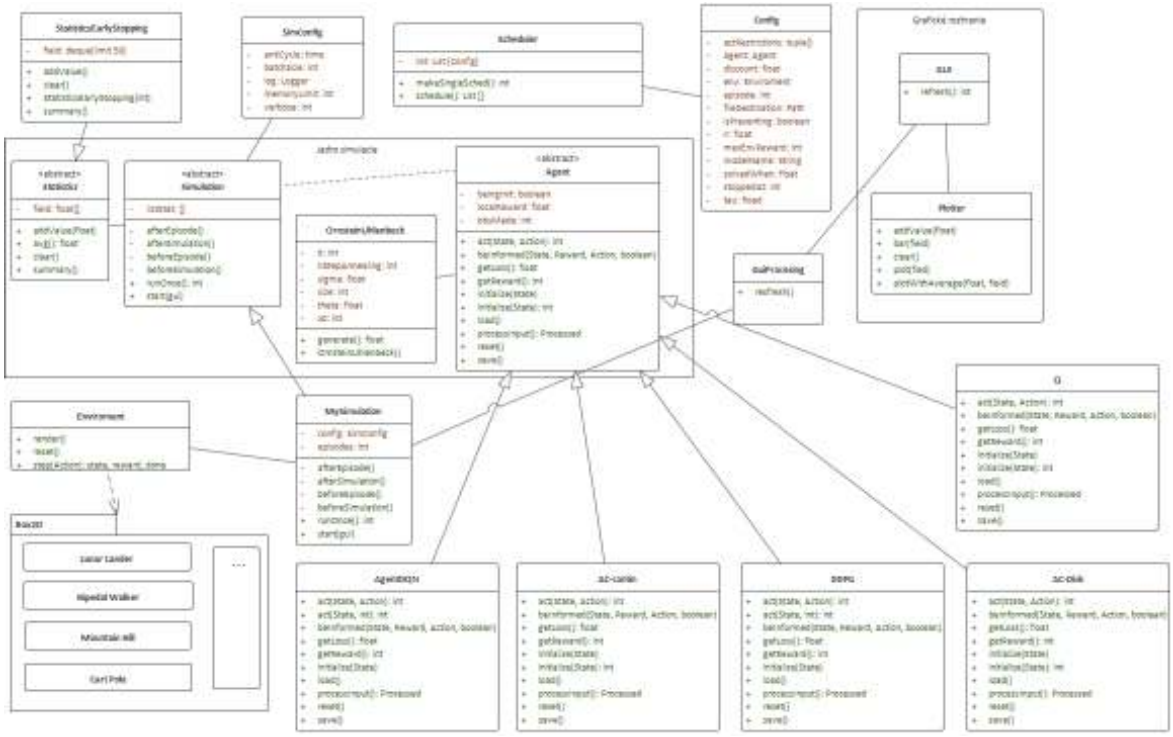
Druhým je spustenie prezentovania. Užívateľ si môže vybrať natrénovaného agenta, ktorého otestuje na vybranom prostredí. Prostredie môže byť ľubovoľné. Treba však rešpektovať, na ktorom prostredí bol agent trénovaný, z hľadiska dodržania korektnej formy vstupov.

Posledným je spustenie rozvrhu. Užívateľ si môže do rozvrhu zaradiť viacero agentov s prostrediami, na ktorých sa spustí tréning. Tento prípad použitia je vhodný pre vytvorenie sekvencie modelov, ktoré je potrebné natrénovať alebo otestovať za sebou.

Nastavenie parametrov je tiež dôležité. Užívateľ môže upravovať parametre agenta, ako aj parametre NN. Tieto parametre sú dôležitou časťou toho, či model bude naozaj fungovať, a preto by im mal užívateľ venovať patričnú pozornosť.

5.3 UML model tried

Aplikácia je zložená z dvoch základných častí. Jednou je abstraktné jadro, ktoré obsahuje simulačný proces, šablónu agenta a štatistík. Druhou je okolie, ktoré obsahuje triedy konkrétnej implementácie agentov, štatistík a ďalších tried.



Obrázok 10 UML model tried

5.3.1 Popis hlavných tried

Simulácia

Základnou jednotkou programu je trieda simulácia. Trieda predstavuje základný abstraktný cyklus učiaceho behu algoritmu, ktorý musia agenti používať. Skladá sa z nasledovných metód:

Metóda	Popis
BeforeSimulation	Inicializuje štatistiky, agenta, prostredie
BeforeEpisode	Obnoví nastavenia agenta, prostredia
SimulationRun	Spustí celú simuláciu
EpisodeRun	Spustí jednu epizódu
AfterEpisode	Zozbiera štatistiky, aktualizuje grafy
AfterSimulation	Uloží agenta, výsledky štatistík

Tabuľka 1 Vnútorný cyklus simulácie

Pri spustení simulácie sa inicializuje agent vstupmi z grafického rozhrania a nastaví svoje hyper-parametre. V simulácii je inicializované prostredie. Pred samotným behom epizódy je agentovi poslaný počiatočný stav, v ktorom sa nachádza.

Po inicializácii sa spúšťa simulačný beh. Pseudokód je popísaný na nasledovne.

inicializuj agenta počiatočným stavom s_0 z prostredia ***prostredie***
pre čas $t <$ (***env limit***):
    a = agent vyber akciu(s_t)
    $s_{t+1}, r_t,$ ***koniec*** = ***prostredie*** urob krok(a)
    Informuj agenta ($s_{t+1}, s_t, r_t,$ ***koniec***)
    ak ***koniec*** -> ukonči epizódu
    oznám agentovi ukončenie epizódy

Obrázok 11 Pseudokód vykonávania operácií agenta v implementovaných prostrediach

Keď agent ukončí beh jednej epizódy, pozbierajú sa štatistiky. Tieto štatistiky sú následne zobrazované na grafickom rozhraní pre užívateľský monitoring učiaceho cyklu. Po ukončení celej simulácie sú agentove štatistiky aj s jeho natrénovanými modelmi uložené do užívateľom zvoleného priečinka s názvom modelu.

5.3.2 Štatistiky a ich meranie

Pre otestovanie dosiahnutých výsledkov agenta sú použité štatistiky. Podľa týchto štatistík sa monitoruje priebeh učenia agenta a užívateľ môže sledovať, či sa jeho agent s danými parametrami učí podľa predpokladov. Štatistiky sú rozdelené na povinné a voliteľné.

Povinné slúžia na určenie výsledku agenta. Sú to štatistiky, ktoré sledujú vývoj priemernej odmeny za celé pozorované obdobie a absolútnu odmenu za posledných 50 epizód. Agenta považujeme za naučeného, pokiaľ dosiahne priemernú odmenu za posledných 50 absolútne zozbieraných hodnôt odmien na dopredu stanovenú percentuálnu presnosť (základne používame 85% z maximálnej odmeny v prostredí).

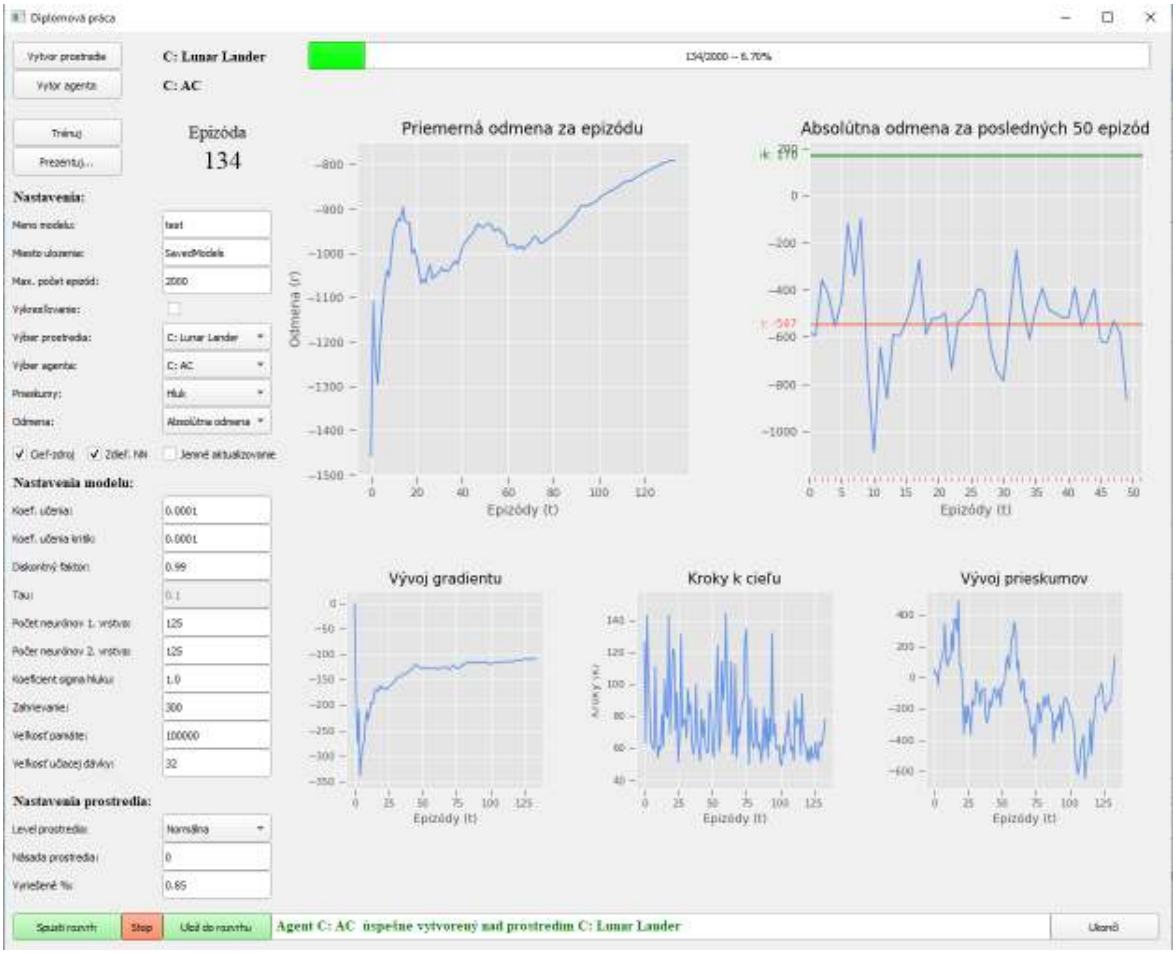
Voliteľné štatistiky nie sú pomenované podľa toho, či sa dajú zobraziť, alebo nie. Ich pomenovanie vyplýva z toho, že nemajú priamy dôsledok na agentove učenie a sú teda len podpornými štatistikami ukazujúcimi, či všetko v agentovi funguje tak, ako má. Tieto štatistiky neurčujú agentovu presnosť, ale zobrazujú smer agentovho vývoju a doposiaľ naučenej politiky. Tieto štatistiky sú zobrazenie počtu krokov k cieľu, vývoj prieskumov a vývoj gradientu, respektíve optimalizačnej funkcie. Ich stručný popis s očakávaným vývoj v prípade dobrého učiaceho procesu a vhodnej implementácie popisuje nasledujúca tabuľka.

Štatistiky	Popis	Očakávaný vývoj
Povinné		
Priemerná odmena	Odmeny agenta merané od začiatku simulácie priemerované podľa epizód	Mierna oscilácia, neskôr mierny nárast
Absolútna odmena (50)	Odmeny za posledných 50 epizód + priemer	Oscilácia, pri konci učenia ustálenie
Voliteľné		
Počet krokov k cieľu	Počet krokov agenta od stavu s_0 do určeného cieľa	Oscilácia, nemusí nastať zníženie
Vývoj funkcie gradientu alebo Q	Vyčíslenie zmien vo váhach agentov	Oscilácia, neskôr ustálenie
Vývoj prieskumov	Počet prieskumov agenta	Oscilácia, ustálenie

Tabuľka 2 Sledované štatistiky

5.4 Grafické prostredie aplikácie

Aplikácia pozostáva z hlavného grafického zobrazenia a zobrazenia bežiacej hry. Hra však nemusí byť vykresľovaná pre zrýchlenie interakcií medzi agentom a prostredím.



Obrázok 12 Užívateľské prostredie aplikácie

Na hlavnej obrazovke môže užívateľ nastaviť parametre modelov v ľavej časti obrazovky. Obmedzené parametre sú veľkosť pamäte a veľkosť učiacej dávky, kvôli obmedzenej výkonnosti počítača. Pri RL však nie sú kladené veľké požiadavky na výkon, no tieto parametre by mohli spôsobiť problémy na menej výkonných zariadeniach. Taktiež nepotrebujú veľkú a zložitú NN (vo veľkom množstve úloh), pretože ich sila je v interakcii s prostredím. Tento proces sa však nedá výrazne urýchliť, a preto sme viacej obmedzení výkonom prostredia ako procesorom (alebo grafickou kartou).

V spodnej časti obrazovky je možné prispôsobiť úroveň obtiažnosti prostredia. Aplikácia poskytuje tri levely – normálny, ťažký a extrémny. Pre zopakovanie vývoja v prostredí poskytuje aplikácia voľbu násady. Avšak opakovanie je možné len pre vývoj prostredia (ak pozostáva z levelov). Agent totiž nedokáže replikovať svoje rozhodnutia v požadovanom poradí počas trénovacej fázy.

Pred spustením tréningu je nutné vytvorenie prostredia. Po jeho výbere, je užívateľovi poskytnutý výber agenta podľa typu prostredia. Agentovi je možné nastaviť jeho vnútorné parametre. Po týchto nastavenia je možné spustiť trénovaciu fázu. Trénovacia fáza je ukončená, keď agent po sebe idúcich 50 epizódach dosiahne priemernú hodnotu odmeny väčšiu ako 85% z maximálne možne dosiahnuteľnej odmeny v prostredí. Percento je možné zmeniť.

Prezentovanie naučeného agenta je možné pomocou tlačidla prezentuj. Po výbere zložky s uloženým modelom sa spustí prezentovanie agentových výsledkov na prostredí, na ktorom bol trénovaný. Pri prezentovaní sa zobrazujú užívateľovi výsledky meraných štatistík pre vhodnú ilustráciu výsledkov. Agent sa v tomto režime už neučí.

Po výbere prostredia a agenta sú užívateľovi sprístupnené nastavenia modulov. Podľa zvoleného modelu je možné vybrať spôsob tvorby prieskumov. Ďalej je možné zmeniť spôsob odmeňovania z klasického na jeden z ponúknutých iných spôsobov. Taktiež je možné použiť moduly pre zlepšenie vlastností modelu na zvolenom prostredí.

Užívateľ môže uložiť konfiguráciu viacerých model do rozvrhu. Po vytvorení prostredia, agenta a po vyplnení ich nastavení môže užívateľ uložiť vytvorenú konfiguráciu do rozvrhu. Po uložení môže tieto konfigurácie trénovať automatizovane po sebe. Využitie je vhodné na trénovanie napríklad počas noci alebo pri porovnávaní výsledkov viacerých modelov.

5.5 Prostredia - Hry

V tejto práci som využil prostredia, ktoré boli uspôsobené pre potreby RL. Každé z nich predstavuje určitý problém vo forme hry, ktoré je nutné vyriešiť. Všetky sú dostupné na zdroji https://gym.openai.com/envs/, kde sú voľné k stiahnutiu. Spomenuté prostredia sú implementované v grafickom balíčku Box2D, v ktorom je možné implementovať viacero hier v 2D grafickom režime.

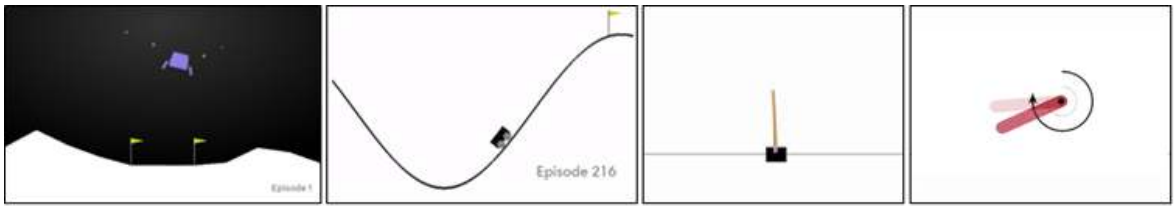
Prostredia sú vnútorne implementované tak, že hlavnou metódou je metóda step, ktorá vykoná na základe zvolenej akcie krok v hre. To znamená, že ak je akcia zvolená „dopredu“, prostredie vykoná túto akciu a agentovi vráti výsledok tejto operácie ako trojicu – nasledujúci stav, odmena a booleovskú hodnotu, či agent vyriešil daný problém. Ďalšie informácie agent nemá prístupné.

Použité hry pre testovanie sú popísané v nasledovnej tabuľke.

Prostredie	Cieľ	Priestor stavov – popis stavov	Priestor akcií	
			Diskrétny	Spojitý
Mountain Car	Dostať auto na kopec	2 - Rýchlosť, pozícia	3 – potlačenie vpravo, vľavo, nič	1 - potlačenie auta <-1,1>
Lunar Lander	Pristáť s modulom	8 – modul X a Y, rýchlosť hore a dole, orientácia, uhlová rýchlosť, nohy ľavá a pravá	3 – hlavný motor, ľavý motor, pravý motor	2 – hlavný motor <-1,1>, bočné motory <-1,1>
Cart Pole	Balansovanie tyč na vozíku	4 – pozícia tyče, rýchlosť vozíka, uhol tyče, rýchlosť tyče na vrchu	2 – Posun vpravo, posun vľavo	-
Pendulum	Udržať tyč vzpriamenú	3 – cos, sin, uhol	-	1 – uhlová rýchlosť

Tabuľka 3 Zoznam použitých prostredí

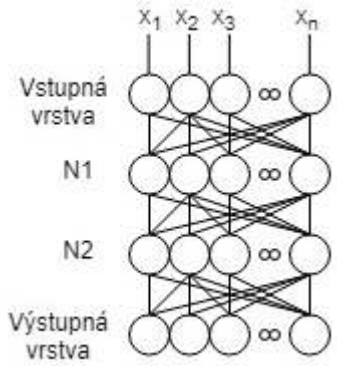
Grafické ukážky prostredí zachytáva nasledujúci obrázok.



Obrázok 13 Testovacie prostredia (Lunar Lander, Mountain Car, Cart Pole, Pendulum)

5.6 Implementácia neurónových sietí

Neurónové siete majú pre RL agentov úlohu aproximátora Q - hodnôt, prípadne hodnôt akcií alebo stavov. V tejto práci boli použité NN typu MLP, ktoré majú dve skryté vrstvy a jednu vstupnú vrstvu. Implementované boli pomocou frameworku keras.



Obrázok 14 Architektúra neurónovej siete typu MLP

Aplikácia neumožňuje pridanie ďalšej vrstvy, nakoľko dopredná neurónová sieť už s tromi vrstvami je považovaná za univerzálny aproximátor [103]. V aplikácii je však možné nastavenie počtu neurónov na jednotlivých vrstvách.

Aktivačná funkcia pre všetky neuróny skrytých vrstiev bola použitá funkcia relu. Predpis funkcie popisuje nasledovná rovnica.

$$relu\ (x)\ \{ \begin{matrix} x & ak\ x > 0 \\ 0 & inak \end{matrix} \tag{16}$$

Vstupná a výstupná vrstva používajú lineárnu aktivačnú funkciu.

$$linear\ (x) = x \tag{17}$$

V diskrétnych prostrediach na výstupných vrstvách (pre modely používajúce NN na predpovedanie pravdepodobnosti akcií) je použitá aktivačná funkcia softmax (označovaná aj ako normalizovaná exponenciálna funkcia), ktorá transformuje predpovedané hodnoty NN na pravdepodobnosti podľa nasledujúceho predpisu.

$$softmax(x) = \frac{\exp(x_j)}{\sum_{i=0}^{n} \exp\ (x_i)} \tag{18}$$

Výsledok aktivačnej funkcie je daný pomerom hodnoty exponenciálnej funkcie neurónu $\boldsymbol{j}$ ku sume hodnoty exponenciálnej funkcie cez všetky neuróny výstupnej vrstvy.

5.7 Implementácia Agentov

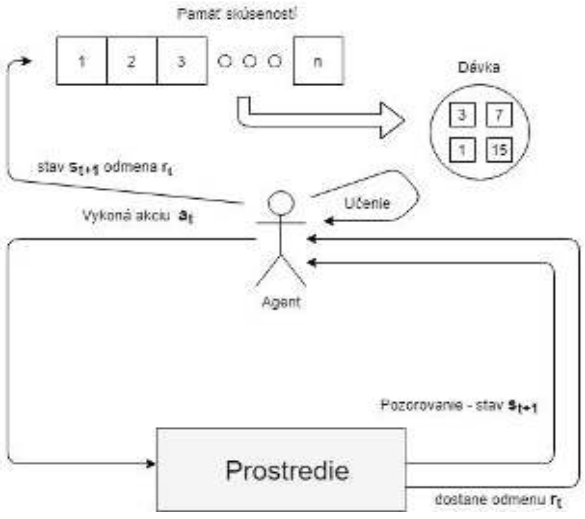
5.7.1 Q - učenie

Základné prostredie pre učenie problému využívajúce tabuľku bolo implementované pre porovnanie výsledkov z ostatnými diskrétnymi agentami. Metóda môže byť zaujímavá v počte krokov, ktoré dosiahne pri nájdení riešenia v niektorých prostrediach.

Široký stavový priestor pri niektorých hrách je problematický. Z toho dôvodu boli stavy prevedené na diskrétnu formu pomocou konštánt $\boldsymbol{\partial_1 = 0.1}$ alebo $\boldsymbol{\partial_2 = 0.01}$. Konštanta bola aplikovaná na každú zložku stavu. Napríklad v diskrétnom prostredí Mountain Car je stav definovaný dvomi zložkami v rozsahu <-1.2 , 0.6> a <-0.07 , 0.07>. Po rozdelení na diskrétne zložky má prvá zložka stavu 18 častí (0.1) a druhá zložka 14 častí (0.01). Spolu je možných 252 kombinácii, čo značne zľahčuje situáciu. Samozrejme, treba počítať stým, že určité vlastnosti v prostredí môžu byť zanedbané v dôsledku diskretizácie stavov prostredia.

5.7.2 DQN

Koncept DQN je z pomedzi RL algoritmov široko používaný na riešenie problémov. Jeho hlavnou výhodou je využitie skúseností, ktoré sa naučil v predošlých epizódach a pri učiacom cykle si náhodnú dávku týchto skúseností pripomína. Pamäť je implementovaná ako zreťazený zoznam s obmedzením kapacity na užívateľom zvolenú konštantu. Po naplnení kapacity sa najstaršie spomienky nahrádzajú novšími.



Obrázok 15 Implementácia pamäti skúseností do učiaceho cyklu

Ako je zobrazené na obrázku vyššie, agent sa po vykonaní akcie dostane do ďalšieho stavu. Tento stav si uloží do pamäte. Potom vyberie z pamäte vzorky vo veľkosti dávky.

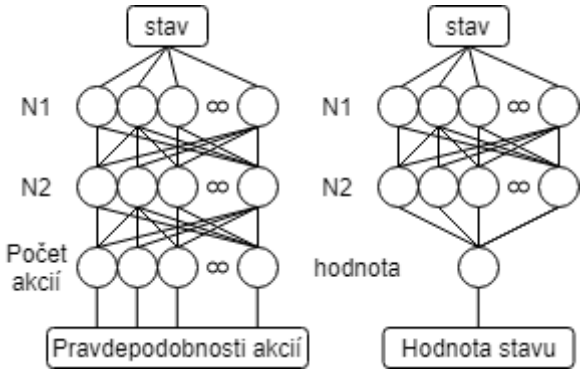
Dávka je použitá na trénovanie NN pomocou jednej epochy, kde cieľová hodnota stavu pre akcie je určená podľa nasledujúceho predpisu.

$$cieľová\ hodnota_s = r_{\ s} + \gamma * \max(\ Q(s', a')) \tag{19}$$

5.7.3 Aktér - kritik

Základný koncept aktér - kritik predstavuje síce silnú myšlienku, ale je potrebné ju ďalej rozšíriť, pretože nie je veľmi stabilná. Taktiež hľadanie optimálnych váh môže uviaznuť v lokálnom extréme, čo tiež nie je žiadúce. Vhodné v niektorých prípadoch môže byť aj redukovanie počtu krokov k cieľu, čo základná forma do veľkej miery nepodporuje. Tieto vlastnosti môžeme však zlepšiť pridaním niektorých vylepšení, ktoré sú popísané v experimentoch.

Agent je implementovaný s neurónovými sieťami vo forme aproximátora modelov aktéra a kritika. Modifikácia NN je zobrazená na nasledujúcom obrázku.



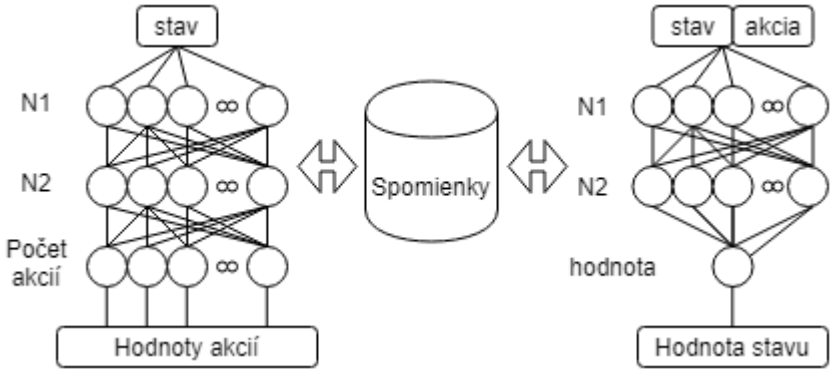
Obrázok 16 implementácia aktér-kritik pomocou NN

Počty neurónov na jednotlivých vrstvách sú voliteľnými parametrami (N1,N2). Vstupná a výstupná vrstva je generovaná podľa prostredia. Vstupná vrstva podľa obrázku vyššie prijíma vstupy vo veľkosti stavu prostredia.

5.7.4 DDPG

Rozšírenie DQN na spojité pomocou AC

Sila použitia DQN sa prejavuje vo viacerých typoch hier. Agent sa učí bez ohľadu na politiku podľa svojich spomienok, ktoré ma uložené. Rozšírenie DQN sa dá previesť viacerými spôsobmi. V tejto bolo rozšírenie implementované pomocou aktér-kritika nasledovne.



Obrázok 17 Rozšírenie pomocou metódy aktér – kritik pre spojitý priestor

Týmto rozšírením bol vytvorený model DDPG. Využíva hlavnú výhodu modelu DQN – pamäť spomienok (skúseností). Použitie modelov aktéra a kritika je možné rozšíriť model z diskrétneho na spojitý. Úloha aktéra ostáva taká, ako pri štandardnom aktér-kritik algoritme. Aktérova NN predpovedá hodnoty akcií, nie však pravdepodobnosti ale predpovedá presné hodnoty týchto akcií.

Vstupy pre NN kritika sú pozmenené. K vstupnému vektoru stavu sú priložené hodnoty akcií. Kritik odhaduje hodnotu nie len na základe stavu, ale aj hodnoty akcií. Podľa toho dokáže pri trénovaní aktér prispôsobovať váhy NN pre presnejšie predpovedanie hodnôt akcií.

Aktér – kritik je algoritmus, ktorý sa učí s ohľadom na politiku. DQN však nie. Tento rozdiel sa však vyrieši tým, že do vstupov kritika sa pripojil aj vektor akcie. Hodnoty akcií sa totiž prenášajú spolu so stavom. Tieto hodnoty boli určené na základe predošlej politiky. Pri učení sa tak použije stav aj s hodnotami akcií podľa minulej politiky a porovná sa s aktuálnou.

5.8 Validácia základných modelov

Vytvorení agenti boli validovaný podľa priemernej dosiahnutej odmeny za posledných 50 epizód. Ak sa im podarilo dosahovať v priemere nad 85% z maximálnej odmeny, model bol považovaný za naučený. Ďalším aspektom validácie bol počet epizód do vyriešenia úlohy. Limit epizód bol stanovený na 2000 so zastavením po dosiahnutí 85% z maximálnej odmeny prostredia. Výsledky boli porovnané s inými modelmi, ktoré boli v prostrediach použité. Externé prostredia však nemajú zverejnené vnútorné parametre, preto boli použité výsledky na porovnanie tých modelov, ktoré boli štruktúrou

najpodobnejšie [27]. Agenti použitý v tejto práci mali nastavené parametre podľa teórie NN a RL. Zoznam agentov s výsledkami zobrazuje nasledujúca tabuľka.

Prostredia	Agenti bez modulov					Najlepšie výsledky porovnávaných agentov
Prostredie - diskrétne	DQN (D)	Q (D)	AC (D)	AC (C)	DDPG (C) bez modulov	
Cart Pole (D)	425	1877	888	-	-	20 - 130
Lunar Lander (D)	580	>2000	1012	-	-	239 - 658
Mountain car (D)	301	1345	411	-	-	75 - 341
Prostredie - spojité						
Lunar Lander (C)	-	-	-	1874	1102	100 - 1500
Mountain car (C)	-	-	-	643	88	120 - 140

Tabuľka 4 Testovanie agentov v dosiahnutí 85% maximálnej odmeny merané počtom potrebných epizód na vyriešenie

Posledným ukazovateľom bolo sledovanie prezentovania agentových schopností v prezentačnom móde. V tomto móde nebolo agentovi umožnené ďalej sa učiť, len prezentoval finálny výsledok na hre, kde bol vizuálne kontrolovaný, či vyriešil dané prostredie.

Aktér – kritik v štandardnom koncepte dosahoval nie celkom uspokojivé výsledky na spojitom prostredí. Je to však spôsobené tým, že je to algoritmus, ktorý sa učí počas epizódy a nevyžíva pamäť skúseností, čiže v jednom kroku disponuje len jedným učiacim cyklom.

6 Experimenty

V tejto časti práce sa venuje vylepšeniam základných agentov. Ich výsledky podľa základných konceptov (tabuľka č.4) síce riešia daný problém, ale môžu ho riešiť efektívnejšie. V nasledujúcich experimentoch bude zamerané na vylepšenie faktorov úspešného riešenia úlohy vo viacerých ukazovateľoch. Hlavným ukazovateľom bude dosiahnutá odmena za posledných 50 epizód. Vedľajšími búdú napríklad vývoj učenia prostredníctvom priemernej chyby, počtu kroku k cieľu s potrebného času na vyriešenie (meraného počtom epizód).

6.1 Predpoklady experimentov

Pri testovaní modelov boli zachované počty vrstiev NN a počty ich neurónov pre agentov. Hodnoty ostatných parametrov sú v nasledujúcej tabuľke:

Prostredie		Počet neurónov		Prieskum
Prostredie - diskrétne	koef. učenia	1. skrytá vrstva	2. skrytá vrstva	Výber prieskumu
Lunar Lander (D)	0.001	1024	1024	Epsilon
Mountain Car (D)	0.003	512	256	Epsilon
Cart Pole (D)	0.001	1024	512	Epsilon
Prostredie - spojité	koef. učenia	1. skrytá vrstva	2. skrytá vrstva	Výber prieskumu
Lunar Lander (C)	0.0001	1024	1024	Šum
Mountain Car (C)	0.0001	512	256	Šum
Pendulum (C)	0.0001	256	256	Šum

Tabuľka 5 Hodnoty hyper-parametrov agentov v experimentoch

Pre agentov využívajúcich koncept aktér-kritik bol koeficient učenia kritika nastavený na hodnotu $\boldsymbol{\beta}$ 0,0003. Parameter presunu váh $\boldsymbol{\tau}$ 0.1 pre cieľovú – zdrojovú NN. Veľkosť dávky bola zafixovaná na 32. Maximálny počet epizód bol stanovený na 2000. Možnosť výberu prieskumu modelujeme pri diskrétnych prostrediach epsilonom spôsobom a pri spojitých prostrediach šumom. Pri experimente, v ktorom sa porovnávajú tieto dve metódy spolu s ďalšími možnosťami prieskumov, sú popísané aj ďalšie spôsoby prieskumov podrobnejšie.

6.2 Experiment vylepšenia agentov pridaním pomocných modulov

Agenti a v nich použité algoritmy majú svoje výhody aj nevýhody. Výhody agentov sú väčšinou dopredu známe z princípu učenia a teoretických podkladov. Nevýhody však vyplývajú z konkrétnej implementácie alebo prostredí. V tomto experimente si priblížime viacero modulov, ktorými je možné vylepšiť fázu tréningu agenta a odmeriame veľkosť zlepšení.

Oblasti, ktoré sa pokúsime zlepšiť, môžeme rozdeliť do dvoch skupín. V prvej časti sa budú pridávať moduly, s ktorými sa pokúsime zvýšiť stabilitu učenia. V druhej sa pokúsime o zrýchlenie dosiahnutia celkového riešenia agenta pomocou iných modulov. Rýchlosť budeme merať v epizódach potrebných na dosiahnutie požadovanej kvality riešenia podľa stanovených percent z maximálne možnej odmeny v prostredí.

6.2.1 Zväčšenie stability

Potreba stabilného vývoja učiaceho cyklu je pre každého agenta žiadúca. Pri RL je oscilácia v učiacom cykle veľmi častá, pretože chyba variuje podľa kvality riešenia meraného odmenou, ktorá je veľmi variabilná. Agent nemusí dosiahnuť najlepšie riešenie, ako napríklad pri nastavení vyššieho koeficientu učenia pri NN, môže agent doslova preskočiť požadované riešenie. Preto je z hľadiska nájdenia riešenia prijateľnejšie pomalšie, ale zato stabilnejšie učenie, ako rapídne a rozkolísané učenie.

Cieľová – zdrojová NN

Táto myšlienka je využívaná vo viacerých modeloch v súčasnosti. Spočíva v rozdelení modelov do dvoch častí. Jedného zdrojového a druhého cieľového modelu. Počas trénovania v rámci jednej epochy sa na odhad cieľových hodnôt používa NN. Tá sa však počas epochy prispôsobuje (buď časovo oneskorenou chybou alebo deltou Q - hodnôt), čo modifikuje jej odhady hodnôt do ďalších časových úsekov počas epizódy. Učenie vyzerá pri DQN nasledovne:

$$Q_\theta(s,a) = r + \gamma * \max(Q_\theta(s',a')) \tag{20}$$

Použitie len jednej NN môže spôsobiť oscilovanie vývoja odmeny, pretože pri ovplyvnení váh hodnoty $\boldsymbol{Q(s_t, a)}$ sa často ovplyvní aj hodnota $\boldsymbol{Q(s_{t+1}, a)}$, čo môže zvýšiť cieľovú hodnotu pre nasledujúce odhady. Z týchto dôvodov sa na odhady používa cieľová NN, ale modifikuje sa len zdrojová NN. Po zvolenom časovom úseku, najprirodzenejšie po jednej epizóde, sa presunú váhy zdrojovej NN do cieľovej NN [23]. Učenie sa teda zmodifikuje nasledovne:

$$Q_{zdrojová}(s,a) = r + \gamma * \max(Q_{cieľová}(s',a')) \quad (21)$$

Cieľová sieť sa počas učenia v celej epizóde nemení, a preto produkuje stabilné hodnoty pre stavy v čase epizódy (zachovanie aktuálnej politiky). Účinok tréningu sa prejaví až presunom váh do cieľovej NN po epizóde alebo po inom zvolenom intervale (zmena aktuálnej politiky).

Jemné aktualizovanie váh

Vylepšenie spočíva v tom, že agent pri presune váh z zdrojovej NN do cieľovej NN použije parameter $\boldsymbol{\tau}$ [24]. Tento parameter je volený v rozsahu (0,1) a bude ním zjemňovaná zmena vo váhach podľa nasledujúceho predpisu.

$$\theta_{cieľová} = \theta_{zdrojová} * \tau + (1-\tau) * \theta_{cieľová} \quad (22)$$

Rovnica vyjadruje jemný presun váh zo zdrojovej NN do cieľovej NN. Učenie sa ďalej modifikuje tak, aby sa váhy častejšie aktualizovali. Vplyv na cieľovú sieť by mal byť taký, že pomalšími krokmi dosiahne požadovanú hodnotu odhadov.

Oproti základnému spôsobu cieľovej – zdrojovej NN sa aktualizuje cieľová sieť aj v priebehu epizódy. Dôsledkom je síce zvyšovanie cieľovej hodnoty predpovedí, ale tento efekt je zjemnený natoľko, že len jemne zvyšuje hodnotu cieľa. Výhodou však je, že skok po presune váh sa redukuje omnoho viacej, ako skok po presune váh v štandardnom koncepte cieľovej – zdrojovej NN.

Výsledky

Pri dodržaní predpokladov experimentov boli otestovaní agenti na diskrétnych aj spojitých prostrediach. Experimentovalo sa len s agentami, ktorý využívali NN. Maximálny počet epizód bol zafixovaný na 2000 s možnosťou skoršieho zastavenia učenia, v prípade dosiahnutia požadovaného výsledku (85% z maximálnej odmeny prostredia). Pri jemnom aktualizovaní váh bol použitý koeficient $\boldsymbol{\tau} = 0.1$. Ako diskrétne testovacie prostredie bolo použité prostredie Cart Pole. Na začiatok experimentu bolo cieľom odmerať rozdiely v epizódach s pridaním modulov. Neskôr bol meraný vplyv na stabilitu gradientu. Výsledky zobrazuje nasledujúca tabuľka.

Diskrétni agenti	Cart Pole (D)		
	Zdroj – cieľ	Jemné aktualizovanie	Bez modulov
DQN (D)	398	615	425
AC (D)	821	985	888

Tabuľka 6 Porovnanie rýchlosti nájdenia riešenia v epizódach v prostredí Cart Pole

Ako môžeme vidieť z tabuľky vyššie, počet epizód na nájdenie riešenia sa predĺžil po pridaní modulov. Najmä pri jemnom aktualizovaní váh, kde bol nárast markantný. Použitie zdrojovej a cieľovej NN výsledky výraznejšie nezmenilo. Podstatnejší je však vývoj gradientu, ktorý je na obrázku č. 18 nižšie.

Ďalšie meranie prebehlo na spojitom prostredí Lunar Lander a boli otestovaní spojití agenti. Ich výsledky zobrazuje tabuľka nižšie.

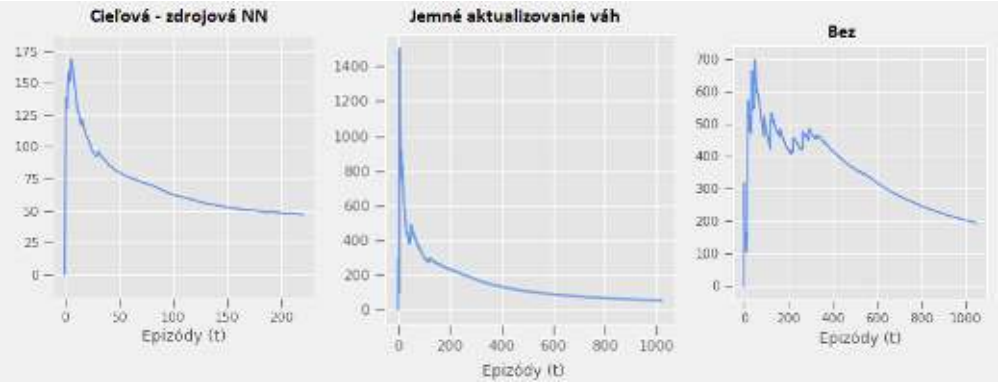
Spojitý agenti	Lunar Lander (C)		
	Zdroj – cieľ	Jemné aktualizovanie	Bez modulov
DDPG (C) bez modulov	897	1157	1102
AC (C)	1757	1954	1874

Tabuľka 7 Porovnanie rýchlosti nájdenia riešenia v epizódach na spojitom prostredí Lunar Lander

Ako sme aj predpokladali, učiaci cyklus sa markantne predĺžilpri jemnom aktualizovaní váh aj na spojitom prostredí. Menšie spomalenie bolo spôsobené agentovi DDPG, ktorý využíva pamäť skúseností, vďaka ktorej aktualizuje svoje váhy. Pri trénovaní si pripomína dávku vzoriek, ktoré ovplyvňujú váhy NN politiky a teda častejšie aktualizuje váhy počas epizód. Vďaka tomu sa neprejaví spomalenie v počte epizód tak veľmi, ako v modely AC.

Bez využitia zdrojovej – cieľovej NN, sa proces učenia stáva nestabilnejším podľa spomínaného problému so zvyšovaním cieľovej hodnoty. Pridaním cieľovej – zdrojovej NN však tento efekt minimalizujeme, pretože hodnota dávky skúseností vstupujúcich do učiaceho cyklu je prepočítaná aktuálnou politikou cieľovej NN v rámci epizódy. Častým javom je zrýchlenie učiaceho cyklu, pretože sú minimalizované nepresnosti, ktoré vznikajú podľa spomínaného problému zvyšovanej hodnoty cieľu. Toto zrýchlenie ukazuje aj tabuľka vyššie pri oboch agentoch.

Dôležitejším aspektom je však stabilita trénovania. V nasledujúcom grafe sú porovnané gradienty spomenutých rozšírení voči modelu DQN bez rozšírení.



Obrázok 18 Porovnanie stability vývoju gradientu s pridanými modulmi na agentovi DQN na Lunar Lander

Na obrázku vyššie môžeme vidieť, že použitie cieľovej - zdrojovej NN stabilizuje vývoj gradientu. V porovnaní s nevyužitím žiadneho modulu sa jeho vývoj podstatne v prvých krokoch stabilizoval. Jemné aktualizovanie váh spôsobilo ešte stabilnejší gradient, ktorý však veľmi pomaly klesá. To znamená, že síce sa učenie stabilizuje a vo veľmi malých krokoch prechádza možné riešenia, ale k riešeniu sa približuje pomalšie.

Metóda cieľovej a zdrojovej NN je praktickejšia, hlavne v menších prostrediach. Avšak v týchto prostrediach použitie aj jemnej aktualizácie váh spomalí učiaci cyklus, čo nemusí byť žiadúce, pretože v menších prostrediach majú agenti tendenciu rýchlejšej konvergencie k cieľu.

Vo väčších a komplexnejších prostrediach je však vhodnejšie použiť použitie aj jemného presunu váh. Práve preto, že riešenie si vyžaduje väčšie množstvo učiacich krokov a teda spomalenie pridaním tohto modulu nie je prekážkou. Jemné aktualizovanie váh viacej stabilizuje vývoj gradientu, čo má lepší vplyv na jemné určovanie výsledných hodnôt akcií, najmä v spojitých prostrediach.

6.2.2 Zvýšenie rýchlosti nájdenia riešenia

Agenti by mali dosiahnuť riešenie problému prijateľnom čase. Táto rýchlosť závisí aj od veľkosti prostredia. Tú však mnohokrát nevieme ovplyvniť, preto musíme hľadať miesta na vylepšenie v agentoch.

Zdieľanie parametrov sietí

Parametre sietí môžu byť zdieľané. Avšak to platí len pre modely podobné aktér – kritik metodike, kde je hlavný model rozdelený do dvoch komponentov (prípadne pre iných agentov používajúcich viac ako jednu NN). Tieto komponenty môžu zdieľať váhy aj

všetkých vrstiev. NN by mali v tomto prípade rýchlejšie konvergovať k riešeniu. Tento predpoklad bude testovaný na diskrétnom aj spojitom prostredí.

Prieskum vs využitie naučených vedomostí

Od agentovej schopnosti preskúmavania prostredia výrazne závisí jeho dosiahnutý výsledok. Pokiaľ agent nedostatočne preskúmava prostredie, nemusí spoznať všetky stavy, medzi ktorými môžu byť aj stavy potrebné pre dosiahnutie kvalitného riešenia.

Pri diskrétnych modeloch je možné robiť prieskumy podľa náhodne generovanej zložky, v spojitých je nutné určiť hodnotu posunutia. V experimente porovnáme najmä využitie náhodnej zložky a generovania šumu v prostrediach oboch typov. Pomocou zmien parametrov sa pokúsime postupne sledovať vplyvy na početnosť a kvalitu prieskumov.

Dobrý výber prieskumov by mohol závisieť aj od ďalších faktorov. Napríklad, výber môže závisieť od kvality riešenia. Tento spôsob bol implementovaný pomocou maximálnej hodnoty v prostredí a doposiaľ získanou priemernou odmenou za posledných 50 epizód. Podľa tejto hodnoty bol prispôsobený výber prieskumov nasledovne.

Kvalita riešenia	0 – 25%	25 % - 50 %	50 % - 75 %	75 % - max hodnota
Možnosť prieskumu	70%	45%	20%	5 %

Tabuľka 8 Spôsob prieskumov podľa kvality riešenia

Nastavenia hyper-parametrov

RL používa ako aproximátor NN. Tieto neurónové siete sú zodpovedné za naučenie sa a udržanie vedomostí v agentovi. Parametre, podľa ktorých sa tieto NN vytvárajú, sú taktiež dôležité z pohľadu rýchlosti dosiahnutia výsledkov.

Pri teórii NN je všeobecne známe, že učiaci koeficient ovplyvňuje rýchlosť učenia. Avšak, NN za normálnych okolností používajú minimalizáciu chyby na základe chybového ukazovateľa. Ukazovateľ je počítaný podľa predpovedanej hodnoty a správnej hodnoty počas učiaceho cyklu. U agentov v RL sa však učia NN na základe získanej odmeny v daných stavoch. Stavy však nemajú priradenú presnú hodnotu, ku akej majú smerovať. Tieto hodnoty sa v čase tréningu menia, čo je zmena oproti normálnemu učeniu. Preto otestujeme vplyvy učiaceho koeficientu v agentovi RL.

Ďalším faktorom, ktorý môže ovplyvňovať rýchlosť získania riešenia, môže byť spôsob inicializácie NN. V niektorých prípadoch môže počiatočné nastavenie váh pomôcť v rýchlejšom nájdení riešenia.

Ďalšími parametrami, ktoré však už ovplyvňujú nastavenie cieľových učiacich hodnôt agenta, sú napríklad diskontný faktor, veľkosť učiacej pamäte alebo veľkosť učiacej dávky. Tieto parametre ovplyvňujú, akým spôsobom sa použijú vzorky pri učiacom cykle. Veľkosť učiacej dávky je však zhora obmedzená silou HW, ale aj rozumný limitom. Pokiaľ HW je schopný udržať veľkú dávku, treba stále dbať nato, aby jej veľkosť bola rozumná a nebrzdila agentov vývoj. Pokiaľ bude vzorka napríklad o veľkosti 128 a v prostredí, kde počas epizódy agentov použije 5 akcií v priemere na ukončenie epizódy, tak v počiatkoch učenia agentovi zvýrazní prvotné stavy a akcie, ktoré zväčša robí agent náhodne vo forme prieskumov, čo môže rozkolísať začiatky učenia.

Výsledky

Zdieľanie parametrov boli testované na prostredí Lunar Lander. Agent, ktorý bol testovaný, bol aktér – kritik. Aktér zdieľa s kritikom všetky vrstvy, okrem výstupnej vrstvy.

Lunar Lander – Agent AC	Diskrétne	Spojité
Zdieľané parametre	701	1759
Nezdieľané parametre	888	1874

Tabuľka 9 Testovanie agenta AC (D) a AC(C) so zdieľanými parametrami v prostredí Lunar Lander (D) a (C)

Ako môžeme vidieť, v diskrétnom aj spojitom prostredí bez zdieľania parametrov agent pomalšie konverguje k riešeniu. Je to spôsobné tým, že pri učení sa aktér aj kritik riadia tým istým gradientom, pretože majú rovnaké vstupy vo forme stavov. Tento gradient teda mení rovnakým smerom oba komponenty, rozdielna je už len jeho veľkosť. V učiacom cykle ale ovplyvní váhy kritika aj aktéra nezávisle.

V prípade, že by sme používali k vstupom v stave aj akciu v modely kritika, zdieľanie parametrov nemusí mať želaný efekt. Najväčší potenciál však môže byť hlavne v zdieľaní váh konvolučnej NN, pri použití obrázkového vstupu.

V spojitom priestore však zdieľanie parametrov nemusí priniesť veľké zlepšenie. V spojitom priestore totižto treba predpovedať, nielen ktorú akciu agent vyberie, ale aj hodnotu tejto vybranej akcie. Zdieľanie parametrov rozkolíše hľadanie hodnôt akcií, čo je zrejmé, nakoľko sa gradientom uspôsobia váhy dvojnásobne. V prípade diskrétneho prostredia by to nebol problém, nakoľko sa vyberá len jedna akcia s najvyššou hodnotou. Hodnota tejto akcie musí byť teda najvyššia, avšak dve rozdielne hodnoty akcie neovplyvnia prostredie (pokiaľ neporušia maximálny výber). Dôležité je len vybratie akcie a nie určenie jej veľkosti.

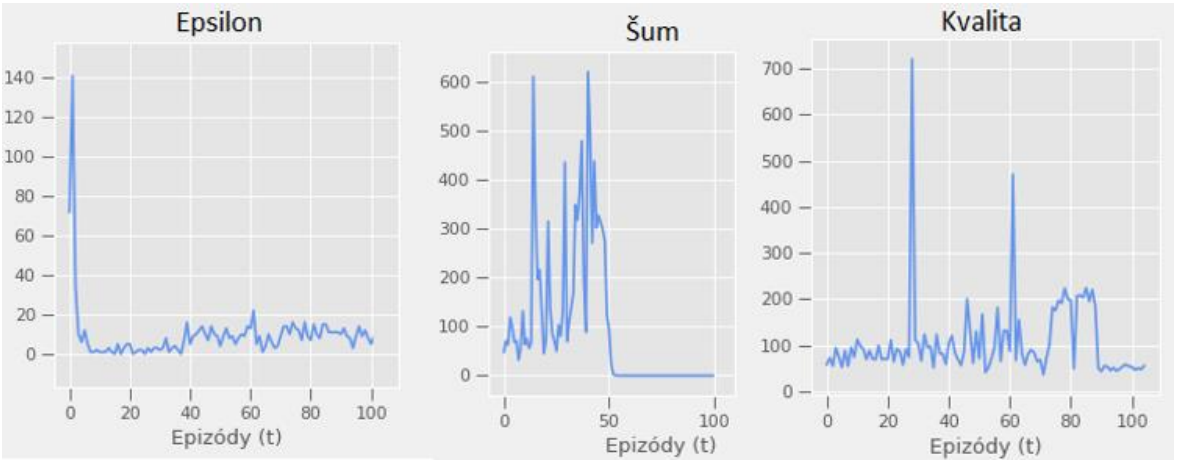
Ďalej bol testovaný vplyv prieskumov na rýchlosť nájdenia riešenia. V nasledujúcej tabuľke môžeme vidieť dopad spôsobu výberu prieskumov taktiež v prostredí Lunar Lander (diskrétnom aj spojitom) meranom počtom epizód.

Agenti Diskrétni	Šum	Epsilon	Kvalita
DQN (D)	591	580	603
Q (D)	>2000	>2000	>2000
AC (D)	1087	1012	1246
Spojití			
DDPG (C) – bez modulov	1102	-	1542
AC (C)	1874	-	1922

Tabuľka 10 Porovnanie vplyvu spôsobu prieskumov na počet epizód potrebných pre agentov na vyriešenie prostredia Lunar Lander (C) aj (D)

V tabuľke môžeme vidieť, že ako najlepší spôsob výberu prieskumov pre spojité prostredie je generovanie šumu. Pre diskrétne prostredie môže byť vhodné tiež niekedy použiť šum. Avšak treba dbať na to, že v diskrétnom prostredí sa vyberá jedna akcia, zväčša podľa pravdepodobnosti akcií. Šumenie pravdepodobností nemusí byť vždy najvhodnejšie, pretože napríklad pri epsilon prieskumoch, nie je potrebné vedieť hodnoty predikcií. To, či sa urobí prieskum, je rozhodnuté dopredu nezávisle od predpovedanej pravdepodobnosti podľa náhodnej hodnoty väčšej ako epsilon. Pri šume však treba túto zložku započítať do pravdepodobností, kde sa aj po započítaní nemusí vyvolať prieskum, pretože v diskrétnych prostrediach je častým javom, že ku naučeným stavom má cieľová pravdepodobnosť najlepšej akcie výrazne vyššiu hodnotu od iných akcií. A teda, aj po pripočítaní náhodnej zložky šumu nemusí byť prieskum vyvolaný. Preto v diskrétnych prostrediach je lepšie použitie epsilon náhodného spôsobu prieskumov, nakoľko nezasahuje do pravdepodobností predpovedanej akcie.

Nasledujúci graf zobrazuje vývoj početnosti prieskumov v čase agentovho učenia. Spôsoby prieskumov boli testované na diskrétnom prostredí Lunar Lander na agentovi DQN. Kvôli prehľadnosti sú grafy obmedzené na prvých 100 epizód, ktoré sú z pohľadu preskúmavania prostredia najdôležitejšie. Graf zobrazuje počet prieskumov v priebehu epizód, ktoré boli vyvolané spomenutými spôsobmi.



Obrázok 19 Porovnanie vývoja počtu prieskumov podľa epizód

Z grafu môžeme vidieť, že počet prieskumov pri epsilon spôsobe prieskumov je najvyšší v počiatkoch trénovania. Vtedy agent preskúma v prostredí ostatné stavy. Ak sa mu však nepodarí na začiatku preskúmať stavy, medzi ktorými by boli aj potrebné stavy pre nájdenie riešenia, v priebehu ďalšieho učenia už nemusí tieto stavy nájsť. V grafe však môžeme vidieť aj to, že v ďalšom priebehu sa začne jemne zvyšovať počet prieskumov. Je to spôsobené tým, že pri prostredí Lunar Lander sa zvyšuje počet krokov potrebných na riešenie, a preto s narastajúcim počtom krokov sa zväčšuje množstvo prieskumov, kvôli početnejšiemu vyhodnocovaniu epsilon náhodných prieskumov.

Generovanie šumu ako náhodnej zložky pre tvorbu prieskumov je vhodné použiť so zahrievacou fázou. Ako môžeme na grafe vidieť, po zahrievacej fáze 50 epizód sa počet prieskumov približuje k nule. Počas zahrievacej fázy však agent robí veľké množstvo prieskumov, ktoré nemajú klesajúcu tendenciu, čo je prijateľná vlastnosť. Ďalšou dobrou vlastnosťou je aj možnosť predlžovať zahrievaciu fázu, vďaka čomu dokáže agent robiť prieskumy dlhšie časové obdobie.

Preskúmavanie okolia podľa kvality riešenia je vhodné napríklad v prostrediach ako je Mountain Car. V nich je totižto nutné dosiahnuť riešenie čo najskôr (hoci aj náhodným spôsobom), pretože agent môže uviaznuť. Vtedy, podľa nezmenenej kvality riešenia dostane agent impulz na robenie prieskumov, čo môže byť nápomocné pre uvoľnenie uviaznutia agenta.

V porovnaní s ostatnými spôsobmi prieskumov sa javí epsilon spôsob ako najvhodnejší pre diskrétne prostredia. Alternatívnym spôsobom pre menšie prostredia môže byť využitie entropie. Hodnota entropie má tú výhodu, že sa bude znižovať nepriamo úmerne priemernej odmene. Presnejšie, s rastúcou odmenou bude klesať entropia, čo bude znižovať

aj počet prieskumov, a zase opačne. V porovnaní s epsilon spôsobom, kde sa najväčší počet prieskumov vykoná na začiatku, počet prieskumov pri entropii by závisel od kvality riešenie. Keby agent uviazol, entropia by bola väčšia a spôsobila by rapídne zväčšenie počtu prieskumov.

Pri experimentovaní s hyper-parametrami NN a hyper-parametrami agenta bolo použité prostredie Mountain Car. Toto prostredie je najvhodnejšie pre tento experiment, pretože agent musí pomocou rozkolísania dostať auto na kopec. Úloha je náročnejšia ako sa zdá, pretože pokiaľ agent nenájde skoré riešenie, uviazne medzi kopcami v lokálnom minime a ďalej sa už nepohne.

Preto bol na tomto prostredí vykonaný experiment s nastavením hyper-parametrov a testovali sme ich dopad na rýchlosť. Tieto nastavenia sa dajú aplikovať však aj na iné prostredia. Ako druhé prostredie bolo vybrané prostredie Lunar Lander, s ktorým sme porovnali výsledky. Toto prostredie je komplexnejšie a nie je náchylné na uviaznutie v lokálnom minime (ako Mountain Car). Na experiment sme vybrali model DDPG bez modulov, nakoľko agent využíva všetky testované parametre. Prostredia boli považované za vyriešené, keď agent dosiahol 85% z maximálnej odmeny prostredia. Pri zmenách testovaných parametrov boli agentove ostatné parametre nezmenené (nastavené podľa testovacích predpokladov na začiatku kapitoly).

Prostredia spojité	Koeficient učenia		Veľkosť dávky		Diskontný faktor		Veľkosť NN	
	Hodnota	Ep.	Hodnota	Ep.	Hodnota	Ep.	Hodnota	Ep.
Mountain Car	0.05	211	128	61	0.99	88	1024	107
	0.0035	178	64	88	0.90	134	512	99
	0.0001	88	32	99	0.1	548	256	88
Lunar Lander	0.05	> 2000	128	872	0.99	1102	1024	801
	0.0035	1350	64	986	0.90	1358	512	872
	0.0001	1102	32	1102	0.1	> 2000	256	1102

Tabuľka 11 Testovanie vplyvu hyper-parametrov na rýchlosť nájdenia riešenia

Vplyv koeficientu učenia je podobný ako pri učení štandardných NN. Pri nižšej hodnote je učenie pomalšie ale stabilnejšie, pri väčších hodnotách zase naopak rozkolísanejšie ale riešenie môže byť nájdené rýchlejšie, alebo vôbec nemusí byť nájdené. Nastavenie tohto koeficientu podľa výsledkov experimentu je najvhodnejšie na minimálnej hodnote 0.001 alebo menšej. Hlavným dôvodom je, že v spojitých prostrediach je nutné predpovedať hodnoty akcií. Určená hodnota je produkovaná na základe vnútorných parametrov NN (synaptických váh). Pri učení s vysokým koeficientom učenia je vplyv

gradientu na zmenu váh väčší, čo spôsobí skokové učenie a teda aj skokové predpovedanie hodnôt týchto akcií. Agent teda môže doslova preskočiť určité vhodné hodnoty akcií potrebné na vyriešenie problému. V prostredí Mountain car je však vhodné použitie skokového koeficientu učenia, nakoľko agent potrebuje čo najskôr nájsť riešenie aby neuviazol v lokálnom minime.

Ďalším testovaným parametrom bola veľkosť dávky. Jej veľkosť je teoreticky neobmedzená. Z praktického hľadiska treba pri jej určovaní dbať na výkon použitého softvéru a hardvéru. Podľa experimentu boli otestované postupne väčšie veľkosti dávok. Vplyv veľkosti dávok je, nie len podľa tabuľky vyššie, zrejmý. Napríklad, použitie 100 učiacich epizód s veľkosťou dávky 32 spôsobím zmenu váh 32 000 približne krát, pričom veľkosť dávky 128 až približne 128 000 krát v rámci jednej epizódy. Z tabuľky ďalej môžeme vidieť, že vo veľkom prostredí ako je Lunar Lander je vhodné použiť najväčšiu možnú veľkosť dávky. Treba však brať aj v úvahu reálny čas na dosiahnutie výsledku. Pri väčšej veľkosti dávky je aj učenie časovo náročnejšie (epochy trvajú dlhšie). V prevedenom experimente pri veľkosti dávky 32 trvalo agentovi DDPG 986 epizód na vyriešenie približne 3 hodiny a pri veľkosti dávky 128 trvalo síce 679 epizód na vyriešenie, ale časovo učenie agentovi zabralo 5 hodín a 30 minút.

Samostatne boli testované inicializácie váh. Pre otestovanie ich vplyvu boli použité taktiež dve prostredia. Namiesto prostredia Lunar Lander však bolo použité prostredie Cart Pole. Nasledujúca tabuľka obsahuje výsledky testovania možného vplyvu inicializácie synaptických váh NN na rýchlosť dosiahnutia riešenia. Ostatné faktory, ktoré by mohli mať vplyv na rýchlosť riešenia, boli zafixované na rovnaké hodnoty pre obidve prostredia ako v predošlom experiment. Pre experiment bol vybratý diskrétny agent aktér – kritik.

Počiatočná inicializácia váh	Mountain Car (D)	Cart Pole (D)
Jednotková	501	1452
Rovnomerné rozdelenie	411	888
Nulová	597	1254
Náhodná (rovnaká násada)	489	1102

Tabuľka 12 Vplyv inicializácie NN na rýchlosť získania riešenia

Z tabuľky môžeme vidieť, že nie je jasný vplyv inicializácie synaptických váh NN na rýchlosť dosiahnutia riešenia. Generovanie hodnoty váh z rovnomerného rozdelenia je podľa tabuľky najvhodnejším spôsobom. Predpokladali sme, že v prostredí Mountain Car, ktorého špecifickou vlastnosťou je možnosť uviaznutia agenta, pokiaľ agent nedosiahne rýchlo cieľ, môže inicializácia váh dopomôcť k rýchlosti nájdenia riešenie. Istá forma

náhodnosti vo váhach síce dopomôže ku mierne väčšiemu kolísaniu auta v počiatkoch trénovania, avšak nemá vplyv na získanie rýchlosti riešenia.

6.2.3 Zhrnutie a celkové výsledky experimentu

Prvá časť experiment ukázala, že použitie cieľovej a trénovacej NN je vhodný koncept na zlepšenie stability učenia. S použitím tohto modulu sa minimalizuje efekt zvyšovania cieľovej hodnoty v priebehu jednej epizódy. Vďaka tomu je agent schopný učiť sa podľa správnych hodnôt a vplýva na neho menšie riziko vzniku prípadnej oscilácie.

Aktualizovanie váh jemnejšie podľa parametra je taktiež užitočným modulom. Z výsledkov však nevyplýva výrazné zlepšenie pri každom type úlohy. Je vhodnejšie do prostredí väčších rozmerov, napríklad do prostredia Lunar Lander. Toto vylepšenie je však vhodné použiť pri agentoch, ktorí sa učia v rámci epizódy (online), a ktorí nie sú obmedzení časom na získanie riešenia. Vtedy je žiadúce zjemňovanie nadobudnutých vedomostí vo forme synaptických váh NN za účelom ich pomalšej zmeny, aby agent preskúmal aj prípadné čiastkové riešenia.

Spomenuté modulárne riešenia je možné aj kombinovať spoločne. Týmto spojením agent dokáže eliminovať nestabilitu, no však treba počítať so spomalením učiaceho cyklu. Toto spojenie by sme odporúčali pri prostrediach s veľkým rozmerom akcií a stavov, kde je nutné z povahy úlohy preskúmanie aj jemne odlíšených stavov.

Z pohľadu zvýšenia rýchlosti nájdenia riešenia zo spomenutých modulov je vhodné využitie zdieľaných parametrov sietí u agentov, u ktorých to je možné (napríklad Aktér – kritik, DDPG). Agenti, ktorý používajú jednu NN (napr. DQN), môžu urýchliť nájdenie riešenia tým, že agentovi dovolia väčšie množstvo prieskumov. Agent môže pri vhodnejšom spôsobe preskúmavaní podľa typu prostredia rýchlejšie nájsť vhodný smer vývoju učenia. V experimente bolo prestavených viacero spôsobov tvorby prieskumov. Najlepšími prostriedkami na tvorenie prieskumov je použitie epsilon náhody alebo generovanie hluku.

Epsilon náhoda je vhodná do diskrétnych prostredí, najmä do takých, kde sa pracuje s pravdepodobnosťami akcií. V týchto podmienkach hluk nedosahoval tak vhodné vlastnosti ako epsilon náhoda. Je to spôsobené tým, že hluk sa pripočítava ako zložka do hodnôt produkovaných NN a jeho hodnota nemusí vyvolať prieskum (navyše vyvoláva veľké množstvo prieskumov dlhší čas), pričom epsilon náhoda vyvolá najväčšie množstvo prieskumov na začiatku učiaceho cyklu. Pri rozsahovo menších prostrediach to je prijateľnejšie.

Pri spojitých prostrediach je najlepšou možnosťou generovanie hluku. Hlavne kvôli tomu, že v tomto prípade je výstup NN číselná hodnota, určujúca napríklad určitú rýchlosť. Zašumením tejto hodnoty dosiahneme novú hodnotu, ktorá pôvodnú rýchlosť zmenší alebo zväčší. Vďaka tomu je veľkosť rýchlosti po započítaní zložky šumu aj naďalej v okolí pôvodne predpovedanej hodnoty. Inými slovami, hluk zmenší alebo zväčší predpovedanú hodnotu, čím jemne upraví agentovu vybratú akciu a vyvolá prieskum.

6.3 Experiment zmeny spôsobu odmeňovania

Odmena môže nadobúdať rôzne hodnoty v rozličných prostrediach. V prostrediach implementovaných v tejto práci je odmena prostredia popísaná v nasledujúcej tabuľke. V ďalších odsekoch bude táto odmena označovaná ako absolútna odmena prostredia.

Prostredie diskrétne	Priebežná odmena	Odmena za vyriešenie
Cart Pole (D)	1 za udržanie palice do 12 stupňov	Žiadna dodatočná odmena
Lunar Lander (D)	Počítané z priblíženia k cieľu (100 – 140) v súčte krokov k pristátiu	100 za pristátie v zóne +10 za každú nezlomenú nožičku)
Mountain Car (D)	-1 pre krok každý krok, 0 za krok na ľavý kopec	Žiadna dodatočná odmena
Prostredie spojité		
Lunar Lander (C)	Počítané z priblíženia k cieľu (100 – 140) v súčte krokov k pristátiu	100 za pristátie v zóne +10 za každú nezlomenú nožičku
Mountain Car (C)	-1 pre každý krok, 0 za krok na ľavý kopec	100 na kopci – štvorcová suma akcií

Tabuľka 13 Popis absolútnej odmeny v prostrediach poskytovaných aplikáciou

Ako môžeme v tabuľke vyššie vidieť, prostredie Cart Pole poskytuje absolútnu odmenu zhodnú s oceňovaním krokov bez kolízií (viď. nižšie). Prostredia Lunar Lander alebo Mountain Car sú prostredia s problémom tzv. kontroly, čiže ako absolútnu odmenu poskytujú pomalé klesanie odmeny. Pre pripomenutie, prostredia pri nastavení absolútnej odmeny prepočítavajú odmenu podľa orientácie voči cieľu, napríklad v prostredí Lunar Lander je priebežná odmena počítaná s priblíženia k pristávacej ploche, uhlovej orientácie modulu a rýchlosti klesania.

6.3.1 Normovanie odmeny a gradientu

Od veľkosti odmeny závisí aj veľkosť gradientu. Čím je však odmena väčšia, tým aj gradient je väčší. Opäť môžeme čeliť problému oscilovania, respektíve obídenia vhodného

riešenia. Tomuto javu však môžeme zamedziť tým, že hodnotu gradientu znormujeme do intervalu <-1,1>. Predpokladom je dosiahnutie zredukovania veľkosti zmien pri učení obmedzením gradientu. Učenie sa naďalej bude vyvíjať potrebným smerom (závisí od smeru gradientu) len jeho krok bude menší.

Neznamená to však, že redukujeme koeficient učenia. Zníženie koeficientu učenia spomalí priebeh učenia, v zmysle, že bude sa agent učiť pomalšie. Normovanie gradientu spôsobí, že agent sa bude učiť pomocou menšieho gradientu, ktorý bude rozpočítaný pomocou koeficientu učenia medzi neuróny.

6.3.2 Oceňovanie úspešného riešenia so záporným sťahovaním odmeny

V tomto experimente sa budeme sledovať vplyv zmeny oceňovania agenta na jeho učiaci proces. Toto nastavenie je vhodné pre problémy kontroly (napríklad objektov – lunárny modul, auto). Agentovi je za každý krok, ktorým nevyrieši prostredie, uberaná odmena -1. Dosiahnutie kolízie je oceňované absolútnym číslom prostredia, napríklad v prostredí Lunar Lander odmenou –100. Za vyriešenie agent dostane väčšiu odmenu, ktorá je výrazne odlíšiteľná od dosiahnutých odmien.

Tento spôsob je vhodný pre prostredia Lunar Lander a Mountain Car. V týchto prostrediach je aj čiastočne implementovaný, avšak agent má uľahčenú úlohu tým, že odmena v sebe obsahuje isté dodatočné informácie (počítaná z polohy voči cieľu). Tento spôsob odmeňovania túto informáciu zanedbáva a každým krokom agentovi konštante prostredie odpovedá hodnotou -1.

6.3.3 Kladné oceňovanie krokov bez kolízií

V úlohách kontrolovania určitého objektu je možné oceňovanie agenta aj za nespôsobenie kolízie. V niektorých prostrediach by to mohlo byť výhodné. Napríklad v prostredí Cart Pole, kde je úlohou agenta udržať vzpriamenú tyč, môže byť takéto oceňovanie vhodné. Je to však alternatívny spôsob odmeňovania. Vo veľkom množstve prostredí je agent motivovaný tým, že za každý krok, ktorým sa nedostal k cieľu, je mu uberaná odmena.

V menších prostrediach z úlohami stálej kontroly by však mohla byť alternatívna forma spomenutého kladného odmeňovania vhodná. Porovnáme v experimente vplyvy kladného spôsobu odmeňovania voči spomenutému zápornému.

Predošlý spôsob odmeňovania je istým protikladom k tomuto spôsobu. Avšak je určený pre problém kontroly, v ktorom je nutné nájsť cieľ. Pri kladnom odmeňovaní krokov

bez kolízií nie je myšlienkou hľadať konečný cieľ, ale udržať modul stabilný. Prostredie Cart Pole by mohlo byť vhodné pre takýto spôsob odmeňovania, pretože cieľ nie je stanovený (cieľ je totiž formulovaný ako udržanie tyče vzpriamenie bez definovania cieľovej pozície).

6.3.4 Zhrnutie a celkové výsledky experimentu

Experimenty boli vykonané na jednoduchšom prostredí Cart Pole, aj na prostredí Lunar Lander. Znormovanie odmeny bolo vykonané pomocou funkcie hyperbolického tangensu podľa nasledujúceho predpisu.

$$tahn\ (x) = \frac{\sinh(x)}{\cosh(x)} \tag{23}$$

Funkcia zabezpečí prevedenie hodnôt do intervalu <-1,1> so zachovaním príslušného znamienka. Experiment prebehol s agentom DQN. Výsledky boli porovnané s bežným použitím absolútnej odmeny, kde bolo hlavným kritériom počet potrebných epizód na riešenie.

	Normovaná odmena	Kladné oceňovanie krokov bez kolízií	Oceňovanie úspešného riešenia so záporným sťahovaním odmeny	Absolútna odmena poskytovaná prostredím
Cart Pole (D)				
DQN (D)	465	425	450	425
Lunar Lander (D)				
DQN (D)	553	894	542	580

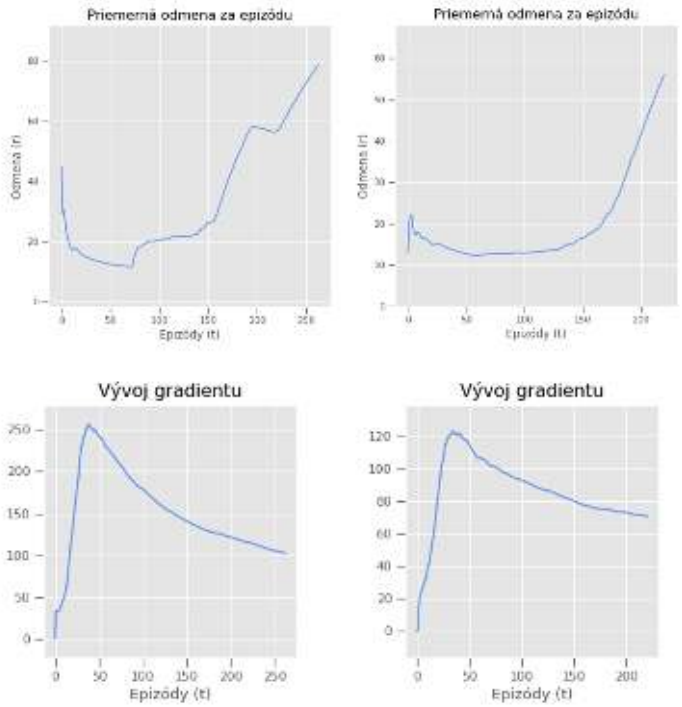
Tabuľka 14 Porovnanie rôznych spôsobov odmeňovania proti klasickému spôsobu na agentovi DQN v prostrediach Cart Pole (D) a Lunar Lander (D)

Podľa tabuľky môžeme vidieť, že prostredie Cart Pole poskytuje pomalú kladnú odmenu. Oproti ostatným formám odmeny, normovaná odmena spomalila model v rýchlosti nájdenia riešenia. Môže to byť spôsobené tým, že agent dostáva len kladnú odmenu (za udržanie palice). Agentov vývoj je teda ovplyvňovaný len kladnými odmenami, kde normovanie spôsobí len spomalenie učiaceho cyklu.

Pre prostredie Lunar Lander sú výsledky trošku iné. Použitie normovanej odmeny pomohlo agentovi zmenšiť počet epizód na dosiahnutie riešenia, avšak použitie kladnej odmeny po každom kroku spôsobilo, že agent uprednostnil vznášanie sa vo vzduchu, namiesto pristátie v cieli. Agent sa totižto naučil, že nepotrebuje nájsť cieľ, ale keď sa dlho udrží vo vzduchu, pričom dostáva kladnú odmenu, bude najvýhodnejšie.

Zaujímavosťou môže byť vývoj priemernej odmeny v porovnaní s vývojom gradientu pri alternatívnych spôsoboch odmeňovania. Porovnanie bolo vykonané

s normovanou odmenou voči klasickej absolútnej odmene Tento vývoj je zobrazený na nasledujúcom obrázku.



Obrázok 20 Porovnanie klasického odmeňovania (vľavo) proti normovanému odmeňovaniu (vpravo)

Z obrázku vyššie môžeme vidieť, že normovaný vývoj odmien má hladší priebeh ako absolútna odmena. To bolo aj očakávané. Čo však ovplyvní takto normovaná chyba, je gradient. Ovplyvní jeho veľkosť, podľa ktorej prebieha učiaci cyklus v algoritmoch používajúcich gradient.

Aj pri algoritmoch používajúcich funkciu ohodnotenia je normovanie chyby vítané. Podľa odmeny a minulej hodnoty je priamo menená hodnota v príslušnom stave. Pri určení hraníc odmeny je vývoj učenia stabilnejší. V tomto prípade sa však môže pri niektorých prostrediach predĺžiť učiaci proces. Avšak pri stabilnejšom učení sa minimalizuje riziko preskočenia jedného z vhodných riešení.

Taktiež to potvrdzuje tabuľka č.14, kde môžeme vidieť, že normovaná odmena vo väčšom prostredí pomohla zrýchliť proces učenia. V prostredí Cart Pole, ktoré je rozsahom menšie, normovanie odmeny nezmenšilo počet potrebných epizód. Normovanie odmeny je teda potrebné zvážiť v závislosti na prostredí.

7 Vylepšenie modelu DDPG na základe výsledkov experimentov

Zo spomenutých výsledkov experimentov sme sa pokúsili vylepšiť jeden zo základných modelov. Z ohľadom na dobré vlastnosti DQN, vďaka využitiu skúseností v učiacom cykle, sme sa pokúsili rozšíriť tento model na spojitý priestor. Rozšírili sme ho na agenta DDPG pomocou Aktér – kritik metodiky. Rozdelenie NN agenta do dvoch komponentov a možnosť učenia sa online bola z pohľadu výsledkov v experimentoch najvhodnejšia.

Ako náhodnú zložku na generovanie prieskumov bol použitý šum. V spojitých prostrediach je považovaný za vhodnú voľbu. V experimente porovnávajúcom možnosti prieskumov dosahoval vhodné výsledky a je ľahko implementovateľný do spojitého prostredia, nakoľko dokáže generovať také množstvo zložiek, koľko je potrebných.

Hyper - parametre boli nastavené podľa výsledku experimentu vyššie. Koeficient učenia aktéra bol použitý 0.0001, nakoľko táto hodnota je veľmi malá a nespôsobuje osciláciu. Pre kritika bola veľkosť koeficientu 0.0005, pre jemné zväčšenie cieľových hodnôt pre aktéra. Veľkosť pamäte sme použili 1 000 000 položiek. Veľkosť dávky sme použili 124 prvkov.

Ako počiatočné nastavenie synaptických váh bolo použité náhodné rozdelenie. Z experimentov nevyplýval jasný vplyv na výsledky agentov, preto bolo ponechané toto základné nastavenie.

Model používa na zvýšenie stability cieľovú – zdrojovú NN pre aktéra aj kritika samostatne. Presunutie váh prebieha po každej epizóde. Použité je aj jemné aktualizovanie váh s parametrom $\boldsymbol{\tau} = 0.1$. Pretože sme v spojitom prostredí, pridanie jemného aktualizovania stabilizuje vývoj gradientu ako aj vývoj predpovedaných hodnôt NN aktéra.

Aj keď je model aplikovateľný len na spojité prostredie, využíva zdieľané parametre sietí. Spolu s jemným aktualizovaním váh dokáže byť rozložená veľkosť gradientu do menších blokov, čo predchádza oscilovaniu. Pridanie modulu zmenší počet parametrov, čo je z hľadiska väčšieho počtu neurónov NN a veľkosti prostredia vhodné z hľadiska rýchlosti nájdenia riešenia.

Použité boli trojvrstvové NN pre aktéra aj kritika o rovnakej veľkosti (vrstvy o veľkosti 1024 neurónov). Tieto nastavenia boli použité pre testované prostredia. Na tomto prostredí sme model DDPG s modulmi porovnali voči ostatným modelom. Modely sa porovnávali na základe počtu epizód potrebných na dosiahnutie 85% z maximálnej odmeny prostredia za posledných 50 po sebe idúcich epizód.

Prostredie	DDPG (C) – bez modulov	DDPG (C) – s modulmi	AC (C) – s modulmi
Lunar Lander (C)	1102	872	1591
Mountain Car (C)	88	61	510

Tabuľka 15 Porovnanie počtu potrebných epizód na nájdenie riešenia modelu DDPG s modulmi na spojitých prostrediach voči iným modelom

Prostredia majú aj svoje diskrétne verzie a preto sme porovnali výsledky DDPG agenta na spojitých prostrediach proti výsledkom diskrétnym agentom.

Prostredie	DDPG (C) – bez modulov	DDPG (C) – s modulmi	DQN (D) – s modulmi	AC (D) – s modulmi
Lunar Lander (D)	1102	872	549	1012
Mountain Car (D)	88	61	301	401

Tabuľka 16 Porovnanie modelu DDPG proti AC, DQN na diskrétnych prostrediach

Z porovnania môžeme vidieť, že vylepšený model DDPG je vhodným agentom. Dosahuje zrovnateľné výsledky na spojitých a zložitejších prostrediach ako diskrétne modely na diskrétnych zmenšeninách týchto prostredí.

V modely sme ďalej testovali normovanú odmenu, čo však nezmenilo markantne rýchlosť nájdenia riešenia. Nasledujúca tabuľka zobrazuje porovnanie.

Prostredie	Kladné oceňovanie krokov bez kolízií	Normovaná odmena	Absolútna odmena prostredia
Lunar Lander (C)	921	969	872
Mountain Car (C)	73	205	61

Tabuľka 17 Porovnanie použitia normovanej odmeny voči pomalej zápornej na agentovi DDPG s modulmi

V prostredí Lunar Lander spôsobila normovaná odmena spomalenie učenia, pretože pri jej použití sa zmení cieľ agenta. Nesnaží sa už totiž o rýchle pristátie, ale skôr sa vznáša vo vzduchu, nakoľko za pristátie je rozdiel v odmene menší.

Model sme ďalej porovnali s najlepšími výsledkami v prostrediach iných agentov, ktoré sú voľne dostupné [28].

Prostredie	DDPG (C) – s modulmi	najlepšie výsledky
Lunar Lander (C)	872	100 - 1500
Mountain Car (C)	61	120 - 140
Bipedal Walker (C)	2480	1000 - 2500

Tabuľka 18 Porovnanie modelu DDPG voči najlepším výsledkom v daných prostrediach

Podľa tabuľky môžeme vidieť, že agent DDPG dosahuj výsledky približne ako tabuľkové hodnoty iných modelov. V prostredí Mountain Car je lepší, ale prostredie je špecifické a výsledku modelu, s ktorými je porovnávaný, môžu byť dosiahnuté na základe inej metodiky.

Najlepšie výsledky, s ktorými je agent porovnávaný, sú určené rozmedzím. Je to preto, že niektoré výsledky sú dosiahnuté skôr a k ďalším je veľká priepasť, čo môže byť spôsobené pridaním určitej formy analýzy prostredia do modelu agentov.

V porovnaní s agentom DPPG, ktorý bol vytvorený bez modulárnych vylepšení, dosahuje vylepšený model lepšie výsledky. Podľa týchto výsledkov môžeme potvrdiť zlepšenie vlastností modelu ich použitím.

7.1 Použitie modelu DDPG na extrémne ťažkom prostredí

Agent bol nakoniec otestovaný na prostredí Bipedal Walker, ktoré je považované za veľké a náročné prostredie so širokou množinou stavov a akcií. Akcie a stavy popisujú nasledujúce tabuľky.

Názov zložky stavu	Interval
Uhol pohľadu	$< 0,2\pi >$
Kontakt ľavej nohy so zemou	< 0, 1>
Kontakt pravej nohy so zemou	< 0, 1>
Rýchlosť po osy X	< -1, 1>
Rýchlosť po osy Y	< -1, 1>
Uhoľ panvového kĺbu ľavej nohy	(- ∞, ∞)
Uhoľ panvového kĺbu pravej nohy	(- ∞, ∞)
Uhoľ kolenného kĺbu ľavej nohy	(- ∞, ∞)
Uhoľ kolenného kĺbu pravej nohy	(- ∞, ∞)
Uhlová rýchlosť panvového kĺbu ľavej nohy	(- ∞, ∞)
Uhlová rýchlosť panvového kĺbu pravej nohy	(- ∞, ∞)
Uhlová rýchlosť kolenného kĺbu ľavej nohy	(- ∞, ∞)
Uhlová rýchlosť kolenného kĺbu pravej nohy	(- ∞, ∞)
10x hodnota vnútorného senzoru	(- ∞, ∞)

Tabuľka 19 Stavy a ich rozsah hodnôt pre prostredie Bipedal Walker (C)

Názov zložky akcie	Interval

Bedro kolena 1(krútiaci moment / rýchlosť)	<-1, 1>
Koleno 1 (krútiaci moment / rýchlosť)	<-1, 1>
Bedro kolena 2 (krútiaci moment / rýchlosť)	<-1, 1>
Koleno 2 (krútiaci moment / rýchlosť)	<-1, 1>

Tabuľka 20 Akcie a ich rozsah hodnôt pre prostredie Bipedal Walker (C)

Zložiek stavu je dohromady 24. V tabuľke sú zoskupené podľa veľkosti hodnôt ktoré nadobúdajú. Počet možných akcií je 4. Všetky hodnoty sú v rozsahu <-1,1>. Agent môže hodnotami akcií ovádať ľudské nohy, podľa zmeny v uhle a rýchlosti v kĺboch panvy a kolien.

Agent DDPG mal pre prostredie Bipedal Walker uspôsobené nasledujúce parametre. Veľkosť NN bola zvolená na 2048 pre každú skrytú vrstvu. Koeficient učenia aktéra bol nastavené podľa experimentov na 0,0001, ale koeficient učenia kritika bol upravený na 0,0001. Veľkosť pamaťe bol použitý 200 000 a veľkosť dávky 64, pretože agent sa učí chodiť, čo je dlhodobý procoes, a teda veľká pamať by mohla obsahovať rozhodnutia, ktoré v postupe času už nemusia byť vhodné pre tréning. Východzali sme z predpokladu, že čím ďalej sa agent učí, tým je jeho chodenie vylepšené a už len vylepšuje, čo sa doposiaľ naučil a preto nepotrebuje veľkú pamäť. Všetky úpravy boli urobené kvoli tomu, že prostredie je veľmi veľké a aj malé odchýlky hodnôt akcií môžu spôsobiť pád agenta,

Agent DDPG dokázal prostredie vyriešiť 2480 epizód. Čas, ktorý potreboval na trénovanie bol 123 hodín a 37 minút. Jeho výsledky zobrazujú nasledujúce obrázky.

Obrázok 21 Vývoj priemernej odmeny

Priemerný vývoj odmeny (obrázok vyššie) zobrazuje vývoj odmeny od prvej epizódy. Ako môžeme vidieť, v počiatku agent preskúmaval okolie rapídne, a preto nedosahoval zlepšenie rýchlo. Približne od epizódy 200 sa osvedčili moduly zabezpečujúce

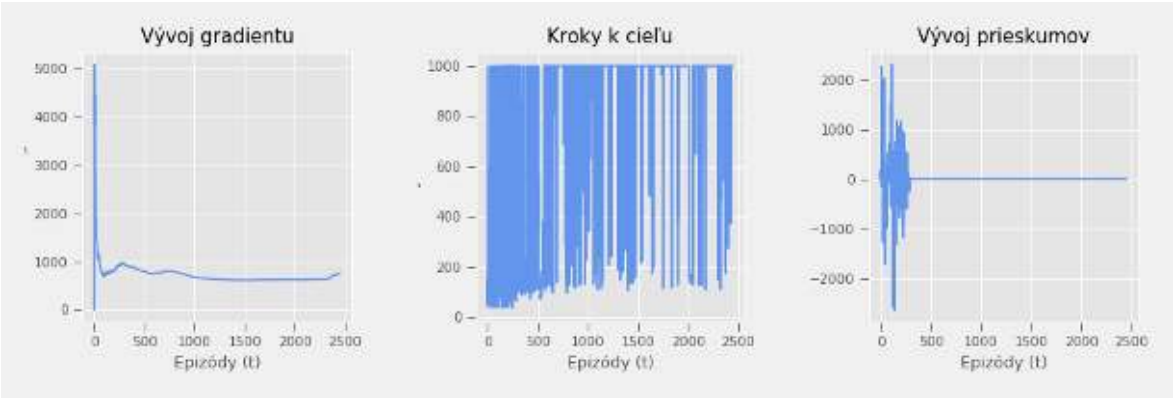
stabilitu (zdrojová – cieľová NN a jemné aktualzovanie váh) a gradient začal pomaly, ale za to stabilne, rásť.

Obrázok 22 Vývoj absolútnej odmeny za posledných 50 epizód (fáza trénovania – vľavo, fáza testovania – vpravo)

Na nasledujúcich obrázkoch môžeme vidieť ostatné štatistiky. Na prvom je zobrazený vývoj gradientu. Ako môžeme na obrázku vidieť, gradient na začiatku vysoko stúpol, no následne po stabilizovaní pomaly klesal.

Zaujímavá je aj štatistika počtu krokov k cieľu, kde môžeme vidieť, že agent zo začiatku veľmi veľa krát padal. Postupom času rástol počet krokov, a teda agent urobil viacej k krokov v prostredí a dosiahol lepšie výsledky.

Poslednou štatistikou je vývoj výskumov. Použitý bol spôsob hluku so zahrievaním 300 epizód. Po týchto epizódach agent už nerobil prieskumy. Nastavenie v tomto prostredí umožní agentovi v počiatku vykonať prieskumy, a následne agent už len prispôsobuje svoje synaptické váhy, aby určovali presnejšiu hodnotu akcie.



Obrázok 23 Sekundárne štatistiky

Agent vyriešil dané prostredie. Podmienkou zastavenia učenia bolo dosiahnutie 85 % odmeny z maximálnej odmeny prostredia, čo predstavuje odmenu 255. Pri testovaní agent presiahol maximálnu odmenu (počítanú aj podľa rýchlosti) a prešiel celú mapu.

8 Výsledky práce a diskusia

Priložená aplikácia k tejto práci obsahuje implementáciu spomenutých agentov. Po vybratí konkrétneho agenta sa zobrazia nastavenia vnútorných parametrov. Tieto parametre sú odporúčané z hľadiska vykonaných experimentoch na agentoch. Avšak nemusia byť vzhľadom na všetky prostredia vhodne použiteľné, preto je v aplikácii umožnená zmena týchto parametrov. V nasledujúcich častiach tejto kapitoly budú popísané výsledky experimentov, ktoré dopomôžu pri voľbe parametrov, prípadne voľbe agenta pre určité prostredie. Testované parametre spolu s testovacími predpokladmi sú popísané v časti Experimenty. Na konci kapitoly sú popísané výsledky vylepšeného algoritmu a sú spomenuté výhody a nevýhody RL algoritmov.

8.1 Odporúčania

V prvom experimente bolo cieľom nájdenie vhodných vylepšení modelov, ktoré by pomohli stabilizovať a zrýchliť učiaci cyklus.

8.1.1 Stabilita

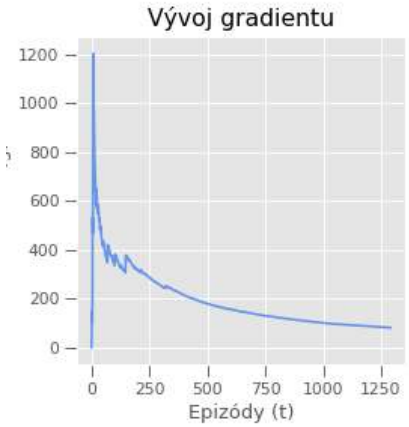
Stabilizovanie vývoju gradientu počas učenia je vylepšením, ktoré pomáha RL agentom v učení. Totižto, učenie odmenou je odlišné od klasického spôsobu učenia v tom, že dopredu nedisponujeme výslednými hodnotami akcií v stavoch – učiacimi vzormi. Po vykonaní akcie v stave vieme len to, akú odmenu za to agent dostane. Odmena je aj tak v čase rôzna pre rovnaké stavy a ich akcie, pretože je určovaná v závislosti k pozícii agenta k cieľu a diskontovanej budúcej odmeny na základe vybranej akcie, ktoré sú variabilné. V klasickom učení je výsledná hodnota nemenná. Spomenutá odmena môže mať rôzne veľkosti (aj pre malé zmeny v hodnotách akcií), čo môže spôsobiť nestabilitu v učení. V prvej časti experimentu prvého experimentu boli otestované dva spôsoby, ako zaručiť stabilitu v učení.

Použitie modulu cieľovej - zdrojovej NN výrazne napomáha stabilite vývoj gradientu aj v spojitých, aj diskrétnych prostrediach. Na obrázku č. 18 je zobrazené porovnanie vývoju gradientov s použitím a bez použitia tohto modulu. Z obrázku vyplýva, že použitie modulu cieľovej – zdrojovej NN výrazne zjemní vývoj gradientu, čím sa aj učiaci proces skvalitní. Tabuľka č.6. ukazuje, že tento modul nemá negatívny vplyv na rýchlosť učenia.

V druhej časti prvého experimentu bol testovaný vplyv jemného aktualizovania váh pri presune do cieľovej NN na stabilitu učenia agenta. Použitie toho modulu vyrovnalo vývoj gradientu podstatne viacej ako samostatný modul cieľovej – zdrojovej NN, čo dokazuje aj obrázok č. 18. Čas v epizódach sa však predĺžil, pretože gradient bol redukovaný parametrom presunu, čo spôsobilo pomalšie konvergovanie. V malom diskrétnom prostredí, napríklad Cart Pole, je použitie jemného aktualizovania váh z pohľadu pomeru stabilita – rýchlosť zbytočné. V diskrétnych prostrediach menších rozmerov by sme odporúčali podľa výsledkov nepoužiť tento modul.

Avšak, svoje využite nájde v spojitých prostrediach. V týchto prípadoch, hlavne ak je veľký priestor akcií v prostredí, agent čelí dlhšiemu učeniu. Agent hľadá v prostrediach čo najlepšie hodnoty akcií, ktorých hodnota sa určuje vnútornými váhami NN. Tieto váhy sa prispôsobujú na základe odmeny. Agent vykoná v stave akciu s určitými hodnotami. Po získaní odmeny, zmení váhy NN. Princíp je však vo viacnásobnom a pomalšom prispôsobení tejto hodnoty, pretože v celej politike prechodov k cieľu dokáže aj malá odchýlka v jednom stave a akcii spôsob iný prechod do iných stavov, kde už veľká hodnota nemusí byť vhodná. Pomalšie približovanie správnym smerom je v hodné pri dosahovaní určitého cieľu, k čomu prispieva aj jemné aktualizovanie váh.

Výsledok môžeme vo väčšom spojitom prostredí vidieť najlepšie na agentovi DDPG v spojitom prostredí Lunar Lander na nasledujúcom obrázku.



Obrázok 24 Vývoj gradientu agenta DDPG v prostredí Lunar Lander

8.1.2 Rýchlosť

Nájdenie riešenia v prostrediach je v niektorých prípadoch náročnejšie a v iných zase jednoduchšie. Agenti RL v menších prostrediach nájdu riešenie rýchlejšie, pretože tieto

prostredia obsahujú menší počet možných stavov a akcií, a v dokážu nájsť riešenie aj hrubou silou. Vo väčších prostrediach trvá nájdene riešenia agentom vo všeobecnosti dlhšie. Potreba nájdenia rýchlejšieho riešenia na veľkom a komplexnom prostredí je v dnešnej dobe veľmi podstatná. Pomocou niektorých nastavení je možné zrýchliť proces učenia napríklad v NN agenta pomocou hyper – parametrov alebo zmenšenia počtu parametrov ich zdieľaním.

Zdieľanie parametrov už podľa predpokladu by malo zrýchliť nájdenie riešenia. Potvrdil to aj experiment. Tento spôsob totižto výrazne redukuje počet trénovaných parametrov. Modely preto dokážu rýchlejšie konvergovať hlavne preto, že vo všeobecnosti majú NN s menším počtom parametrov, ale aj preto, že aplikujú gradient viacnásobne na tie isté parametre.

Ďalšou skúmanou možnosťou zrýchlenia nájdenia riešenia bolo zmenenie spôsobu preskúmavania okolia. V diskrétnom prostredí podľa výsledkov experimentu najlepšie výsledky dosiahol spôsob epsilon prieskumov. Diskrétne prostredia sú vo všeobecnosti menšie a sú založené na pravdepodobnosti výberu akcie. Použitie epsilon prieskumov je vhodné preto, že rozhodne o výbere prieskumu bez ohľadu na predpovedané pravdepodobnosti. Napríklad použitie hluku je závislé od predpovedanej pravdepodobnosti. K tejto pravdepodobnosti je pripočítaná zložka hluku, ktorá však nemusí vyvolať prieskum (najmä pri vysokej pravdepodobnosti najvhodnejšej akcie a ostatných blízkych 0).

V spojitých prostrediach však najlepšie uplatnenie nájde použitie hluku. V týchto prostrediach musí agent hľadať najlepšie nastavenie hodnôt akcií. Napríklad v prostredí s jednou akciou v rozsahu <-1,1> je predpovedaná hodnota 0,3. Ak by sme považovali za výber prieskumov epsilon, vybratá by bola náhodná nezávislá hodnota, napríklad 0,6. Pri hluku sa započíta náhodná zložka do akcií, čo spôsobí, že v čase sa budú hodnoty menej a menej odlišovať od predpovedanej, a teda bude počet prieskumov klesať. Problém s využitím iného spôsobu ako hluku je v tom, že pri väčšom priestore akcií je obtiažne určovať hodnotu prieskumu. Hluk je pre tento spôsob ideálny, nakoľko je možné nastaviť zahrievaciu fázu (kde je zväčšená veľkosť prieskumov) a postupom času konverguje k 0, a teda už zachováva hodnoty zvolenej politiky.

Hyper – parametre sú taktiež dôležitým nastavením. Nastavenia parametrov NN môžu taktiež domôcť k zrýchleniu nájdenia riešenia. Hodnota koeficientu učenia je hlavným parametrom zodpovedným za rýchlosť učenia NN. Jeho hodnotu pri RL je však vhodné nastaviť na najnižšiu možnú hodnotu, napríklad 0,0001. Táto hodnota spôsobí, že

prispôsobenie synaptických váh NN bude v malých krokoch, čím sa minimalizuje riziko preskočenia riešenia.

Veľkosť NN bola určená dopredu. Podľa teórie sme použili trojvrstvové NN [103]. Meniť je však možné počet neurónov na vrstvách. Tento počet by sa mal odvíjať od zložitosti prostredia. Napríklad v malom diskrétnom prostredí, kde je počet stavov s kombináciou akcií, napríklad 10 000, môžeme predpokladať, že potrebujeme aspoň 10 000 trénovateľných parametrov. Podľa podobných úvah je potrebné nastaviť počet neurónov v závislosti na prostredí.

Ďalším skúmaným parametrom je veľkosť dávky. Testovaná bola na modeloch, ktoré používali pamäť skúseností. Pri nastavovaní veľkosť dávky je nutné byť dbať o to, akú výpočtovú silu HW máme k dispozícii. Treba uvážiť to, či je HW schopný uchovať dávku v RAM pamäti (hlavne ak používa agent vstup vo forme obrázkov) a koľko dokáže pamäť uchovať vzoriek.

Hodnota diskontného faktoru je odporúčaná čo najbližšie k 1, napríklad 0,99. Táto hodnota spôsobí uprednostnenie skorších hodnôt oproti hodnotám v budúcnosti. Pri použití nižšej hodnoty považujeme odmenu v aktuálnom stave za podstatnejšiu ako očakávanú odmenu v budúcnosti. Pri hľadaní cieľu je najvhodnejšie použiť hodnotu blízku 1, avšak nie priamo 1, kvôli zlepšeniu konvergencie.

8.1.3 Spôsoby odmeňovania

Štruktúra odmeňovania v prostredí je tiež dôležitá. Motivácia agenta k vyriešeniu prostredia môže byť upravovaná spôsobom odmeny. V prípadoch, kde je úlohou agenta vyriešiť problém, je vhodné oceňovať nízkou zápornou odmenou každý jeho krok, ktorý nevedie k nájdeniu riešenia. To spôsobí, že agent sa neuspokojí s riešením v aktuálnom stave, ale bude sa snažiť dosiahnuť cieľ, kde dostane veľkú kladnú odmenu.

Oproti tomuto klasickému štýlu odmeňovania sme testovali spôsoby normovanej odmeny, oceňovania len úspešných pokusov a oceňovanie krokov bez kolízie kladnou hodnotou. Pri probléme kontroly sme porovnali klasický spôsob odmeňovania oproti alternatívnemu, kde sme oceňovali agenta za nespôsobenie kolízie.

Podľa výsledkov experimentu, použitie klasického spôsobu odmeny je lepšie ako alternatívne. Oceňovanie malou kladnou hodnotou za nespôsobenú kolíziu spôsobí, že agent sa bude fixovať na stavy, kde má odmenu už overenú. Často nastane zacyklenie agenta len v známych stavoch, kde nenastáva kolízia a nedosiahne riešenie v prostredí.

Absolútna odmena prostredia je najlepšou voľbou. Je to tak preto, že prostredie počas kroku má v odmene zakódované agentove priblíženie k cieľu. Napríklad v prostredí Lunar Lander, je agent oceňovaný podľa uhlu k cieľu a rýchlosti klesania (pokým nedosiahne cieľ tak zápornou odmenou, avšak z odmeny je možné vyčítať, ako dobre sa približuje k cieľu podľa veľkosti odmeny blízkej 0).

Použitie normovanej odmeny však dosiahlo zaujímavejšie výsledky. V prostredí Lunar Lander je odmena definovaná tak, že agent dostáva kladnú alebo zápornú hodnotu prepočítanú zo smeru, rýchlosti a priblíženia k cieľu. Za dosiahnutie cieľu agent dostane veľkú kladnú odmenu. Pri normovaní sa však mení zadanie úlohy pre agenta. Treba myslieť na to, že pri normovaní odmeny je agent hnaný menšou silou k cieľu. Avšak, odmena ako jedna zo zložiek gradientu, sa po normovaní prejaví aj v zmenšení a stabilizovaní gradientu, čo môže v niektorých prostrediach (napríklad s problémom kontroly) byť užitočné.

Normovanie gradientu je zahrnuté v tejto kapitole, pretože do neho zasahuje aj získaná odmena. Normovanie gradientu využívajúceho v modely (napríklad DQN) Q - hodnotu spôsobí zmenšenie rozhľadu agenta do budúcnosti na základe odmeny. Výsledkom normovania je stabilizovanie učenia za cenu spomalenia učenia. Z hľadiska výpočtov je vhodnejšie však používať menšie hodnoty. Ďalšou výhodnou môže byť použitie v prostrediach s čiastkovými cieľmi, kde je agent kladne odmeňovaný aj za dosiahnutie čiastkového cieľu.

8.1.4 Vylepšený model

Podľa výsledkov experimentov a stochastickom určení vhodných modulov na zlepšenie agentov bol navrhnutý agent DDPG, ktorý využíva viaceré rozšírenia. Na štandardných prostrediach dosahoval lepšie výsledky ako štandardné modely, čo dokazujú aj tabuľky č. 14 a 15.

Model bol ďalej aplikovaný na extrémne ťažké spojité prostredie Bipedal Walker. V tomto prostredí dokázal nájsť riešenie v adekvátnom čase voči tabuľkovým výsledkom iných agentov na danom prostredí. Treba podotknúť, že parametre ostatných agentov neboli zverejnené a teda mohli použiť aj určitú analýzu prostredia, čo nebolo v tejto práci použité.

Agent využíval všetky moduly, ktoré sa podľa experimentov javili ako vhodné. Z hľadiska zlepšenia stability boli použité moduly cieľovej – zdrojovej NN a jemného aktualizovania váh spoločne. V prostredí Bipedal Walker je veľká dimenzia stavov a akcií, čiže stabilizovanie vývoja učenia je rozhodne dobrou vlastnosťou.

Pre zlepšenie rýchlosti učenia bolo použité zdieľanie parametrov sietí. Problémy, ktoré by mohli nastať v spojitom prostredí, boli minimalizované tým, že bol použití modul jemného prenosu váh. Forma prieskumov bola použitá formou hluku so zahrievacou fázou vo veľkosti 300 prvých epizód.

Nastavenie hyper – parametrov bolo urobené podľa experimentu č.2. Tieto parametre boli aplikované do modelu podľa výsledkov experimentov. Za spomenutie stojí veľkosť dávky 124, veľkosť pamäte 200 000 vzoriek, učiaci koeficient aktéra 0,0001, učiaci koeficient kritika 0,0005 a diskontný faktor 0,99.

Model dosiahol dobré výsledky, v porovnaní s tabuľkovými výsledkami modelov v prostredí Bipedal Walker. Cieľom v tejto úlohe je, aby sa agent naučil chodiť na dvoch nohách. Konečná odmena je stanovená vzdialenosťou, ktorú dokáže úspešne prekráčať bez spadnutia. V porovnaní dosahoval výsledky ako tabuľkové modely, avšak tieto modely mohli využívať dodatočné predspracované informácie o prostredí. Model DDPG použitý v tejto práci však môže byť použitý na väčšiu škálu prostredí s rôznym zameraním, na ktorých dosahuje lepšie výsledky ako iné modely.

8.2 Výhody RL

Podľa teórie RL, výhoda RL agentov je v tom, že pracujú na základe interakcie s prostredím. Nepotrebujú veľkú základňu označených dát pred spustením učiaceho cyklu a sú aplikovateľné na širokú škálu problémov. V neposlednom rade, interpretácie učiaceho cyklu a výsledkov agentov RL je človeku najzrozumiteľnejšia.

Po vykonaní všetkých experimentov a záverečného testovania modelu DDPG boli odsledované niektoré ďalšie výhody RL agentov, ktoré môžu dosiahnuť pridaním niektorých vylepšení. Učiacu fázu je možné zrýchliť pridaní pamäte skúseností, ktorá násobí učiaci proces. Modely dokážu riešiť aj extrémne náročné a komplexné prostredia v reálnom čase, čo nie je vždy možné pomocou štandardných NN. Ak je t aj možné, potrebné prostriedky na riešenie pomocou NN budú výrazne väčšie ako pri RL agentoch.

8.3 Nevýhody RL

Ako už býva zvykom u algoritmov, ktoré nie sú exaktné, najväčšou nevýhodou môže byť nenájdenie vhodného riešenia. Síce agent prejde všetky učiace epizódy, nemusí nájsť

také riešenie, ktoré by bolo v úlohe zrovnateľné s človekom alebo by riešilo prijateľne danú úlohu. Ako príklad môže slúžiť prostredie Mountain Car, v ktorom sa môže stať, že aj po ukončení učiaceho cyklu sa niekedy agentovi nepodarí vyjsť na kopec v niektorých prípadoch. V iných sa mu to podarí. Na rozdiel od takéhoto riešenia by exaktný algoritmus vyriešil prostredie tak, že v každom pokuse agent vyjde na kopec.

Ďalšou nevýhodou môže byť to, že pre riešenie problému potrebujeme jeho počítačový model. Napríklad pri simulovaní križovatky by sme potrebovali model križovatky, na ktorom by pracoval agent RL. Štandardným NN by mohli postačovať dáta o križovatke, z ktorých by sa natrénovala.

9 Záver

Vývoj v oblasti informatiky v posledných rokoch je stále nezadržateľný. Taktiež aj vývoj AI. Vylepšujú sa možnosti učenia, hľadajú sa nové spôsoby, ktoré by reflektovali potreby doby. Pod pojmom potreby sa skrýva rýchlosť trénovania, rýchla aplikovateľnosť modelov a výsledky simulujúcich ľudské správanie a rozhodovanie.

Aj keď RL vychádza zo správania zvieraťa, použitie to neobmedzuje len na zvieraciu ríšu. Učenie, tak ako u zvierat, prebieha na základe odmeňovania. Vďaka tomu je RL model vhodný na aplikovanie do oblastí, kde je potreba interagovať z riešenou úlohou, napríklad v simuláciách a hrách. Tým, že model interaguje s problémom, je jeho učiaci cyklus viac pochopiteľný pre človeka, kde sledovaním vývoja odmeny vieme určiť, či sa model učí dobre ale zle.

Ako bolo spomenuté, v problémoch, kde je potrebná interakcia človeka môže byť nahradená modelom RL. V príklade simulácie, napríklad pohybu áut na križovatke, tak ako aj človek, sa musí agent učiť spôsobom skúšania rôznych spôsobov riadenia dopravy na križovatke. Po učiacom cykle dokáže agent RL na základe jeho naučenej stratégie ovládať simuláciu podobnou kvalitou (ak nie lepšou) ako človek.

V dnešnej dobe je však populárna aj oblasť hrania hier. Agenti RL v tejto oblasti môžu hrať viacero rolí. Napríklad formu AI, proti ktorej hráči hrajú. Vďaka tomu, že sa dokáže model RL učiť z interakcií, dokáže byť v celku kvalitným a náročným protivníkom. RL sa naučí svoju stratégiu na základe toho, ako hrá proti nemu ľudský hráč, čiže sa môže zlepšovať, čím bude rásť aj obtiažnosť pre hráča.

Na druhej strane, pri pohľade na hru ako určitú formu problému, dokážu RL agenti napríklad vyriešiť problém dosiahnutím cieľa, prejdením levelu alebo ovládaním určitého objektu. Počas procesu učenia sú schopné sa naučiť vlastnú stratégiu, ktorá môže byť lepšia ako stratégia ľudského hráča. Dokonca aj takú, ktorá by človeka nenapadla. Vďaka tomu sú agenti RL veľmi populárnymi.

Ďalšími možnými využitiami sú napríklad oblasti obchodovania na burze. Veľa robotov obchodníkov v tejto oblasti je vytvorených na princípoch RL. V tomto prostredí je odmena prirodzená – vo forme peňažného zisku alebo straty. Podľa nej sa dokážu modely učiť kontinuálne stratégie generovania obchodných signálov.

V tejto práci boli implementovaný do priloženej aplikácie rôzny agenti RL. Prostredia, ktoré riešili, boli viacerých typov. Niektoré jednoduchšie, iné zložitejšie alebo spojité. Tieto prostredia obsahovali reálne problémy implementované pomocou hry, napríklad pristátie s vesmírnym modulom alebo vyjdenie s autom na kopec. Výsledky agentov sú porovnateľné, dokonca lepšie, ako výsledky ľudského hráča.

Modely RL, ako aj ostatné modely, majú svoje silné aj slabšie stránky. Silné stránky sú väčšinou známe podľa modelu. Nevýhody sa však ukážu až v prostredí. Pomocou experimentov sme sa snažili pridať moduly, ktoré by dokázali vyvážiť slabé stránky modelov globálne vo väčšine prostredí.

V prvom experimente bolo cieľom nájdenie vhodných vylepšení modelov, ktoré by pomohli k stabilizovaniu vývoju gradientu a k zrýchleniu dosiahnutia riešenia agentov. Stabilizovanie gradientu síce vedie k stabilizovaniu gradientu, avšak experiment ukázal, že stabilizovanie je na úkor času nájdenia riešenia. Použitie týchto modulov spoločne sa podľa experimentu ukazuje ako najvhodnejšie v spojitých prostrediach väčších rozsahov. Pri menších alebo diskrétnych prostrediach je vhodnejšie používanie modulu cieľovej – zdrojovej NN samostatne. Rýchlosť dosiahnutia riešenia agentom môže byť zvýšená pridaním alebo modifikovaním agentových hyper – parametrov, spôsobu tvorenia prieskumov alebo zdieľaním parametrov sietí. Tieto moduly môžu byť použité aj spoločne. Ich prítomnosť v agentoch podľa výsledkov experimentu má nemalý vplyv na rýchlosť získanie riešenia agentov.

Ako bolo spomenuté vyššie, učenie RL agentov prebieha na základe veľkosti získanej odmeny. Preto v druhom experimente boli testované vplyvy nastavenia štruktúry odmien na učiaci cyklus agentov. Spôsob odmeňovania môže byť ľubovoľný. Treba však dbať na to, čo od agenta chceme, aby v konečnom dôsledku vykonával. Ak sa jedná o problém kontroly nad objektom, treba zvážiť oceňovanie agenta kontinuálne kladnými hodnotami (najlepšie nízkymi) za akciu, ktorá nespôsobí zrážku. Tento spôsob odmeňovania spôsobí, že agent sa bude snažiť čo najviac robiť také pohyby, ktoré ma overené a nespôsobia kolíziu. V prostrediach, kde je však nutné nájdenie konečného riešenia (napr. vyjdenie na kopec), by tento spôsob narobil problémy. V takýchto úlohách je najlepšie používať zápornú nízku odmenu za každý krok, ktorý nevedie k riešeniu. Ukončenie epizódy môže byť hodnotené kladne alebo záporne väčšou hodnotou podľa toho, či agent dosiahol cieľ alebo nie.

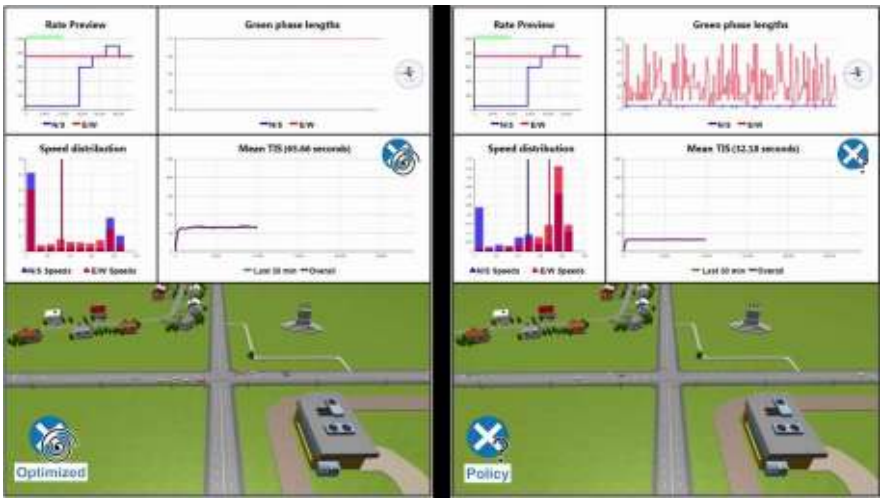
V poslednom experimente boli využité všetky výsledky predošlých experimentov dohromady do modelu DDPG. Tento model sa z hľadiska použitého konceptu javí ako veľmi vhodný na širokú škálu problémov. Základný koncept modelu však počíta len s použitím na spojité prostredie. Pre diskrétne prostredie je zastúpený modelom DQN, z ktorého je DDPG pomocou princípu Aktér – kritik odvodený na spojité prostredia. Aplikovanie rozšírení (kapitola výsledky práce) pomohlo vylepšiť model, ktorý je vďaka rozšírenia vhodne aplikovateľný na širokú škálu problémov. Jeho výsledky v testovaných prostredia rapídne prevyšujú ostatné algoritmy.

Modely v tejto práci, ale aj RL ako celok, predstavujú iný pohľad na umelú inteligenciu. Ich učiaci cyklus je pre človeka prirodzený a jeho výsledky sú ľahko interpretovateľné, napríklad vo forme hry.

A čo ďalej? Do budúcnosti RL algoritmy majú veľký potenciál. Vďaka ich aplikácii na simulácie, interaktívne prostredia a problémy sú vhodné na riešenie širokej škály problémov. Nepotrebujú dopredu poznať veľkú základňu údajov, ktoré musia byť označené aby sa ich naučili odhadovať. Pokiaľ je k dispozícii vhodný model, vedia sa naučiť pôsobiť v danej problematike a vyriešiť ju.

Vývoju AI dosť napomáha rast sily HW, vďaka čomu sú stále menej a menej limitované HW požiadavkami. To umožňuje rozširovanie modelov do väčších rozmerov, kde dokážu pracovať efektívnejšie aj na komplexnejších problémoch.

O tom svedčí aj smerovanie spoločnosti AnyLogic, ktorá rozširuje svoje simulačné modely o umelú inteligenciu, a aj o RL [25]. Príkladom môže byť simulácia semaforov, kde výsledky porovnávajú s AI. Uvedomujú si silu AI v tejto oblasti, preto aj do ďalšieho obdobia plánujú rozširovanie svojej simulačnej platformy o AI, konkrétne aj o RL.



Obrázok 25 AI vs štandardné algoritmy[25]

V oblasti hier, spoločnosť DeepMind sa zameriava (po úspešných pokusoch s inými hrami) na komplexnejšie hry. Príkladom je výskum v strategickej hre StarCraft [26], ktorá sa vyznačuje jej komplexnosťou. V čase písania tejto práce už prezentujú svoje čiastkové výsledky v tejto hre, ktoré sú zrovnateľné s najlepšími hráčmi. Do budúcna plánujú naďalej rozvíjať potenciál AI v hrách, čím prispievajú k vedeckej oblasti a skvalitneniu umelej inteligencie, napríklad vo forme protihráča v hrách.

Zoznam použitej literatúry

Knihy

[100] D. Marček, M. Marček „Neurónové siete a ich aplikácie", EDIS, Žilinská univerzita v Žiline, 2006

[101] R. Sutton, Andrew Barto „Reinforcement Learning", The MIT Press, 2018

[102] Uhlenbeck, George E and Ornstein, Leonard S. On the theory of the brownian motion. Physical review, 1930.

[103] Hornik, K., Stinchcombe, M., and White, H.: Multilayer feedforward networks are universal approximators. Neural Networks 2 (1989).

Internetové zdroje

[1] http://www.contrib.andrew.cmu.edu/~mndarwis/ML.html

[2] https://sge.wonderville.ca/machinelearning/history/history.html

[3] http://incompleteideas.net/book/ebook/

[4] https://chatbotslife.com/a-brief-history-of-chatbots-d5a8689cf52f

[5]https://www.techrepublic.com/article/ibm-watson-the-inside-story-of-how-the-jeopardy-winning-supercomputer-was-born-and-what-it-wants-to-do-next/

[6]https://www.sri.com/case-studies/the-man-the-myth-the-legend-meet-shakey-the-robot-the-worlds-first-ai-based-robot/

[7] https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf

[8] https://www.retrogames.cz/konzole_atari2600.php

[9]http://pdf.truni.sk/e-ucebnice/iktv/data/3c6adade-1ed9-4e83-becb-9b7f61fa5036.html?ownapi=1

[10] http://kti.mff.cuni.cz/~marta/mdp.pdf

[11] https://arxiv.org/pdf/1507.06527.pdf

[12] https://arxiv.org/pdf/1911.10635.pdf

[13]https://medium.com/mlreview/making-sense-of-the-bias-variance-trade-off-in-deep-reinforcement-learning-79cf1e83d565

[14] https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

[15] https://arxiv.org/pdf/1712.01815.pdf

[16] https://spinningup.openai.com/en/latest/spinningup/rl_intro.html#bellman-equations

[17]https://www.scientificamerican.com/article/20-years-after-deep-blue-how-ai-has-advanced-since-conquering-chess/

[18] https://www.bbc.com/news/technology-35785875

[19] https://www.computerweekly.com/photostory/450423802/AI-A-brief-history-of-man-versus-machine-intelliegnce/3/IBM-Watson-versus-Jeopardy

[20] https://www.aaai.org/Magazine/Watson/watson.php

[21] https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf

[22] https://gym.openai.com/envs/

[23] http://files.davidqiu.com//research/nature14236.pdf

[24] https://arxiv.org/pdf/1509.02971.pdf

[25]https://www.anylogic.com/blog/machine-learning-and-simulation-example-and-downloads/

[26] https://deepmind.com/blog/article/AlphaStar-Grandmaster-level-in-StarCraft-II-using-multi-agent-reinforcement-learning

[27] https://github.com/openai/gym/wiki/Leaderboard

[28] https://github.com/openai/gym/wiki/Leaderboard

Prílohy

Príloha A DVD

Prílohy

Príloha A: DVD

Priložené DVD obsahuje:

- Aplikácia učenia s posilňovaním na vybraný problém (formát PDF)
- Programová implementácia aplikácie .exe
- Zdrojové kódy k aplikácii spustiteľné v PyCharm 2019.3.2 v jazyku Python 3.6.x
- Potrebné knižnice (pre spustenie projektu):
    - Keras 2.3.1
    - Tensorflow 2.1.0
    - Tensorflow – gpu 2.1.0
    - Box2D 2.3.2
    - PyQt5 5.14.1
    - Gym 0.15.4
    - Numpy 1.16.3