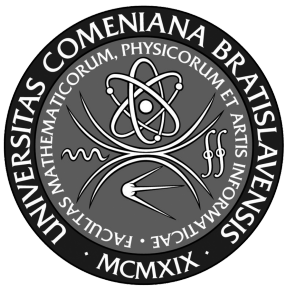
UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

EXPLORAČNÉ STRATÉGIE V UČENÍ POSILŇOVANÍM

Diplomová práca

2019 Bc. Jerguš Adamec

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



EXPLORAČNÉ STRATÉGIE V UČENÍ POSILŇOVANÍM

Diplomová práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	prof. Ing. Igor Farkaš, Dr.
Konzultant:	Mgr. Matej Pecháč

Bratislava, 2019 Bc. Jerguš Adamec

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Jerguš Adamec
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: aplikovaná informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Exploračné stratégie v učení posilňovaním
Exploration strategies in reinforcement learning

Anotácia: Učenie modelov (napr. neurónových sietí) pomocou posilňovania (reinforcement learning, RL) je štandardným prístupom v strojovom učení. Súčasťou aktívneho RL, teda hľadania optimálnej stratégie na základe odmeny z prostredia, je explorácia stavového priestoru. Je potrebné skúmať rôzne exploračné stratégie, ktoré umožňujú toto prehľadávanie zefektívniť, keďže RL je pomerne pomalé a vyžaduje si vysoký počet iterácií.

Cieľ: 1. Preštudujte základné exploračné stratégie pri učení posilňovaním, vrátane stratégie založenej na počte návštev jednotlivých stavov (count-based).
2. Porovnajte bežne používané stratégie s count-based stratégiou v jednoduchšom a zložitejšom diskrétnom priestore (napr. bludisko) a preskúmajte vplyv hyperparametrov na rýchlosť konvergencie skúmaných stratégií.

Literatúra: Sutton R.S., Barto A.G. (2018). Reinforcement learning: An introduction (2. vyd.). Cambridge: MIT press.
Kober J., Bagnell A., Peters J. (2013). Reinforcement Learning in Robotics: A Survey. International Journal of Robotics Research, 32: 1238-1274.
Bellemare M., Srinivasan S., Ostrovski G., Schaul T., Saxton D., Munos R. (2016). Unifying count-based exploration and intrinsic motivation. In Advances in Neural Information Processing Systems,pp. 1471-1479.

Vedúci: prof. Ing. Igor Farkaš, Dr.
Konzultant: Mgr. Matej Pecháč
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 07.10.2017

Dátum schválenia: 09.11.2017 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestne prehlasujem, že túto diplomovú prácu som vypracoval samostatne len s použitím uvedenej literatúry s pomocou môjho školiteľa a konzultanta.

. . . . . .

Bratislava, 2019 Bc. Jerguš Adamec

Poďakovanie

Moja vďaka patrí vedúcemu mojej diplomovej práce prof. Ing. Igorovi Farkašovi, Dr. a môjmu konzultantovi Mgr. Matejovi Pecháčovi, za ich odborné rady, ľudský prístup, čas a ochotu pri vypracovaní tejto práce. V neposlednom rade ďakujem mojej rodine a najbližším za ich neustálu podporu.

Abstrakt

Exploračné stratégie v aktívnom učení posilňovaním umožňujú agentovi skúmať prostredie a napomáhajú efektívnemu priebehu procesu učenia. Kompromis medzi exploráciou a exploitáciou, teda využívaním aktuálnej stratégie, je stále otvoreným problémom a predmetom výskumu. V rámci práce sa oboznámime s niektorými typmi explorácií. Preskúmame stochastické explorácie a jednoduchý typ explorácií založených na vnútornej motivácii a implementujeme ich v rámci aproximovanej verzie algoritmu Q-learning. Hlavnou časťou práce je systematická realizácia experimentov v prostredí bludiska s dvoma úrovňami zložitosti, kde cieľom je nájsť najkratšiu cestu. Pomocou simulácii získame výsledky pre jednotlivé explorácie a porovnáme ich pomocou viacerých kvantitatívnych mier.

Kľúčové slová: učenie posilňovaním, exploračné stratégie, neurónové siete

Abstract

Exploration strategies in active reinforcement learning allow the agent to explore the environment and they support its effective learning process. An exploration-exploitation tradeoff is still the subject of research. In the thesis we will become familiar with several of exploration methods. We explain stochastic explorations and the simplified kind of exploration based on intrinsic motivation. Our next step is its implementation using the approximate version of the Q-learning algorithm. The main part of the thesis is deals with the systematic design of the experiments in the environment of a maze with two level of complexity, where the aim is to find the shortest path. Using the simulations we get results for each exploration and compare them with different metrics.

Keywords: reinforcement learning, exploration strategies, neural nets

Obsah

Zoznam použitých termínov

Markov Decission Process	MDP	Markov rozhodovací proces
Policy	π	Stratégia
Process Policy Prediction	PPP	Vyhodnotenie stratégie
Process Policy Improvement	PPI	Vylepšenie stratégie
Policy evaluation improvement	PEI	Metóda vylepšenej predikcie stratégie
Greedy policy	π'	Chamtivá stratégia
Multilayer perceptron	MLP	Viacvrstvový perceptrón
Intrinsic motivation	IM	Vnútorná motivácia
Policy Iteration	PI	Iterovanie stratégie
Value Iteration	VI	Iterovanie hodnôt
Gradient descent	GD	Gradientový zostup

Úvod

Učenie posilňovaním predstavuje skupinu algoritmov, ktoré sú schopné učiť sa na základe vlastnej skúseností bez primočiareho usmerňovania. Je to druh odmenovo orientovaného učenia, v ktorom agent, situovaný v nejakom prostredí, sa na základe interakcie snaží maximalizovať odmenu získanú z prostredia.

Existuje viacero metód a algoritmov v učení posilňovaním, avšak všetky majú jednu spoločnú vec a tou je spôsob explorácie prostredia. Znamená to kvalitu, s akou agent interaguje s prostredím. Explorácia radikálnym spôsobom ovplyvňuje celkový proces učenia agenta. Pretože aj pri efektívnych algoritmoch a jednoduchých úlohách môže nevhodná explorácia zapríčiniť veľmi dlhý proces učenia bez lepších výsledkov. A naopak, efektívne metódy explorácie dokážu riešiť komplexné úlohy aj v extrémne náročných prostrediach. Je to aj vďaka nedávnym pokrokom vo výskume neurónových sietí, ktoré ovplyvnili aj metódy učenia posilňovaním, čím zapríčinili aj ich aplikáciu v rôznych oblastiach.

Cieľom práce je sa oboznámiť so základnými exploračnými stratégiami v učení posilňovaním. Na základe získaných teoretických poznatkov v ďalšej časti práce zostavíme experimenty, ktorých cieľom bude simulačne porovnať vlastnosti jednotlivých explorácií na úlohe hľadania najkratšej cesty.

V kapitole 1 sa venujeme teoretickým poznatkom potrebným pre splnenie cieľa práce. Formalizujeme si problém učenia posilňovaním a opíšeme si jeho základné metódy a k nim prislúchajúce algoritmy. Pozornosť venujeme algoritmu Q-learning a jeho aproximovanej verzii pomocou neurónovej siete. Na záver kapitoly si ukážeme, akú úlohu zohrávajú exploračné stratégie v procese učenia agenta. Priblížime si základné typy stochastických explorácií a taktiež sa v krátkosti oboznámime s jednoduchou verziou stratégií založených na vnútornej motivácii.

V kapitole 2 sa zameriame na podobné práce z danej problematiky. V krátkosti si opíšeme hlavné myšlienky týchto prác. Bližšie si uvedieme opis prác, z ktorých sme čerpali inšpiráciu. Taktiež si uvedieme niektoré vedecké práce, ktoré predstavujú aktuálny stav problematiky.

Kapitola 3 predstavuje praktickú časť práce. Venujeme v nej pozornosť zostaveniu experimentom, pomocou ktorých skúmame exploračné stratégie. Opíšeme si, akým spôsobom sme zostavili experimenty a v nasledujúcej kapitole 4 si prezentujeme výsledky explorácií vo forme tabuliek a vizualizácií kriviek učenia.

Kapitola 1

Učenie posilňovaním

Táto kapitola je venovaná teroretickej časti práce, v ktorej charakterizujeme koncept učenia posilňovaním (Reinforcement learning - RL). Popíšeme si, z akých teoretických predpokladov RL vychádza a aké typy problémov rieši. Popri tradičným metódam a ich algortimom si ukážeme ich modernejšie alternatívy, založené na optimalizáciách gradientového typu. Špeciálne si objasníme takýto prípad optimalizácie pomocou neurónovej siete a na záver si ukážeme akú úlohu zohrávajú exploračné stratégie v RL.

1.1 Paradigma RL

Problém ktorý RL rieši je učenie agenta interakciou s prostredím. Rôzne stavy prostredia a možné akcie v ňom sú ohodnotené numerickou hodnotou – odmenou, ktorá závisí od problému a je súčasťou prostredia. Úlohou agenta je maximalizovať takúto odmenu. Inak povedané, je to rozhodovací proces, ktorého cieľom je pre každý stav daného prostredia vybrať takú akciu, aby bola maximalizovaná odmena z dlhodobého hľadiska (Sutton & Barto, 2017).

1.1.1 Markovov rozhodovací proces

Problematika RL je formalizovaná Markovovým rozhodovacím procesom (Markov Decission Process - MDP). Skladá sa z nasledujúcej 5-tice: $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, pričom

- $\mathcal{S}$: množina stavov,
- $\mathcal{A}$: množina akcií,

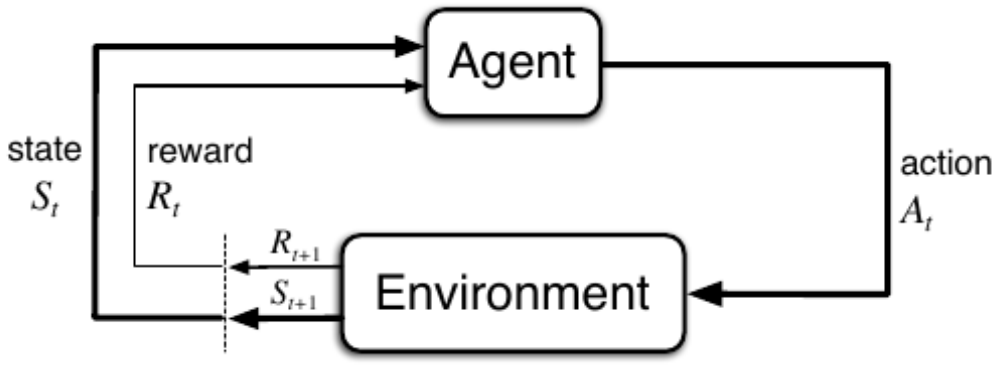
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ prechodová funkcia,
- $\mathcal{R} : \mathcal{S} \rightarrow \mathbb{R}$: je odmeňovacia funkcia $R(s)$ odmena pre stav s,
- γ: diskontný faktor v rozsahu $(0, 1]$.

Prechodová funkcia $\mathcal{T}$ nám vyjadruje podmienenú pravdepodobnosť $p(s'|s, a)$, že agent sa ocitne v stave s', ak v stave s vykonal akciu a. Výber akcie v danom stave určuje tzv. stratégia (policy) π:$S \rightarrow A$, ktorá definuje správanie agenta. Z toho vyplýva, že maximalizovanie danej odmeny získame iba za predpokladu, že stratégia je optimálna, označujeme π^*. Preto cieľom RL je nájdenie takejto optimálnej stratégie. Existuje mnoho metód, ktoré ju buď vypočítajú priamo alebo s pomocou hodnotových funkcií, ktoré si priblížime nižšie.

Spomínaná kumulovaná odmena z dlhodobého hľadiska je vyjadrená nasledovne:

$$R_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k, \tag{1.1}$$

kde diskontný faktor γ určuje váhovanie odmien. Čím je γ väčšia, tým viac je zohľadnená odmena z dlhodobého hľadiska. Vzťah 1.1 vyjadruje očakávanú odmenu pre epizodický typ úlohy agenta a aj pre typ úlohy ktorá sa deliť na epizódy nedá. Zjednotenie týchto dvoch typov úloh umožňuje zavedenie tzv. *absorpčného stavu*. Ak prechodová funkcia $\mathcal{T}$ premietne agenta do *absorpčného* stavu, tak takýto prechod sa bude opakovať so získanou odmenou 0. Absorpčný stav znamená ukončenie epizódy (Sutton & Barto, 2017).



Obr. 1.1: Schéma interakcie agenta s prostredím. Prebraté zo (Sutton & Barto, 2017).

Cyklus MDP nám znázorňuje obrázok 1.1, na ktorom vidíme ako agent intera-

guje s prostredím v priebehu času. V každom kroku t agent získa reprezentáciu stavu prostredia s_t a na základe toho si vyberie akciu a_t. V ďalšom kroku agent získa odmenu r_{t+1} ako dôsledok výberu predchadzajúcej akcie a dostane sa do nového stavu s_{t+1}.

Hodnotové funkcie

Hodnotová funkcia V^π pre stratégiu π je očakávaná hodnota odmeny s faktorom zľavy pre daný stav s a následným sledovaním tejto stratégie od tohto stavu. Táto funkcia môže byť definovaná pre stav, alebo pre pár stav–akcia. Hodnotové funkcie odhadujú, ako výhodné je pre agenta byť v danom stave, alebo ako výhodné je vykonanie danej akcie v stave. Ukážeme si jej vyjadrenie pomocou Bellmanových rovníc, bez ktorých by sme MDP nedokázali vyriešiť. Bellmanová rovnica pre hodnotovú funkciu $V^\pi(s)$: $\mathcal{S} \to \mathbb{R}$ je definovaná:

$$\begin{aligned} V^\pi(s) &= E_\pi[R_t \mid s_t = s] \\ &= E_\pi[r_{t+1} + \gamma R_{t+1} \mid s_t = s] \\ &= \sum_a \pi(s,a) \sum_{s'} p(s'|s,a)[R(s) + \gamma V^\pi(s')], \text{ pre } \forall\ s \in S \end{aligned} \tag{1.2}$$

Analogicky Bellmanova rovnica pre hodnotovú funkcia **stav–akcia** má tvar:

$$\begin{aligned} Q^\pi(s,a) &= E_\pi[R_t \mid s_t = s, a_t = a] \\ &= E_\pi[R_{t+1} + \gamma R_{t+1} \mid s_t = s, a_t = a] \\ &= \sum_{s'} p(s'|s,a)[R(s) + \gamma \sum_{a'} \pi(a',s') Q^\pi(s',a')] \end{aligned} \tag{1.3}$$

Rovnice 1.2 a 1.3 vyjadrujú rekurzívny vzťah medzi hodnotou súčasného stavu a hodnotami nasledujúcich stavov. Keď poznáme ohodnotenie rôznych stratégií pomocou Bellmanových rovníc, môžeme si objasniť princíp ako dostaneme optimálnu stratégiu.

Docielime to pomocou Bellmanových rovníc pre optimálne hodnotové rovnice:

$$\begin{aligned} V^*(s,a) &= \max_a v_\pi(s) \\ &= \max_a \sum_{s'} p(s'|s,a)[R(s) + \gamma V^*(s')] \\ Q^*(s,a) &= \max_a q_\pi(s,a) \\ &= \sum_{s'} p(s'|s,a)[R(s) + \gamma \max_{a'} Q^*(s',a')] \end{aligned} \tag{1.4}$$

Vychádzali sme zo (Sutton & Barto, 2017).

1.2 Metódy riešenia RL

Táto podkapitola je venovaná metódam RL, pomocou ktorých dokážeme vypočítať optimálnu π^*. Metódy delíme do troch hlavných kategórií: **Dynamické programovanie (DP)**, **Monte-Carlo metódy (MC)** a **TD učenie (TD)**. Najprv upriamime pozornosť na klasickú metódu DP pre vyriešenie MDP. Opíšeme si v nej základné algoritmy, z ktorých vychádzajú metódy MC aj TD. Zadefinujme si výpočtovo efektívny spôsob odhadu hodnotovej funkcie *inkrementálnou metódou*, ktorej predpis vo všeobecnosti je:

$$NewEstimate = OldEstimate + StepSize[Target - OldEstimate] \tag{1.5}$$

Výraz $[Target - OldEstimate]$ vyjadruje znižovanie *chyby* čím sa odhady blížia $Target$u. *StepSize* je konštanta v rozsahu $(0, 1]$, pretože uvažujeme nestále prostredie (odhady sa menia v čase). Tento predpis je kľúčový pre niektoré typy algoritmov metód MC aj TD (Sutton & Barto, 2017).

1.2.1 Dynamické programovanie

Dynamické programovanie vo všeobecnosti znamená spôsob riešenia problémov, ktoré sa dajú rozdeliť na podproblémy. Riešenia podproblémov vieme znovu použiť a tento proces opakujeme, až kým nedostaneme globálne optimálne riešenie, vyskladané z týchto mnoha menších subriešení.

V kontexte RL vieme pomocou algoritmov patriacich do tejto skupiny vypočítať optimálnu stratégiu iba za predpokladu, že poznáme aj model MDP. Pod pojmom model myslíme celú dynamiku prostredia, čiže prechodovú funkciu $\mathcal{T}$ a odmenovú funkciu $\mathcal{R}$ z pätice definujujúcej MDP. Kvôli tomuto predpokladu a taktiež výpočtovo náročným metódam (ukážeme nižšie), koncept Dynamického programovania v RL je dôležitý skôr z teoretického hľadiska, ktorý poskytuje základ pre pochopenie odvíjajúcich sa ďalších metód a ich algoritmov na riešenie MDP. Hlavnou myšlienkou metód DP je využitie rekurzívnej formy Bellmanových rovníc hodnotových funkcií pre *učiace pravidlo*, ktoré využijeme na hľadanie optimálnej stratégie.

Iterovanie stratégie

Algoritmus Iterovania stratégie (Policy Iteration - PI) je založený na myšlienke striedania procesov vyhodnotenia stratégie (Process Policy Prediction - PPP) a procesu vylepšenia stratégie (Process Policy Improvement - PPI). Takéto striedanie procesov môžeme vyjadriť ako sekvenciu podľa (Sutton & Barto, 2017), ilustrovanú nasledovne:

$$\pi_0 \rightarrow^E q_{\pi_0} \rightarrow^I \pi_1 \rightarrow^E q_{\pi_1} \rightarrow^I \pi_2 \rightarrow^E ... \rightarrow^I \pi_* \rightarrow^E q_{\pi_*} \tag{1.6}$$

Iteratívne algoritmus najskôr vypočíta hodnotovú funkciu V^π pre danú π v kroku PPP (označenie E v rovnici 1.6), aby mohol následne pomocou PPI (označenie I v rovnici 1.6) nájsť lepšiu π' vypočítanú V^π. Ďalší krok je opäť vypočítanie hodnotovej funkcie $V^{\pi'}$ pre novú π'. To sa opakuje, až kým nenájdeme stratégiu, ktorá sa nemení. Stratégia konverguje, pretože uvažujeme konečný MDP s konečným počtom π, a preto π a V konvergujú v konečnom počte iterácií.

Proces vyhodnotenia stratégie. Cieľ spočíva vo vypočítaní hodnotovej funkcie pre danú stratégiu. To môžeme uskutočniť buď analytickým spôsobom, kde máme n rovníc o n neznámych, pričom $n = |S|$. Alebo iteratívnym spôsobom, kde pravidlo učenia je nasledovné (Sutton & Barto, 2017):

$$V_{k+1}(s) = \sum_a \pi(s,a) \sum_{s'} p(s' \mid s,a)[R(s) + \gamma V_k(s')], \tag{1.7}$$

pre $k \longmapsto \infty$ V_k konverguje ku V^π.

Proces vylepšenia stratégie vyberá akciu $a \neq \pi(s)$ a potom nasleduje výber akcií podľa stratégie π. Hodnotová funkcia V^π pre stratégiu π nám určí, nakoľko je to výhodné. Avšak teraz chceme určiť, nakoľko je výhodné vybratie danej akcie a v stave s. Na to nám poslúži hodnotová funkcia Q^π spôsobom: Ak je hodnota $\max_a Q(s,a)$ väčšia ako $V(s)$, tak je výhodnejšie vybrať akciu a, než akciu danú $\pi(s)$. Ak tento proces generalizujeme, čiže aplikujeme pre všetky stavy a akcie, tak dostaneme novú tzv. pažravú (greedy) stratégiu π', ktorá je daná nasledovne:

$$\begin{aligned} \pi'(s) &= \arg\max_a Q^\pi(s,a) \\ &= arg\max_a \sum_{s'} p(s' \mid s,a)[R(s) + \gamma V(s')] \end{aligned} \tag{1.8}$$

Ak docielime to, že π' je taká dobrá ako pôvodná π, tak $\pi'(s)$ je optimálna stratégia π^*.

Iterovanie hodnôt

Algoritmus Iterovania hodnôt (Value Iteration - VI) má oproti PI výhodu v tom, že každá z jej iterácií nevyžaduje nové vyhodnotenie stratégie (predikciu), čo je veľmi zdĺhavý proces. VI sa snaží o efektívnejší prístup a to tak, že kombinuje proces vylepšenia stratégie a skrátenú verziu procesu vyhodnotenia stratégie (jedna aktualizácia v každom stave). Iteračná metóda hodnotovej funkcie je definovaná nasledovným pravidlom:

$$V_{k+1}(s) = \max_a \sum_{s'} p(s' \mid s,a)[R(s) + \gamma V_k(s')] \tag{1.9}$$

Môžeme si všimnúť, že výraz je podobný tomu pri PPP až na max operátor. Keď máme vypočítanú hodnotovú funkciu, vieme stratégiu vyjadriť nasledovne:

$$V_{k+1}(s) = \arg\max_a \sum_{s'} p(s' \mid s,a)[R(s) + \gamma V_k(s')] \tag{1.10}$$

1.2.2 Monte-Carlo metódy

V krátkosti si ukážeme princíp MC metód, ktoré oproti DP hodnotové funkcie odhadujú. Najvačší rozdiel oproti DP je v tom, že MC metódy nepoznajú model, čiže prechodová funkcia $\mathcal{T}$ je neznáma a taktiež MC nerobí *bootstrapping*. Bootstrapping

je technika v RL, kedy použijeme odhadované hodnoty v učiacom pravidle pre úpravu odhadov rovnakého druhu. Preto Monte Carlo rieši MDP pomocou spriemerňovania vzoriek. Tieto vzorky - získane skúsenosťou, teda interakciou s prostredím - sa využívajú na aktualizáciu V, Q a π iba na konci epizód. Preto predpokladáme, že MC úlohy by mali byť nadizajnované v tejto forme.

MC riadenie

Riešením problému riadenia (control) je nájdenie optimálnej stratégie, ktorú vypočítame pomocou málo pozmenenej verzie algoritmu 1.2.1 z predchádzajúcej podkapitoly. Proces vyhodnotenia stratégie je realizovaný pomocou hodnotovej funkcie V, kde namiesto použitia modelu pre vypočítanie hodnoty daného stavu, spriemerujeme tzv. *returns* definovaný vzťahom 1.1. Čím väčší počet vzoriek získame, tým viac sa priblížime k $V(s)$ stavu s. Proces PPP zrealizujeme pomocou hodnotovej funckie Q, takže žiadny model prostredia nie je potrebný pre nájdenie chamtivej (greedy) stratégie π'. Pre akúkoľvek Q funkciu získame π' nasledovným výrazom:

$$\pi' = \arg\max_a Q(s, a) \tag{1.11}$$

Pri procese predikcie a vylepšenia stratégie pri MC rozlišujeme dva spôsoby: on-policy a off-policy. Pri on-policy používamé cieľovú stratégiu, na ktorú aplikujeme krok predikcie a vylepšenia a zároveň ju používame aj na generovanie akcií zabezpečujúcich správanie agenta. Pri off-policy máme okrem cieľovej stratégie aj tzv. behaviorálnu stratégiu, ktorú využívame práve pri definovaní správania agenta. Čerpali sme zo zdroja (Sutton & Barto, 2017).

1.2.3 TD učenie

Metódy TD učenia sú kombináciou MC a DP. Model nie je známy ako pri MC a aktualizácia parametrov prebieha po každom kroku. Nečaká sa na koniec epizódy, takže robia *bootstraping* ako pri metódach DP (Sutton & Barto, 2017).

TD Predikcia

Tak ako problém predikcie pri MC metódach, tak aj pri TD sa predikcia odhaduje vzorkami nadobudnutými skúsenosťou. Obidve metódy aktualizujú svoj odhad $V(s_t)$ pomocou $V(s_{t+1})$ pre neterminálne stavy. Veľká výhoda TD oproti MC spočíva v tom, že na aktualizáciu odhadu hodnotovej funkcie sa čaká iba jeden časový krok. Táto vlastnosť je výhodná, pretože niektoré aplikácie majú veľmi dlhé trvanie epizód a niektoré nemusia mať vôbec epizódy. Pravidlo učenia pre tabuľkovú verziu odvodené z pravidla 1.5 je (van Hasselt, 2012):

$$\begin{aligned} \delta_t &= r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t) \\ V_{t+1}(s_t) &= V_t(s_t) + \alpha_t(s_t)\delta_t \end{aligned} \tag{1.12}$$

kde δ_t je *TD chyba* a rýchlosť učenia je $\alpha \in [0, 1]$.

1.3 Algoritmy TD

Algoritmy TD učenia patria momentálne k najviac používaným algoritmom RL. Výhodou je, že sú ľahko implementovateľné, s minimálnym množstvom výpočtov potrebných v on-line spôsobe aktualizácie parametrov. Bližšie si uvedieme základne dva algortimy TD učenia pre riadenie kontroly on-policy (SARSA) a pre off-policy (Q-learning).

1.3.1 SARSA

SARSA (Rummery, 1994) je on-policy typ algoritmu TD učenia. *Pravidlo učenia* využíva päticu prvkov $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ po každom prechode do neterminálneho stavu s_t, z ktorej pochádza aj názov algoritmu. Ak stav s_{t+1} je terminálny, tak $Q(s_{t+1}, a_{t+1}) = 0$. SARSA sa učí Q hodnoty, ktoré sú asociované so stratégiou, ktorú práve nasleduje. Učiace pravidlo má tvar:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)] \tag{1.13}$$

SARSA konverguje k optimálnej Q funkcii a tým zároveň aj k optimálnej stratégie, keď všetky páry stav–akcia Q hodnotovej funkcie sú navštívené nekonečný počet krát.

Pseudokód SARSA je uvedený v algoritme 1.1 (Sutton & Barto, 2017).

Algorithm 1.1 SARSA (on-policy TD metóda) pre odhad $Q \approx q_*$

Initialize $Q(s, a)$, for all $s \in S$, arbitrarily, and $Q(terminal - state, .) = 0$
Repeat(for each episode):
    Initialize s
    Choose a from s using derived from Q
    repeat(for each step of episode):
        Take action a, observe r, s'
        Choose a' from s' using derived from Q
        $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
        $s \leftarrow s'; a \leftarrow a';$
    until s is terminal

1.3.2 Q-learning

Q-learning (Watkins, 1992) je iteratívny, off-policy algoritmus, ktorý sa snaží nájsť optimálnu stratégiu pomocou priameho aproximovania Q hodnotovej funkcie, nezávisle od toho, ktorú stratégiu nasleduje. Pravidlo učenia je nasledovné:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_a Q(s', a) - Q(s, a)] \tag{1.14}$$

Na konvergenciu Q-learningu je potrebné všetky Q páry aktualizovať nekonečne veľa krát. Pseudokód Q-learningu je uvedený v algoritme 1.2 (Sutton & Barto, 2017).

Algorithm 1.2 Q-learning (off-policy TD metóda) pre odhad $\pi \approx' pi^*$

Initialize $Q(s, a)$, for all $s \in S$, arbitrarily, and $Q(terminal - state, .) = 0$
Repeat(for each episode):
    Initialize s
    repeat(for each step of episode):
        Choose A from S using derived from Q
        Take action a, observe r, s'
        $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_a Q(s', a) - Q(s, a)]$
        $s \leftarrow s'$
    until s is terminal

1.4 Gradientové TD učenie

V tejto podkapitole upustíme od tabuľkového spôsobu riešenia TD metód a pozrieme sa na zaujímavejší spôsob - riešenie pomocou aproximácie funkcie. Tento spôsob prináša

niekoľko výhod. Dokáže generalizovať, čiže vie dobre odhadnúť výstup aj pre úplne nové vstupy. Vyžaduje omnoho menšiu pamäťovú náročnosť. Zaoberať sa budeme nelineárnou aproximáciou, kde aproximátor bude dopredná neurónová sieť, ktorú si priblížime neskôr v práci. Každý aproximátor, či už lineárny alebo nelineárny, obsahuje sadu aktualizovateľných parametrov a hyperparametrov, ktoré rozhodujú o hodnote výstupu aproximátora. Existuje viacero druhov optimalizácií, no my si bližšie popíšeme iba gradientové metódy, konkrétne algoritmus zvaný gradientový zostup (gradient descent). Princíp aproximácie funkcií môžeme aplikovať na tri rôzne úrovne MDP (van Hasselt, 2012):

1. Aproximácia modelu MDP,
2. Aproximácia hodnotových funkcií,
3. Aproximácia stratégie.

Priblížime si iba druhý spôsob riešenia MDP aproximáciou hodnotovej funkcie.

1.4.1 Gradientový zostup

Pred použitím tohto algoritmu si musíme zadefinovať diferencovateľnú chybovú funkciu, t.j. funkciu, ktorú chceme optimalizovať - chceme nájsť jej extrém (maximum alebo minimum). Gradient parametrov aproximátora je vektor parciálnych derivácií chybovej funkcie podľa každého parametra. Parciálna derivácia chybovej funkcie podľa daného parametra hovorí, aká je citlivosť (závislosť) chyby od zmeny tohto parametra (v aktuálnom stave). Ak máme vypočítaný gradient a chceme funkciu minimalizovať, tak zmenu parametrov robíme proti gradientu, čím dosiahneme menšiu hodnotu chybovej funkcie. Tento postup je vyjadrený v gradientovom pravidle aktualizácie parametrov:

$$W_i = W_i - \alpha \frac{\partial E}{\partial W_i} \tag{1.15}$$

kde W sú aktualizované parametre, E je chybová funkcia a α je rýchlosť učenia, ktorá zabezpečuje veľkosť posunu. Pravidlo 1.15 iteratívne aplikujeme, až po kritérium zastavenia, keď sa rozhodneme ukončiť učenie (van Hasselt, 2012).

1.4.2 Neurónová sieť

Pre aproximáciu Q hodnôt sme zvolili štandardný dopredný model neurónovej siete - viacvrstvový perceptrón (Multilayer perceptron - MLP). Takýto model siete sa trénuje pomocou gradientového zostupu (gradient descent). Na výpočet gradientu neurónovej siete sa používa algoritmus Back-propagation (Rumelhart, Hinton, Williams, 1986) založený na spätnom šírení chyby. Teoretické odôvodnenie, prečo aproximácia ľubovoľnej funkcie neurónovou sieťou funguje, nám poskytne teorém o univerzálnom aproximátore (Baldi, Hornik, 1989).

Model pozostáva zo vstupnej vrsty, niekoľkých skrytých vrstiev a výstupnej vrstvy. Každá vrstva má určitý počet neurónov, ktoré sú váhami poprepájané medzi vrstvami. Informácia zo vstupu je postupne transformovaná jednotlivými vrstvami na výstup. Transformácia je realizovaná lineárnymi kombináciami neurónov z predchádzajúcej vrstvy a váhami, ktorých výsledok je prenásobený nelineárnou aktivačnou funkciou, aby sieť dokázala generalizovať aj zložitú štruktúru dát alebo povahu aproximovanej funkcie.

Pre každú vrstvu môžeme zvoliť inú aktivačnú funkciu a spolu s počtom neurónov a vrstiev tvoria hyperparametre modelu. Okrem toho je dôležitá aj inicializácia váh, zvolenie rýchlosti učenia, normalizácia vstupného vektora atď.

Trénovateľné parametre neurónovej siete tvoria matice váh $\mathbf{W}$ a prahové vektory b. Princíp tréningu siete, jednoduchého dopredného typu pozostáva z dopredného prechodu, spätného prechodu a gradientového pravidla.

Dopredný prechod

Vstupný vektor x transformujeme prechodom jednotlivými vrstvami, až kým na výstupnej vrstve nezískame odhad výslednej hodnoty. Pre chybovú funkciu volíme zvyčajne strednú kvadratickú chybu (MSE). Princíp fungovania modelu si ukážeme nasledovnými vzťahmi, pričom zvolíme maticový spôsob zápisu:

$$\begin{aligned} a^{(i)}(x) &= f^{(i)}(\boldsymbol{W}^{(i)} a^{(i-1)}(x) + b^{(i)}) \\ E &= \frac{1}{n} \sum_{x}^{n} \frac{1}{2} (d(x) - a^{i}(x))^2, \ ak \ i = L \end{aligned} \tag{1.16}$$

kde n je celkový počet vzoriek, na ktorých sieť trénujeme a L je celkový počet vrstiev siete. Člen $f^{(i)}$ je vektorová aktivačná funkcia pre i-tu vrstvu, $\boldsymbol{W}^{(i)}$ je matica váh medzi i-tou a $i-1$ vrstvou a b je prahovým vektorom pre i-tu vrstvu. Výstupom $i-1$ vrstvy je vektor $a^i(x)$ a člen $d(x)$ vyjadruje želaný výstup pre vstupnú vzorku x. MSE je v rovnici 1.16 vyjadrená členom E, ktorý je vyrátaný ako súčet n vzoriek kvadratických chýb medzi $a^L(x)$ a $d(x)$.

Spätný prechod

Keď poznáme chybu na výstupnej vrstve, vypočítame gradient pre každú váhu siete pomocou parciálnej derivácie chybovej funkcie podľa danej váhy siete. Derivácia po aplikovaní tzv. reťazového pravidla (*chain rule*) na výstupnej a skrytej vrstve vyzerá nasledovne:

$$\delta^i = \begin{cases} (d - a^i) f'^i(\boldsymbol{W}^i a^{i-1}) & i = L \\ \boldsymbol{W}^{i+1}\delta^{i+1} f'^i(\boldsymbol{W}^i a^{i-1}) & \text{pre } i \in \{1, 2, ..., L-1\} \end{cases} \tag{1.17}$$

Zapísali sme zjednodušene, kde $a^i(x) = a^i$ a rovnako pre ostatné členy.

Gradientové pravidlo

Teraz môžeme aplikovať pravidlo gradientu 1.15 pre aktualizáciiu váh:

$$\boldsymbol{W}^i = \boldsymbol{W}^i - \alpha \frac{\partial E}{\partial \boldsymbol{W}^i} \tag{1.18}$$

kde α je rýchlosť učenia v intervale $[0, 1]$. Príliš vysoká hodnota ma za následok divergenciu a príliš nízka zase spomalí učenie a riešenie môže ostať v nepostačujúcom lokálnom minime. Deriváciou chyby $\frac{\partial E}{\partial \boldsymbol{W}^i}$ získame výraz $\delta^i a^{i-1}$ a deriváciu chyby $\frac{\partial E}{\partial b^i}$ podľa *prahu* získame výraz δ^i. V tejto podkapitole sme vychádzali zo zdrojov (Nielsen, n.d.) a (Haykin, 2009).

1.4.3 Aproximovaný Q-learning

Vráťme sa teraz späť k TD učeniu a ukážme si ako vyzerá TD predikcia pomocou aproximátora. Hodnotovú funkciu vyjadríme v jej parametrizovanej forme (van Hasselt,

2012):

$$V_t(s) = V(\theta_t, \phi(s)), \tag{1.19}$$

kde V je už opisovaná neurónová sieť a θ_t je vektor jej parametrov (t.j. váh a prahov) v čase t. Člen $\phi(s)$ vyjadruje vektor atribútov stavu s. Derivácia chyby TD predikcie pre Q-learning (van Hasselt, 2012):

$$\nabla_\theta E_t(s_t, a_t, \theta) = -(r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t))\nabla_\theta Q_t(s_t, a_t) \tag{1.20}$$

Ak poznáme deriváciu chybovej funkcie, zakomponujeme ju do gradientového pravidla 1.15, čím získame výraz pravidla učenia:

$$\theta_{t+1} = \theta_t - \alpha_t(s_t)\nabla_\theta E_t(s_t, a_t, \theta) \tag{1.21}$$

Pri aproximovanej forme Q-learningu sa vyskytuje problém so zaručenou konvergenciou. Problém okrem iného spočíva v použití samotého aproximátora, ktorý môže obsahovať priveľa šumu kvôli jeho stochastickej povahe. To môže zapríčiniť priveľkú odchýlku medzi odhadnutou hodnotou aproximátorom $\max_a Q_t(s, a)$ a optimálnou $\max_a Q^*(s, a)$. Takáto odchýlka môže zapríčiniť pomalú konvergenciu alebo dokonca model aproximátora môže divergovať (van Hasselt, 2012).

1.5 Exploračné stratégie

Fundamentálnym problémom v RL je balans medzi exploráciou a exploitáciou prostredia. Teda otázkou je, či má agent vybrať najlepšiu možnú akciu pre daný okamih - exploitovať. Alebo vybrať akciu inú s očakávaním, že to bude viesť v budúcnosti k vyššej kumulovanej očakávanej odmene - explorovať.

Zvolenie vhodnej explorácie je kľúčové z hľadiska dĺžky tréningu agenta, ako aj z využitia výpočtových zdrojov. Čím efektívnejšie agent spoznáva prostredie exploráciou, tým menej času potrebuje na tréning. Z pohľadu exploitácie, zase chceme minimalizovať tzv. cenu tréningu, alebo inak povedané, chceme maximalizovať výkonnosť agenta. Tým máme na mysli to, že i keď agent splnil danu úlohu, jej dokončenie ho stálo priveľa prostriedkov - jednoducho bolo príliš drahé. Ťažkou úlohou vo zvolení vhodnej

exploračnej stratégií teda kompromis medzi exploráciou a exploitáciou, čiže čím menší potrebný čas na tréning, tým vyššia jeho cena. Väčšina metód obsahuje parametre, ktoré možno ladiť používateľom, čím sa snažia bojovať s touto dilemou.

Existuje viacero metód, ktoré sa snažia kombinovať exploráciu s exploitáciou rôznymi spôsobmi. Súčasne explorovať a exploitovať alebo v rôznych kombináciach najprv explorovať a potom exploitovať, alebo naopak.

Ďalším negatívnym faktom je, že medzi exploračnými metódami je slabá znovupoužiteľnosť metód. Rovnaká explorácia, ktorá dobre funguje v jednom prostredí, už nemusí byť vôbec optimálnou v podobnom prostredí a to i napriek tomu, že obe prostredia sa líšia iba minimálne (McFarlane, 2003).

1.5.1 Stochastické explorácie

Stochastické explorácie sú charakterizované tým, že neberú do úvahy históriu postupu učenia tréningu a akcie sa generujú len na základe pravdepodobnostnej distribúcie, ktorej veľkosť je rovná počtu akcií. Explorácie spadajúce do tohto typu majú taktiež väčšinou spoločné to, že menia túto distribúciu zvyčajne pomocou hyperparametra, ktorý ju mení buď smerom k explorácii alebo k exploitácii (McFarlane, 2003).

ε-chamtivá

ε-chamtivá stratégia patrí medzi najjednoduchšie exploračné stratégie. Je to kombinácia *náhodného* a *chamtivého* výberu akcie. Je definovaná nasledovným vzťahom:

$$\pi(s) = \begin{cases} \max_{a \in A} Q(s,a) & \text{s pravdepodobnosťou } 1-\epsilon \\ \text{náhodná akcia} & \text{inak} \end{cases} \tag{1.22}$$

Agent vyberie akciu v danom stave s najvyššou hodnotou Q funkcie s pravdepodobnosťou $1-\varepsilon$, inak vyberie náhodnu akciu. Takto vieme nasimulovať proces, keď agent postupom času prostredie spoznáva viac a viac, čiže výber náhodných akcií sa časom znižuje a zároveň výber akcie s maximálnou priradenou Q hodnotou sa zvyšuje. To zabezpečíme znižovaním hyperparametra ε o nejakú konštantu. Náhodný výber akcie sa vykonáva z uniformnej distribúcie (Tijsma, Drugan, & Wiering, 2016).

Boltzmannová metóda

Táto explorácia je založená na exponenciálnej funkcii softmax. Po aplikovaní tejto funkcie na vektor Q hodnôt - akékoľvek reálne čísla, získame vektor pravdepodobností. Vytvorená pravdepodobnostná distribúcia rešpektuje pomery pôvodných Q hodnôt a teda najväččia z nich bude mať aj vo výslednej pravdepodobnosti najväčšiu hodnotu (Tijsma et al., 2016).

$$\pi(s_t, a) = \frac{e^{Q_t(s_t,a)/T}}{\sum_{i=1}^{m} e^{Q_t(s_t,a_i)/T}}, \tag{1.23}$$

kde m je počet akcií a T je hyperparameter teploty usmerňujúci kompromis medzi exploráciou a exploitáciou. Pre $T \to \infty$ vyberáme akcie náhodne ako pri ε-chamtivá. Pre $T = 0$ vyberáme iba akciu s aktuálne najvyššou Q hodnotou.

Pursuit

Pursuit je stochastická explorácia definovaná iteratívnymi pravidlami, ktoré upravujú parametre distribúcie pravdepodobností na základe ich predchádzajúcich parametrov:

$$\begin{aligned} \pi_{t+1}(s_t, a^*_{t+1}) &= \pi_t(s_t, a^*_{t+1}) + \beta[1 - \pi_t(s_t, a^*_{t+1})] \\ \pi_{t+1}(s_t, a) &= \pi_t(s_t, a) + \beta[0 - \pi_t(s_t, a)] \; pre \; \forall \; a \neq a^*_{t+1}, \end{aligned} \tag{1.24}$$

kde β určuje rýchlosť učenia preferovaných akcií. Čím je bližšie k hodnote 1, tým s väčšou pravdepodobnosťou vyberieme chamtivú akciu a^*_{t+1}. Po každom časovom kroku t pravdepodobnosť akcie a^*_{t+1} definovanej ako $\arg\max_a Q_{t+1}(s_t, a)$ upravíme podľa prvej rovnice z 1.24 a všetky ostatné pravdepodobnosti poďla druhej rovnice z 1.24. Samotný výber akcie zrealizujeme podobne ako pri Boltzmannovej metóde (Tijsma et al., 2016).

1.5.2 Explorácia vnútornou motiváciou

Vnútorná motivácia (Intrinsic motivation - IM) je koncept, ktorý podľa (Chentanez, Barto, & Singh, 2005) sa snaží agenta motivovať k tomu, aby svojvoľne skúmal zaujímavé oblastí stavového priestoru. Exploračné stratégie založené na IM uchovávajú tzv. znalosť o tom, ako prebieha proces učenia. V tomto kontexte odmeňovaciu funkciu $\mathcal{R}$ vnímame ako zdroj externej motivácie agenta, ktorú získava z prostredia. IM je interným zdrojom odmeny, ktorú agent samostatne generuje počas tréningu. *Explo-*

račný bonus je kvantifikovaný typ odmeny IM a existujú dva spôsoby ako ho môžeme zakomponovať do procesu explorácie:

- **modifikovanie odmeny** - *bonus* pripočítame priamo k odmene, ktorú agent získava z prostredia: $r_t = r_t + bonus_t(s_t, a)$,
- **modifikovanie behaviorálnej stratégie** - bonus pripočítame ku Q odhadom pri výbere akcie: $a_t = \arg\max_a Q(s_t, a) + bonus_t(s_t, a)$.

Mnoho metód realizuje IM myšlienku rôznymi spôsobmi, my si však uvedieme iba *count-based* exploráciu.

Count-based explorácia

Ukážeme si count-based exploráciu (Pecháč, 2019), ktorá modifikuje odmenu a pre každý stav uchováva počet návštev. Vzťah pre odmenu má tvar:

$$r_t = r_t + \frac{\beta}{\sqrt{n(s_t)}}, \tag{1.25}$$

kde β je škálovací faktor, ktorého hodnota závisí od odmeňovacej funkcie konkretného prostredia a $n(s_t)$ určuje počet návštev stavu s v čase t. Tento člen môže byť nahradený aj $n(s_t, a)$, čo znamená počet návštev stavu s pri vykonaní akcie a. Čím častejšie agent stav navštívi, tým menšiu odmenu získa. Teda núti ho to skúmať nové stavy. Okrem rátania počtu stavou existujú ďalšie alternatívy ako uvádzajú (McFarlane, 2003):

- *Error-based* - prehľadávame oblasti prostredia, v korých odhad TD chyby bol najväčší,
- *Recency-based* - myšlienka spočíva vo výbere takých stavov, ktoré neboli navštívené najdlhší čas.

Upper Confidence Bound 1 (UCB-1)

(Tijsma et al., 2016) uvádzajú UCB-1 stratégiu, ktorá je taktiež verziou count-based explorácie kde exploračným bonusom modifikujeme behaviorálnu stratégiu. Upravili pôvodnú UCB exploráciu, aby bola použiteľná pre Q-learning a pre nenormalizovanú odmeňovaciu funkciu. Myšlienkou UCB-1 je vyjadriť hodnotu neistoty, ktorú agent má

pre každú akciu a cieľom je vybrať tú s najvyššou hodnotou.

$$bonus(s_t, a^i) = 100 \times C \times \sqrt{\frac{2 \times \log N(s_t)}{N(s_t, a^i)}}, \tag{1.26}$$

kde C je hyperparameter ovplyvňujúci mieru explorácie. Pri $C = 0$ neprebieha žiadna explorácia. Čím častejšie danú akciu vyberieme, tým sme si ňou viac istejší a preto hodnota bonusu klesá.

Pseudo-counts

Predchádzajúce metódy majú predpoklady byť efektívnymi, avšak iba pre diskrétny stavový priestor prostredia MDP s malým počtom stavov. Ako zložitosť stavového priestoru MDP narastá, stávajú sa tieto metódy nepoužiteľnými. Dôvodom je malá alebo žiadna šanca, že daný stav agent navštívi viac krát ako aj praktické dôvody z hľadiska pamäťovej náročnosti. (Bellemare et al., 2016) ponúkajú riešenie, kde namiesto rátania stavov prostredia, odhadujú pravdepodobnosť výskytu stavu v preskúmanej množine stavov. Definujú *model hustoty* (density model), ako model ktorý predpokladá, že stavy majú medzi sebou nezávislú distribúciu, preto môžeme takýto model považovať za akýsi typ generatívneho modelu (učí sa pravdepodobnostnú distribúciu z ktorej sú vzorky (v tomto prípade stavy) vygenerované).

Aby sme mohli odvodiť *pseudo-counts* od *modelu hustoty*, potrebujeme si najskôr zadefinovať ako spomínanú pravdepodobnosť výskytu stavu vyjadriť daným *modelom*:

$$\begin{aligned} \rho_n(x) &= \rho(x; s_{1:n}) \\ \rho'_n(x) &= \rho(x; s_{1:nS}), \end{aligned} \tag{1.27}$$

kde s je konečná sekvencia stavou. Člen $\rho_n(x)$ *modelu hustoty* určuje pravdepodonosť výskytu stavu s v sekvencií dĺžky n označenej ako $1{:}n$. Člen $\rho'_n(x)$ je tzv. *recoding probability*, t.j. pravdepodobnosť, že daný stav s by bol videný v sekvencii ešte raz $(1{:}nS)$. *Model hustoty* môže byť implementovaný generatívnym modelom. *Pseudo-counts* $\hat{N}_n(s)$ je funkcia, ktorá vracia odhadovaný počet návštev stavu s a člen $\hat{n}$ vyjadrujúci odhad celkového počtu návštev stavu s sekvencie dĺžky n. Tieto dva členy odvodiť pomocou dvoch lineárných rovníc, ktoré majú vlastnosť, že pri pozorovaní stavu s príde k

jednotkovému nárastu *pseudo-count* pre daný stav.

$$\begin{aligned} \rho_n(x) &= \frac{\hat{N}_n(s)}{\hat{n}} \\ \rho'_n(x) &= \frac{\hat{N}_n(s)+1}{\hat{n}+1}, \end{aligned} \tag{1.28}$$

Na základe rovníc 1.28 môžeme definovať vzťah, na základe ktorého získame *pseudo-counts*:

$$\hat{N}_n(s) = \frac{\rho_n(s)(1-\rho'_n(s))}{\rho'_n(s)-\rho_n(s)} \tag{1.29}$$

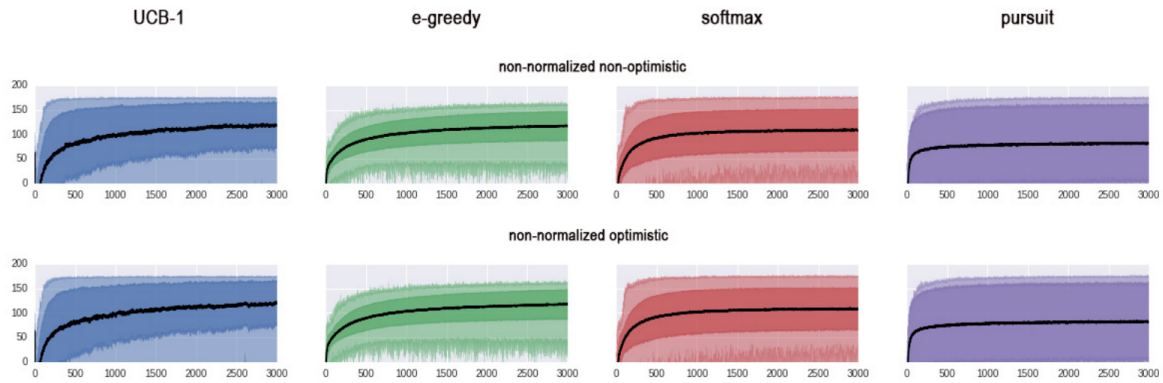
Takto definovaný *pseudo-counts* môže byť pripočítaný k bonusu ako sme si to ukázali v rovnici 1.25. Môže byť tiež vnímaný ako *aproximátor explorácie*.

Kapitola 2

Prehľad súvisiacej problematiky

V tejto časti práce si spomenieme niektoré súvisiace práce zaoberajúce sa výskumom exploračných stratégií. Explorácie v učení posilňovaním sú predmetom výskumu, aj pretože sú stále otvoreným problémom. Spomedzi veľkého množstva publikovaných prác a vedeckých článkov si vyberieme len tie pre nás najzaujímavejšie. Uvedieme si tri práce, pričom každá vychádza z iného pozadia. Prvou prácou je článok *Comparing Exploration Strategies for Q-learning in Random Stochastic Mazes*, ktorý bol pre nás najpodstatnejší, pretože sme sa zaoberali podobnými metódami explorácií a aj podobným dizajnom experimentov. Ostatné dva vedecké články *Count-Based Exploration with Neural Density Models* a *Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning*, sa špecificky zaoberajú problematikou pseudo-counts, ktorú sa snažia riešiť odlišnými spôsobmi.

V práci (Tijsma et al., 2016) autori experimentovali so stochastickými bludiskami v ktorých sa snažili nájsť taktiež najkratšiu cestu. Bludiská mali tvar mriežok veľkostí 10×10 a 20×20. Pozície prekážok v bludisku okrem agenta a cieľa boli generované náhodne. Učenie zrealizovali pomocou tabuľkového Q-learningu. Pri výbere exploračných metód sme sa inšpirovali práve touto prácou. Skúmali explorácie ε-greedy bez poklesu, Boltzmannovu metódu, tabuľkový Pursuit a taktiež UCB-1exploráciu. Okrem prehľadávania hyperparametrov explorácií autori rozšírili experimenty aj o verziu normalizovanej odmeny v intervale $[0, 1]$. Taktiež v experimentoch rozlišovali spôsob inicializácie Q hodnôt v tabuľke - optimistická a neoptimistická. Ukážeme si krivky učenia pre dva typy experimentov podľa jednotlivých explorácií v rámci všetkých prehľadávaných hodnôt ich hyperparametrov:

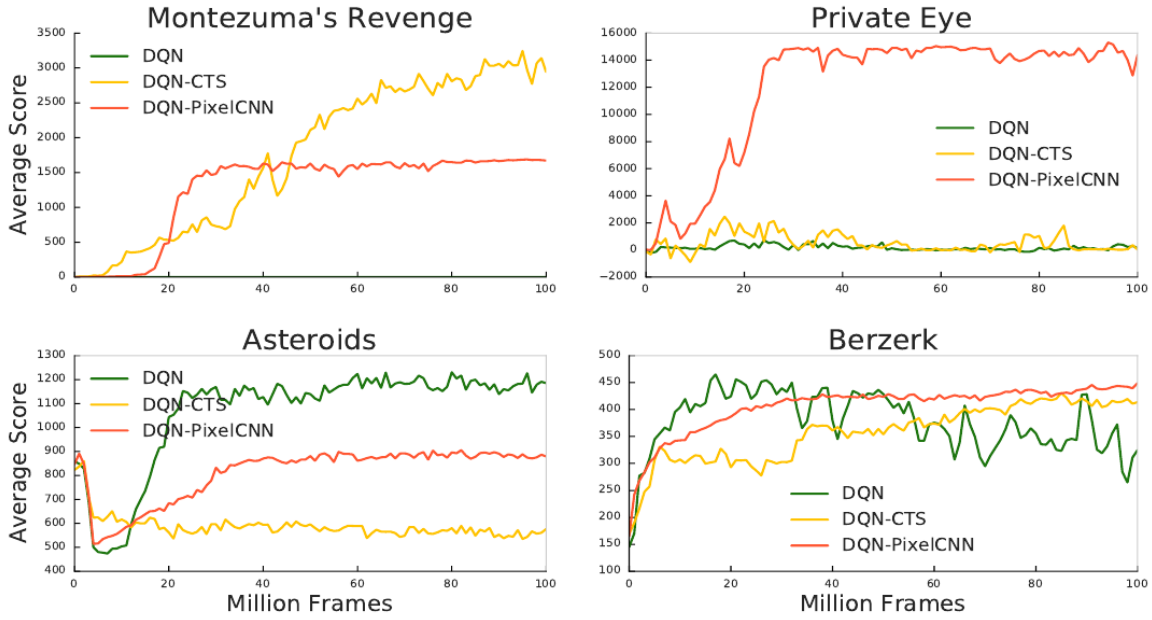


Obr. 2.1: Krivky učenia jednotlivých metód pre nenormalizovanú odmenu. Prebraté z (Tijsma et al., 2016).

Výsledky tvorili priemer zo simulácií, v ktorých pre každú vygenerovali náhodné nové bludisko. Najlepšiu efektivitu dosiahla Boltzmannova metóda a ε-greedy obstál najhoršie aj z hľadiska stability. Po prehľadávaní pomerne veľkého priestoru hyperparametrov jednotlivých explorácií vo výsledkoch uvádzajú, že UCB-1 obstála najlepšie čo sa týka výkonnosti a taktiež v jej spočíva v ľahkom nastavovaní hyperparametrov. Pursuit implementovaný tabuľkou najrýchlejšie konvergoval, i keď to bolo za cenu asymptotickej výkonnosti. ε-greedy stratégia obstála najhoršie aj z hľadiska stability. V prípade bludísk veľkosti 20×20 však explorácie už nedokázali nájsť optimálne stratégie.

Práca autorov Ostrovski, Bellemare, Oord, and Munos (2017) vychádza z práce (Bellemare et al., 2016), v ktorej definovali pojem *pseudo-counts* a ktorý sme si objasnili v kapitole 1. Hlavným úsilím tejto práce bolo skúmanie, akú dôležitú úlohu pri explorácii pomocou *pseudo-counts* zohráva implementácia *modelu hustoty*. Pri výbere modelu sa rozhodli zvoliť model konvolučnej neurónovej siete - PixelCNN predstavenú v práci (van den Oord, Kalchbrenner, & Kavukcuoglu, 2016), ktorá dosiahla úspech pri modelovaní distribúcie obrázkov. Zostrojili niekoľko experimentov s viacerými RL architektúrami, avšak spomenieme si iba tú, ktorá je založená na architektúre hlbokého Q-learningu (DQN) (V. Mnih et al., 2015). V experimentoch pre realizáciu prostredia zvolili simulátor Atari hier, ktorého voľba je už v podstate štandardom pri porovnávaní najlepších RL algoritmov vo výkonostných testoch. Je to kvôli ich zložitému priestoru a riedkej odmene, ktorými Atari hry väčšinou disponujú. V prvej sade experimentov na 57 Atari hrách porovnávali navzájom tri modely. Prvý model tvoril už spomínaný pôvodný DQN agent. Druhým modelom bol agent DQN-CTS, ktorého autori predstavili

v spomínanej práci (Bellemare et al., 2016). Pôvodnú sieť PixelCNN upravili na jednoduchšiu verziu, aby bola zložitosťou porovnateľná s DQN agentom. Porovnávali teda tri modely, pričom dva z nich (CTS a PixelCNN) mali exploračné metódy založené na vnútornej motivácii, realizovanej odlišnými spôsobmi. Tréning všetkých modelov vykonali online (kvôli predpokladom stanoveným *modelom hustoty*). Po každom časovom kroku *model hustoty* po prijatí snímky zo simulátora aktualizuje svoje parametre a vráti *exploračný bonus* - viac o detailoch modelov je možné nájsť v danej práci. Nasledujúci obrázok ukazuje grafy kriviek učenia jednotlivých modelov pre 4 vybrané prostredia:



Obr. 2.2: Krivky učenia spomínaných modelov vo vybraných Atari hrách. Hore sú ťažké, dole ľahké hry. Prebraté z (Ostrovski et al., 2017).

Horné obrázky predstavujú extrémne náročné verzie prostredia. Model DQN-PixelCNN dosiahol na hre *Private Eye* výrazne najlepšie výsledky (taktiež aj v experimente *Venture* - možné pozrieť v práci). Dôvod prečo základná DQN metóda dosiahla väčšiu výkonnosť oproti zvyšným dvom pri experimentoch s ľahším typom hier (spodná časť obrázku), autori odôvodňujú exploračným bonusom možné vychýlenie odmeny prostredia. Model založený na sieti PixelCNN prekonal model CTS a z celkového počtu experimentov vyhral model PixelCNN v 52 prípadoch z 57. Zhodnotenie z celkového hľadiska je také, že generatívny model siete PixelCNN je robustnejší a ním generovaný exploračný bonus je postačujúci pre rýchly a úspešný tréning agenta a to aj v zložitých prostrediach s riedkou odmenou.

Práca autorov Tang et al. (2016) sa zaoberala iným prístupom ako zovšeobecniť empirické počítanie stavov aj pre zložitejšie domény. Predstavujú pomerne jednoduchú metódu, ktorá preukázala prekvapivo dobré výsledky. Myšlienkou práce je použiť hešovaciu funkciu pre účely diskretizácie stavu získaného z prostredia. Výpočet exploračného bonusu je teda rovnaký ako sme si uviedli v rovnici 1.25 až na to, že rátajú počet výsledkov po aplikovaní hešovacej funkcie pre daný stav. Výber hešovacej funkcie je dôležitý z hľadiska výkonnosti tréningu modelu, pretože ňou definujú spôsob akým chcú generalizovať stavy a rozlišovať jednotlivé stavy medzi sebou. Výhodné je nájsť takú funkciu, aby stavy, ktoré sú od seba ďalej rátala ako odlišné a stavy, ktoré su bližšie rátala ako podobné. Preto zvolili typ hešovacej funkcie tzv. LSH (locality sensitive hashing), pretože pre podobný vstup táto funckia vráti s veľkou pravdepodobnosťou rovnaký heš kód. Rozlišujú medzi dvoma typmi hešovacích funkcií - statickou a učenou. Pri statickom type priamo aplikujú konkrétny typ LSH funkcie na stav a pri učenom type funkcie natrénujú autoenkóder, ktorý pre vstup v podobe stavu vygeneruje heš kód. Zostavené modely experimentálne porovnali taktiež na Atari hrách, avšak ich výsledky neprekonali tie najlepšie.

Veľmi krátko si uvedieme ďalšie zaujímavé práce, ktoré sa venujú problematike explorácii a je v nich možné získať množstvo informácií. Práca (Semmler, 2017) ponúka široký záber na viaceré metódy exploračných stratégií. Mimo explorácií, ktoré sme spomínali v našej práci sa venujú taktiež metódam ako *Bootstrapped DQN*, čo je využitie algoritmu *Thomson sampling* na modifikovanie DQN metódy tak, aby aproximovala distribúciu Q-hodnôt pomocou *bootstrapu*. Uvádzajú taktiež exploráciu, ktorá kombinuje predchádzajúcu s UCB metódou, tzv. *UCB Ensemble Exploration*. Experimenty zostavujú so zložitejšími bludiskami a s inými komplexnými prostrediami. Ďalšia práca autorov Hester, Lopes, and Stone (2013) sa venuje exploračným metódam v MDP, v ktorom model prostredia je známy. Na záver si uvedieme prácu (Coggan, 2004), ktorá ponúka veľmi rozsiahli rozbor exploračných stratégií a aj samotných algoritmov RL.

Kapitola 3

Výpočtové experimenty

V tejto časti práce využijeme nadobudnuté znalosti z kapitoly 1 a na ich základe zostavíme experimenty. Opíšeme úlohu agenta a akým algoritmom učenia v spolupráci s exploráciami ju bude riešiť. Po bližšom popísaní realizácie explorácií si opíšeme prostredie - MDP model, v ktorom agent pôsobí a nakoniec si priblížime samotný dizajn experimentov a spôsoby ich evaluácie.

Pri konštruovaní experimentov sa inšpirujeme prácou (Tijsma et al., 2016). Úlohou nášho RL agenta je nájsť najkratšiu možnú cestu v jednoduchom bludisku. Cieľom experimentov je porovnať výkonnosť agenta poháňaného exploračnými stratégiami, ktoré určujú spôsob navigácie v prostredí - bludisku. V rámci stochastických explorácií skúmame správanie agenta založeného na jednoduchej metóde ε-chamtivej s dvoma typmi poklesu epsilonu. Pri Boltzmannovej metóde by mala byť explorácia omnoho efektívnejšia v porovnaní s predchádzajúcou. To iba za predpokladu, ak nájdeme optimálnu hodnotu hyperparametra teploty, ktorá rozhoduje o miere explorácie. Nakoniec sa pozrieme na Pursuit exploráciu, ktorej správanie je pre nás najzaujímavejšie kvôli implementácií pomocou neurónovej siete. Obzvlášť sme zvedaví ako si agent poradí s riešením úlohy, ak jeho správanie poháňa typ IM explorácie implementovanej jednoduchým počítaním stavov $N(s)$ a ako v prípade zložitejšej UCB-1 explorácie.

3.1 Algoritmus učenia

Typ algoritmu učenia sme zvolili Q-learning, pretože navzdory jeho jednoduchej implementácií vie byť dostatočne efektívny. Efektivita a robustnosť algoritmu vzrastá o to viac, ak Q tabuľku nahradíme Q funkciou implementovanou doprednou neurónovou sieťou.

Algorithm 3.1 Aproximovaný on-line Q-learning

Inicializuj Q sieť hodnotovej funkcie náhodnými váhami
Repeat(pre každú epizódu e):
  Inicializuj s_0
  repeat(pre každý krok t z epizódy):
    Vyber akciu a_t pomocou behaviorálnej stratégie
    Vykonaj akciu a_t, získaj odmenu r_t a nový stav s_{t+1}
    Nastav $y_t = \begin{cases} r_t & \text{pre terminálny stav } s_{t+1} \\ r_t + \gamma \times \max_{a'} Q(a', s_{t+1}; \theta) & \text{pre neterminálny stav } s_{t+1} \end{cases}$
    Vykonaj gradient descent pre $(y_t - Q(a_t, s_t; \theta)^2$ podľa vzťahu 1.15
    $s_t = s_{t+1}$
  until stav s_t je terminálny alebo $t > \theta$

Algoritmus 3.1 podľa (Mnih et al., 2013) poskytuje detailný pseudokód aproximovanej verzie Q-learningu. Vyhotovenie experimentov sa skladá z troch komponentov: 1) *Simulátor prostredia*, 2) *exploračná stratégia* – teda implementovanie behaviorálnej stratégie a 3) model neurónovej funkcie pre Q-sieť, ktorá zabezpečuje fázu učenia agenta. Všetky komponenty ako aj problémy, na ktoré sme počas realizácie experimentov narazili, si detailne priblížime v nasledujúcich podkapitolách.

3.2 Simulátor postredia

Ako simulátor prostredia sme zvolili jednoduchú dvojrozmernú mriežku - bludisko. Formálne predstavuje tento typ prostredia diskrétny stavový priestor MDP. Zvolili sme ho kvôli jeho jednoduchosti a kvôli tomu, že veľmi dobre zodpovedá ľudskej intuícii o explorácií prostredia ako takej. Experiment rozšírime o simulácie z hľadiska **zložitosti**. Ostaneme ale stále pri relatívne jednoduchom stavovom priestore, pretože pre potreby nášho experimentu nám to postačí a taktiež aj kvôli obmedzeným výpočtovým prostriedkom. Ukážme si konkrétne mapy bludísk použité v experimentoch:

(a) 5×5

(b) 8×8

Obr. 3.1: Štruktúra bludísk použitých v experimentoch.

Mapa 8×8 predstavovala náročnejšiu verziu prostredia. Agent sa v ňom môže pohybovať štyrmi smermi - hore, dole, doprava a doľava. Po bludisku sa agent pohybuje voľne až kým nenarazí na okraj bludiska - to ho vráti späť do predošlého stavu. Rovnaké správanie nastane aj keď narazí do prekážok (B), ktoré su rozmiestnené po bludisku. Okrem prekážok bludisko obsahuje ešte dva typy políčok - cieľ (G) a priepasť (H). Obe políčka vyjadrujú koniec hry a zároveň reset parametrov explorácie.

3.2.1 Odmeňovacia funkcia

Keďže náš agent sa učí na základe odmeny, musíme zadefinovať odmeňovaciu funkciu $\mathcal{R}$ prostredia. Odmeňovacia funkcia sa zvykne deliť do dvoch kategórií, a to na *hustú* a *riedku*. Všeobecne platí, že husto nadizajnovaná odmena urýchli konvergenciu, takže preto zvolíme tento typ. Konkrétne hodnoty pri definovaní takejto funkcie sú celkom svojvoľné, platí však, že hodnota pre dosiahnutie želaného cieľa alebo výsledku by mala byť podstatne vyššia ako hodnota pre ostatné stavy. Napriek tomu nebolo definovanie odmeny úplne priamočiare. Problém vznikol hlavne pri Pursuit metóde, kde tréning agenta prebiehal v spolupráci dvoch neurónových sietí. (Viac k tomu v časti 3.4.1.) Po viacerých pokusoch sme dospeli k nasledovnému nastaveniu hodnôt odmien:

	Odmena
krok	-1
blokáda (B)	-10
priepasť (H)	-20
cieľ (G)	100

Povaha prostredia

Pod povahou prostredia máme na mysli spôsob akým prostredie reaguje na požiadavky agenta. Simulátor vykoná danú akciu alebo ju vykoná iba s určitou pravdepodobnosťou. Z hľadiska typu prechodovej funkcie $\mathcal{T}$ modelu MDP rozlišujeme na dva typy prostredia. **Deterministický** typ prostredia vykoná presne takú akciu, o ktorú ho agent požiada. Môžme povedať, že ide o spoľahlivý simulátor. Pri **stochastickom** type zavedieme do simulátora prostredia náhodnú premennú, takže sa vykoná želaná akcia no jej výsledok nemusí byť želaný. Pri tomto type sú teda okolnosti tréningu agenta sťažené, pretože narozdiel od predošlého typu, toto prostredie sa nemusí správať podľa jeho očakavání a agent potrebuje k tréningu viac vzoriek. Pre náš experiment sme zvolili verziu, kde nám prostredie vykoná požadovanú akciu s 80% pravdepodobnosťou a so zvyšnou 20% pravdepodobnosťou prostredie vyberie akciu inú. Ukážme si konkrétny príklad pre zostavenie prostredia, ak behaviorálna stratégia vybrala akciu pre pohyb smerom hore:

akcia	hore	dole	vľavo	vpravo
pravdepodobnosť	0.8	0.067	0.067	0.067

Čiže prostredie pri výbere akcie bude pracovať s rovnomernou pravdepodobnostnou distribúciou definovanou pre množinu akcií (ak sa nezvolí akcia s 80% pravdepodobnosťou).

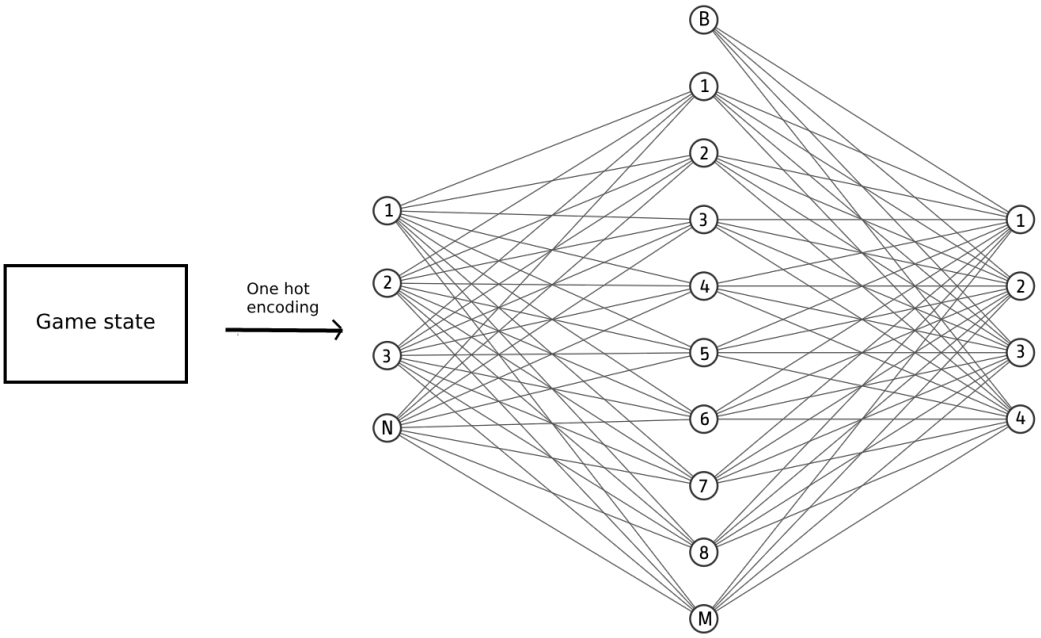
3.3 Model siete Q-funkcie

V tejto podkapitole si opíšeme neurónovú sieť, ktorá reprezentuje Q funkciu algoritmu 3.1. Autori (Mnih et al., 2013) pri výbere typu architektúri siete uvádzajú dve možností.

Stav-akcia $\rightarrow$ Q hodnota. V prvom prípade sieť dostane na vstupnej vrstve dvojicu stav-akcia a na výstupe nám vráti odhad skalárnej Q hodnoty pre túto dvojicu. V tomto prípade je počet pustení siete rovný kardinalite množine akcií.

Stav $\rightarrow$ Q hodnota pre každú akciu. V druhom prípade na vstupnej vrstve sieť dostane iba stav a na výstupnej nám vráti odhady Q hodnôt pre všetky akcie, čiže

jedným spustením siete dostaneme hneď všetky potrebné Q hodnoty. Keďže druhý spôsob je postačujúci pretože potrebujeme získat iba argument maxima z Q hodôt a aj omnoho efektívnejší. V experimentoch zvolíme tento typ architektúry pre náš model.



Obr. 3.2: Model neurónovej siete Q funkcie

Obrázok 3.2 nám ponúka ilustratívny náhľad zakomponovania modelu Q siete do experimentu. Aby sme získaný *stav* zo simulátora prostredia mohli aplikovať na vstup siete, musíme ho najprv spracovať. Reprezentáciou *stavu* vektorom zaručíme správne fungovanie siete. To dosiahneme tým, že daný *stav* zakódujeme pomocou algoritmu *one-hot encoding*, ktorý pomocou 1 a 0 zabezpečí unikátnosť medzi všetkými atribútmi. Je to dôležité preto, aby sieť nehľadala nechcené súvislosti medzi atribútmi, ktoré nemajú medzi sebou nič spoločné.

Zvolili sme plne prepojený dopredný model siete s jednou skrytou vrstvou. Počet neurónov na vstupnej vrstve je rovný N, čo je veľkosť stavového priestoru simulátora prostredia. S určením počtu neurónov v skrytej vrstve je potrebné experimentovať. Počet označujeme Ma písmenom B sme označili *prah* siete. Na výstupnej vrstve je počet neurónov rovný štyrom, čo je počet možných akcií agenta: 0. neurón pre pohyb vľavo, 1. dole, 2. doprava a 3. pre smer nahor. Aktivačná funkcia na výstupe je lineárna, pretože chceme, aby nám sieť odhadovala Q hodnoty, ako numerické ohodnotenia akcií

pre daný stav.

V prípade nášho simulátora prostredia je reprezentácia stavu pomerne jednoduchá, keďže jediný meniaci sa prvok je pozícia agenta. Preto stačí, ak nám prostredie vráti skalárnu hodnotu, ktorá vyjadruje pozíciu agenta od 0 do N, kde N je veľkosť mriežky. Vektor pre vstupnú vrstvu siete tvorí pole núl, v ktorom na danom indexe priradíme 1, čo je pozícia agenta v bludisku.

Vo všeobecnosti na to aké parametre siete zvoliť neexistujú pevné pravidlá, prevažne iba empirické zistenia vhodných parametrov. Pri ich voľbe sa inšpirujeme (Zafrany, 2019) pri procese hľadania hyperparametrov. Prvým krokom bolo teda experimentovať s deterministickým bludiskom veľkosti 5×5 pri hľadaní takej siete, aby pre každú metódu explorácií dokázala konvergovať Q hodnoty do optimálnych hodnôt. Nasledujúca tabuľka ukázuje dané hyperparametre modelu:

Tabuľka 3.1: Detaily hyperparametrov Q-siete

Inicializácia váh	Lecun normal
Aktivačná funkcia na skrytej vrstve	ReLU
Aktivačná funkcia na výstupnej vrstve	Lineárna
Chybová funkcia	MSE
Optimalizátor	Adam
Rýchlosť učenia	0.001

Tréning Q-siete z hľadiska množstva dát na ktoré môžeme aplikovať gradient, členíme na *batch*, *mini-batch* a *stochastic gradient descent (SGD)*. *Batch gradient descent* je ale neprípustný kvôli on-line povahe učenia. *Mini-batch* je proti SGD stabilnejší a pri zložitejších problémoch sa zvykne uprednostniť (LeCun, Bottou, Orr, & Müller, 2012). V experimente sme zvolili alternatívu tréningu pomocou SGD, pretože mini-batch je z výpočtového hľadiska efektívnejší ak máme dostupný hardvér optimalizovaný pre operácie s maticami. Ukážeme si však, ako principiálne tréning pomocou *mini-batch* vyzerá. Mini-batch podľa (Mnih et al., 2013) zakomponujeme do algoritmu 3.1. Inicializuj dátovú štruktúru front D istej dĺžky a v cykle pre každý krok opakujeme:

1. V stave s_t získame a_t, s_{t+1} a r_{t+1} a uložíme ich do D ako štvoricu $(s_t, a_t, s_{t+1}, r_{t+1})$,
2. opakujeme až kým nezaplníme D,
3. ak je D plný, vyberieme náhodnú vzorku n prvkov,

4. iterujeme vybranú vzorku a vytvoríme cieľovú maticu **y** pomocou učiaceho pravidlo uvedeného v algoritme 3.1 a vstupnú maticu **x** obsahujúci iba stavy s,
5. použi matice **x** a **y** pre mini-batch tréning Q-siete,
6. opakujeme (okrem bodu 2) a udržujeme veľkosť D, takže pridaním nového prvku do D musíme posledný vymazať (štruktúra front nám to robí automaticky).

Chybová funkcia *MSE* je definovaná ako $\frac{1}{n}\sum_i^n E_i$, pretože chceme minimalizovať chybu predikcie Q hodnôt, kde E_i je i-tá kvadratická chyba. Pri snahe optimalizovať tréning sme vyskúšali metódy SGD, RMSProp a Adam. Spomedzi aktivačných funkcií *tanh* a *sigmoid* sme zvolili *ReLU* definovanú ako $f(x) = max(0, x)$, kde pre x väčšie ako 0 sa správa ako lineárna funkcia (Ruder, 2016).

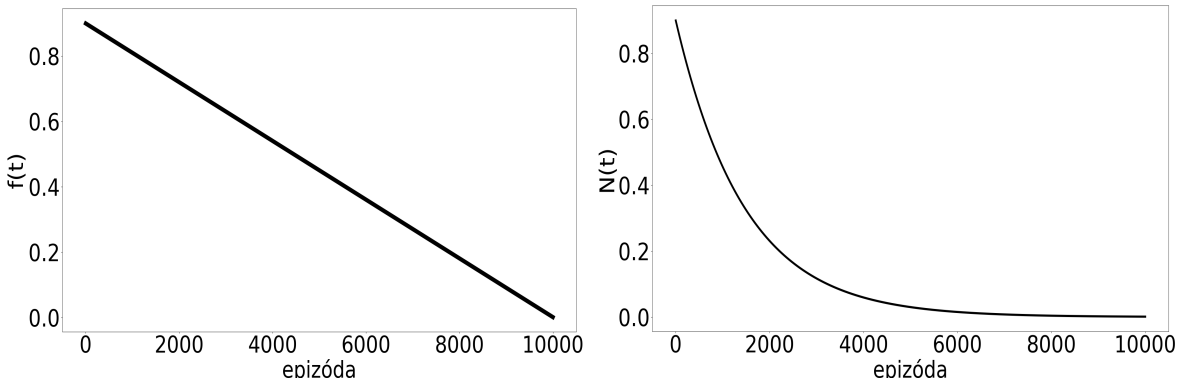
3.4 Explorácie v experimente

Keď máme opísaný simulátor prostredia, prejdime ďalej k exploračných metódam. Pre potreby experimentov si sich opíšeme z praktickejšieho hľadiska. Ako sme už párkrát spomínali, explorácie popisujú správanie agenta, čiže spôsob ako si agent vyberá akcie. Keď sa pozrieme na algoritmus 3.1, tak tieto metódy realizujú zvýraznenú časť - komponent zodpovedný za výber akcie prostredníctvom implementácie behaviorálnej stratégie $\pi(s)$.

3.4.1 Stochastické explorácie

ε-chamtivá Ako prvú metódu pre behaviorálnu stratégiu implementujeme exploráciu ε-chamtivú, ktorá je najpriamočiarejšia. Ako už sme spomínali v časti 1.5.1, v každom kroku agenta vyberieme náhodnú akciu z rovnomerného rozdelenia s pravdepodobnosťou ε. Inak vyberieme argument maxima z vektora Q hodnôt. Pri tejto stratégií sa zvykne ε postupom času znižovať, čím sa zvyšuje exploitácia prostredia agentom. Čiže čím agent pozná prostredia viac, tým sa spolieha na svoju vybudovanú znalosť o prostredí.

V experimente si ukážeme lineárny a exponenciálny pokles epsilonu ε, ktoré sú priblížené na nasledujúcich obrázkoch:



(a) Lineárna funkcia (b) Exponenciálna funkcia

Obr. 3.3: Typy funkčných poklesov epsilonu

Pri lineárnom poklese (a) hodnotu epsilonu v t epizóde získame vzťahom:

$$\varepsilon_t = \varepsilon_0 - k.t, \tag{3.1}$$

kde ε_0 je počiatočná hodnota epsilonu a k je konštanta poklesu vyrátaná ako podiel ε_0 s celkovým počtom epizód.

Pri exponenciálnom poklese (b) hodnotu epsilonu vyrátame podľa vzťahu:

$$N(t) = N_0.e^{-k.t}, \tag{3.2}$$

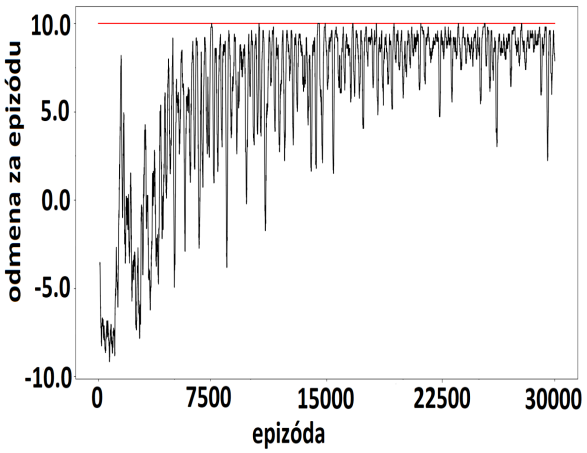
kde N_0 je počet počiatočná hodnota epsilonu a k je konštanta poklesu.

Boltzmannova metóda. Táto stratégia založená na Boltzmannovej distribúcii je oproti predošlej explorácii v nevýhode z hľadiska nastavenia hyperparametra. Hyperparameter epsilon pri ε-chamtivej explorácií je jednoduchšie nastaviť oproti Boltzmannovej metóde, pretože potrebujeme špecifickú hodnotu *teploty*, ktorá sa pre rôzne prostredia môže výrazne líšiť. Pre zvolenie hodnoty pri prvotnom spustení experimentu sa inšpirujeme prácou (Tijsma et al., 2016) a nastavíme pre dolné a horné ohraničenie *teplôt* hodnotami 0.1 a 6. Na základe výsledkov, ktoré získame pre tieto hodnoty upravíme *teplotu* smerom k optimálnej hodnote. Predpokladáme, že so zvyšovaním zložitosti prostredia sa môže zvýšiť optimálna hodnota *teploty*. Pri tejto metóde nebudeme znižovať hodnotu *teploty* počas tréningu, ako sme to robili s *epsilonom* pri ε-chamtivej explorácii.

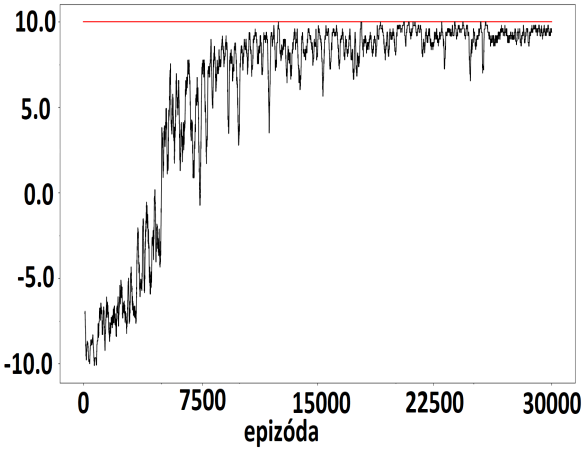
Aproximovaný pursuit Pursuit má iteratívnu vlastnosť, pretože podobne ako jednoduchá verzia algoritmu Q-learning, uchováva všetky páry stav–akcia v tabuľke. V každom kroku agenta upravíme súčasný pár stav–akcia pomocou predchádzajúceho páru podľa vzťahov 1.24. Od tabuľky upustíme aj v tomto prípade a nahradíme ju opäť neurónovou sieťou. Architektúru siete sme zvolili rovnakú ako v prípade Q-siete iba s tým rozdielom, že na výstupnej vrstve sme lineárnu aktiváciu nahradili funkciou softmax. To nám zaručí, že suma hodnôt na výstupe bude rovná jednej, čo je želaný stav, keďže chceme upravovať pradepodobnosti akcií podľa už vyššie spomínaných rovníc tejto metódy. V krátkosti si opíšme, ako takáto verzia behaviorálnej stratégie pomocou neurónovej siete vyzerá:

1. Odhadni Q hodnoty pre daný stav s pustením predikcie Q-siete a získaj argument maxima,
2. odhadni pravdepodobnosti akcií pre s pustením predikcie Pursuit siete,
3. uprav pravdepodobnosti z kroku 2 pomocou rovníc 1.24,
4. vyber argmax predstavujúci akciu z distribúcie z kroku 3,
5. uprav parametre Pursuit siete podľa upravených vstupných hodnôt z kroku 3.

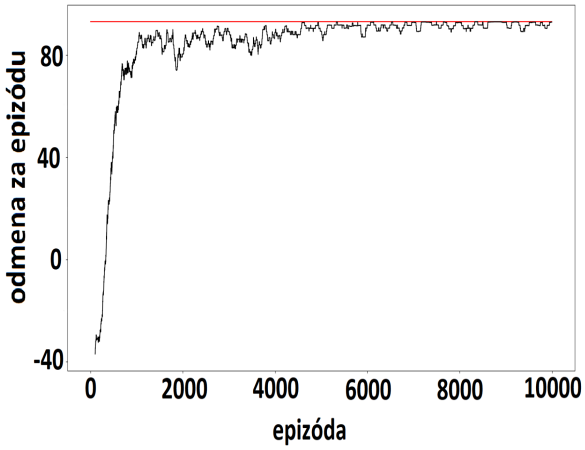
Pri nedostatočne vysokej odmene pre políčko cieľa (G), mala Pursuit sieť problém konvergovať ako to je zobrazené na obrázku nižšie. To bol dôvod prečo sme pri úspešnom dokončení úlohy agenta odmenili podstatne viac oproti ostatným stavom bludiska. To však nebol jediný problém pri implementácií Pursuit siete, pretože aj pri jednoduchom deterministickom bludisku 5×5 vykazovala sieť vysokú varianciu v porovnaní s ostatnými metódami. Túto nestabilitu siete sme znížili nižšou rýchlosťou učenia.



(a) α: 0.001



(b) α: 0.0001



(c) α: 0.001

Obr. 3.4: Vplyv rýchlosti učenia α na vývoj odmeny v čase v modeloch pursuit.

Obrázky 3.4 ukazujú krivku učenia pre kumulatívnu získanú odmenu za epizódu, vyhladenú metódou *moving average* s veľkosťou okna 100. Horizontálna červená čiara znázorňuje optimálnu odmenu. Krivky učenia na obrazkoch (a) a (b) sú pre odmeny prostredia: 10, -10, -1 pre políčka G, H, B a dĺžku tréningu až 30 000 epizód. Krivka na obrázku (c) znázoňuje prostredie s finálnou odmenou zvolenou v podkapitole 3.2.1. Dĺžku tréningu v tomto prípade stačilo zvoliť 3-krát kratšiu a vidíme, že Q sieť začala konvergovať približne už pri 4000 epizódach. To bol aj dôvod prečo sme zvolili odmeňovaciu funckiu $\mathcal{T}$ s výrazne vyššou hodnotou pre políčko (G). Pursuit sieť dokázala pomerne stabilne konvergovať ak hodnoty rýchlosti učenia α boli z množiny $\{1e-04, 1e-05, 1e-06\}$. Všetky krivky sú pre vykreslené pre deterministické prostredie veľkosti 5×5.

3.4.2 IM explorácie

Pri metódach explorácií pomocou IM sme v experimentoch implementovali obidva spôsoby, ktoré sme si objasnili v podkapitole 1.5.2. Prvým spôsobom je explorácia založená na manipulácií odmeňovacej funkcie $\mathcal{T}$ podľa rovnice 1.25. Preto potrebujeme vyrátať *exploračný bonus*, ktorý zakomponujeme priamo do cieľového vektora pre neterminálny stav určeného pre výpočet gradientu nasledovne:

$$y_t = r_t + \underbrace{\frac{1}{C\sqrt{n(s_t)}}}_{\text{exploračný bonus}} + \gamma \max_{a'} Q(a', s_{t+1}; \theta), \tag{3.3}$$

kde C je škálovací faktor, $n(s_t)$ určuje počet návštev stavu s a člen r_t je získaná odmena z prostrediav v čase t. Z rovnice 3.3 je zrejmé, že čím je hodnota $n(s_t)$ väčšia, tak tým je prírastok k odmene - *exploračný bonus* - menší. Takouto úpravou odmeny teda zabezpečíme aby agent bol motivovaný explorovať menej často navštívené stavy. Hodnoty odhadov pre akcie posledného členu sú získané Q-sieťou. Takto modifikované odmeny sa premietnu do predikcie Q hodnôt siete, preto musíme ešte vyriešiť, akým spôsobom uskutočníme výber akcie. Uvažujeme nad chamtivou a softmax exploráciou.

Chamtivá explorácia vyberie stále tú akciu, ktorá ma priradenú najvyššiu Q hodnotu. Ide o zjednodušenú verziu ε-chamtivej explorácie, kedy nastavíme ε na hodnotu 0. Explorácia je teda silným spôsobom ovplyňovaná *exploračným bonusom.*

Softmaxová explorácia je podobná metóde Boltzmannovej metóde, avšak s tým rozdielom, že hodnota teploty je stále rovná 1. To má za následok, že pri vytváraní pravdepodobnostnej distribúcie z predikovaných Q hodnôt sa zachová pomer pravdepodobností ku daným Q hodnotám. To je rozdiel oproti skutočnej Bolzmannovej explorácii, kde s vyššou hodnotou teploty sa rozdiely Q hodnôt znižovali a s nižšou teplotou sa rozdiely zvyšovali.

V experimente sme zvolili Softmax exploráciu, pretože chamtivá stratégia sa ukázala pri simuláciách ako nepostačujúca.

Pri druhom spôsobe implementujeme metódu UCB-1, kde pomocou nej modifikujeme behaviorálnu stratégiu. Rátame počet výskytov výberu akcie pre každý stav a

na základe vzťahu 1.26 vyrátame *exploračný bonus*. Ten pripočítame ku Q odhadom akcií získaných Q-sieťou a vyberieme z nich argmax.

3.5 Zostavenie experimentov

V závere tejto kapitoly, keď už poznáme všetky potrebné techniky použité v experimente, môžeme si nadizajnovať jeho priebeh a spôsob evaluácie výsledkov experimentov.

Tabuľka 3.2: Prehľadávaná množina hyperparametrov H pre jednotlivé explorácie.

	ε-chamtivá	Boltzmann	Pursuit		N(s)	UCB-1
	ϵ	T	β	α	β	C
low	0.1	0.1	1e-05	1e-03	0.1	0.1
high	0.9	6	1	1e-06	10	10

Tabuľka 3.2 nám ukazuje dolné (low) a horné (high) ohraničenie prehľadávanej množiny hyperparametrov jednotlivých explorácií. V prípade ε-chamtivej explorácie sme množinu hodnôt hyperparametrov prehľadávali s lineárnym poklesom ε a exponenciálnym, ako sme si uviedli v podkapitole 3.3. Hyperparametre Pursuit a ε-chamtivej explorácie sú viazané na pravdepodobnosti, takže ich ohraničenie je od 0 do 1. Pri zvyšných exploráciách to neplatí a museli sme vhodné ohraničenie empiricky simuláciami odhadnúť. Dolná hranica hyperparametrov explorácií okrem Pursuit metódy, znamená najvyššiu mieru exploitácie a horná najvyššiu mieru explorácie. Pri Pursuit metóde je to naopak. Zadefinujme si teraz terminológiu, ktorou vyjadríme jednotlivé etapy experimentu:

- **Epocha** - V kontexte tréningu neurónovej siete to znamená vykonať učenie váh vzhľadom nad všetkými vzorkami vstupných dát.
- **Epizóda** - Jedna epizóda je rovná dĺžke jednej hry, ktorá trvá maximálne K krokov. Agent je v hre, kým nevstúpi na terminálne políčko - G, H - alebo kým neurobí K krokov. Je to hyperparameter, ktorý má zabrániť zaseknutiu agenta (čo môže zapríčiniť napríklad zlá inicializácia váh siete).
- **Beh** - Pod týmto pojmom máme na mysli absolvovanie tréningu agenta, čiže vykonanie t epizód.

- **Simulácia** - Simuláciou rozumieme pustenie *n beh*.
- **Prehľadávanie** - Pod pojmom *prehľadávanie* máme na mysli pustenie m simulácií pre všetky hodnoty hyperparametrov danej explorácie.
- **Experiment** - Vykonanie s *searchov* pre všetky exploračné metódy v rámci jedného typu prostredia.

Neurónové siete trénujeme väčšinou počas N epoch, keď N-krát prejdeme všetky vzorky dát a aplikujeme na ne metódu SGD. Pri RL algoritmoch však podstatu *epoch* napĺňajú *epizódy*, pretože nemá význam natrénovať sieť viackrát jednou vzorkou získanou interakciou agenta s prostredím. Počas jednej epizódy natrénujeme sieť K-krát. Model Q-siete, ktorý sme navrhli v podkapitole 3.1 ostane bez zmeny pre všetky simulácie rovnako aj *diskont* faktor, ktorý nastavíme na $\gamma = 0.98$. Pri všetkých simuláciách nemanipulujeme s pozíciou agenta ani s pozíciami iných políčok. Sumarizujme si parametre experimentu: Počet epizód pre bludisko veľkosti 5×5 a 8×8 sme stanovili na 10 000 a 15 000. Počet krokov približne dvakrát väčší ako je veľkosť danej mriežky, pričom pri stochastickej verzií je veľkosť približne o 20 krokov väčšia. Počet *simulácií* sme stanovili na 10 a počet *prehľadávaní* na 5, pretože robíme 5 explorácií. S daným nastavením pripravíme 4 experimenty pre jednotlivé typy prostredia.

3.5.1 Evaluácia explorácií

Pri skúmaní exploračných stratégií nás zaujíma ich správanie z hľadiska ich *asymptotickej výkonnosti*, *stability* a v neposlednom rade podľa ich *rýchlosti konvergencie*. Výsledky pre dané tri parametre evaluácie zobrazíme numerickou formou v tabuľkách. Okrem toho nás zaujíma aj vizualizácia priebehu tréningu vo forme *krivky učenia*. Získame ju spriemernením zaznamenanej kumulovanej odmeny za epizódu a rovnako spriemerujeme aj počet krokov agenta za epizódu vo fáze tréningu.

Zadefinujme si formálnym spôsobom dané spôsoby vyhodnotenia explorácií. Uvažujme teda maticu $\mathbf{M}$ výsledkov simuácií pre konkrétnu inštanciu metódy explorácie. Matica je rozmerov $S \times T$, kde $S = 10$ je počet simulácií a $T \in \{10000, 15000\}$ je daný počet epizód. Definujme si vektor priemeru a štandardnej odchýlky matice $\mathbf{M}$ podľa

j-teho stĺpca:

$$\begin{aligned} \mu_j &= \sum_{i=1}^{S} \frac{m_{ij}}{S} \\ \sigma_j &= \sum_{i=1}^{S} \sqrt{\frac{1}{S} \sum_{i=1}^{S} (m_{ij} - \mu_j)^2}, \end{aligned} \tag{3.4}$$

kde μ_j znamená priemernú hodnotu j-teho stĺpca matice $\mathbf{M}$ a podobne σ_j znamená štandardnú odchýlku.

Na základe vzťahov 3.4 vieme vygenerovať vektory, ktoré použijeme pre vykreslenie kriviek učenia. Pre priemer: $\{\mu_j\} \in (0, T)$ a štandardnú odchýlku: $\{\sigma_j\} \in (0, T)$, pričom obidva sú vektory dĺžky $1 \times T$. V prípade vizualizácie priemerného počtu krokov postupujeme analogicky.

Pre evaluáciu *asymptotickej výkonnosti* urobíme priemer z vektora priemerov $\mu_\mathbf{j}$, čím získame skalárnu hodnotu. Analogicky to zopakujeme aj pre štandardnú odchýlku, čím zase získame hodnotu, podľa ktorej môžeme porovnávať *stabilitu* explorácií. Pre potreby vyhodnotenia *rýchlosti konvergencie* potrebujeme určiť epizódu, od ktorej začala daná explorácia konvergovať. Zadefinujeme si najprv odhad hodnoty konvergencie a označme ho θ:

$$\theta = \frac{1}{T} \sum_{j=t_0}^{T} \mu_j, \tag{3.5}$$

kde T je počet epizód a t_0 označuje epizódy od ktorej rátame priemer, ktorého hodnota bude θ. Nájdenie odhadu začiatku konvergencie epizódy definujeme nasledovne:

$$\text{hľadáme také } j : \forall i \in S \; m_{ij} \geq \theta, \tag{3.6}$$

kde j predstavuje hľadanú epizódu pre odhad konvegencie.

V prípade bludiska sa nám ponúka vizualizácia pomocou tepelnej mapy (heatmap), vyrátanej z počtu návštev jednotlivých stavov. Čím je farba teplejšia, tým stav bol navštívený častejšie. Takáto tepelná mapa nám prezradí aké rôzne stratégie agent počas tréningu skúmal.

Kapitola 4

Výsledky

V záverečnej časti tejto práce si uvedieme výsledky experimentov jednotlivých explorácií. Ako sme už spomínali v predošlej kapitole, ukážeme si grafické a numerické vyhodnotenie explorácií. Výsledky kvôli ich množstvu prezentujeme iba pre ich optimálne hyperparametre.

Najskôr si ukážeme numerické vyjadrenie v dvoch tabuľkách. V prvej tabuľke si ukážeme dané optimálne hyperparametre metód a v druhej tabuľke k nim zodpovedajúce výsledky. V prípade evaluácie podľa *rýchlosti konvergencie* sme simuláciami zistili, že dosiahneme lepšiu presnosť ak vo výraze 3.6 k hranici θ pripočítame aj štandardnú odchýlku σ, vyrátanú taktiež pre interval $[t_0, T]$. Pri deterministickej verzii experimentov nám postačí pôvodný vzťah pre výpočet θ, pretože pri konvergencií krivka neosciluje a priemer θ z neho vypočítaný je maximálny možný, ktorý explorácia môže dosiahnuť.

V druhom kroku ich zoskupíme podľa experimentov, takže získame sadu štyroch výsledkov pre päť typov explorácií. Keďže výsledky pre deterministický typ prostredí a pre všetky typy explorácií dosahovali príliš dobré výsledky, tak vizualizácia kriviek učenia by nemala výpovednú hodnotu. Pre stochastický typ prostredí vizualizujeme krivky učenia vo forme priemernej kumulovanej odmeny za epizódu a priemerného počtu krokov za epizódu. Čím priebeh krivky pre odmenu rastie rýchlejšie, tým prudšie by mala krivka pre počet krokov klesať. Pre každý experiment si uvedieme tepelnú mapu návštev stavov počas tréningu pre optimálne hyperparametre explorácii.

Tabuľka 4.1 nám poskytuje optimálne hyperparametre explorácií pre dané experimenty, ktorých zostavenie sme si navrhli v predchádzajucej kapitole. Ako môžeme

vidieť, väčšina explorácií si vystačila s dolnými hodnotami, prípadne s jej blízkym okolím prehľadávanej množiny H.

Tabuľka 4.1: Optimálne hyperparametre explorácií pre deterministické i stochastické prostredia oboch veľkostí.

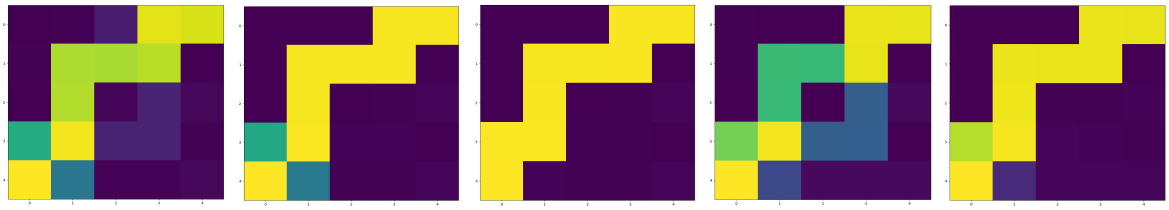
	ϵ-chamtivá		**Boltzmann**	**Pursuit**		**N(s)**	**UCB-1**
5x5-det	ϵ	*pokles*	T	β	α	C	C
	0.1	exp	0.2	1	1e-04	0.1	0.1
5x5-stoch	ϵ	*pokles*	T	β	α	C	C
	0.1	exp	0.3	0.1	1e-04	0.1	0.2
8x8-det	ϵ	*pokles*	T	β	α	C	C
	0.9	exp	0.8	0.1	1e-04	0.1	0.1
8x8-stoch	ϵ	*pokles*	T	β	α	C	C
	0.1	lin	1	0.1	1e-05	0.1	0.2

V tabuľke 4.2 máme zobrazené dosiahnuté výsledky pre každy experiment. Členy na pravej strane tabuľky vyjadrujú aké hodnoty dosiahli explorácie s danými hyperaprametrami v simuláciách. Hodnoty v tabuľke sú vyrátané z 10 simulácií. Člen μ predstavuje dosiahnutý priemer, σ štandardnú odchýlku, t epizódu v ktorej explorácia konvergovala a člen $rank(\mu)$ znamená usporiadanie explorácií podľa priemeru. Pre prostredia s 5×5 mriežkou celkový počet epizód tréningu bol 10 000 a pre 8×8 15 000.

Tabuľka 4.2: Priemerná suma odmeny pre optimálne hyperparametre explorácií pre deterministické i stochastické prostredia oboch veľkostí z Tab. 4.1.

	ϵ-chamtivá	**Boltzmann**	**Pursuit**	**N(s)**	**UCB1**	
5x5-det	896.6	919.9	921.7	**938.7**	914.2	μ
	62.6	5.5	5.5	**3.3**	4.3	σ
	553	114	89	**77**	143	t
	5	3	2	**1**	4	$rank(\mu)$
5x5-stoch	596.1	581.6	576.6	**615.0**	575.7	μ
	450.2	465.7	439.9	**456.3**	455.2	σ
	530	210	1308	**240**	390	t
	2	3	4	**1**	5	$rank(\mu)$
8x8-det	996.1	1258.8	1200.9	**1341.8**	1273.0	μ
	250.7	51.5	87.3	**21.9**	16.8	σ
	5524	923	1546	**96**	284	t
	5	3	4	**1**	2	$rank(\mu)$
8x8-stoch	716.0	843.4	646.1	**1034.6**	834.5	μ
	617.6	544.5	493.1	**383.5**	548.5	σ
	3736	2127	6434	**416**	904	t
	4	2	5	**1**	3	$rank(\mu)$

4.1 Deterministické prostredie 5×5



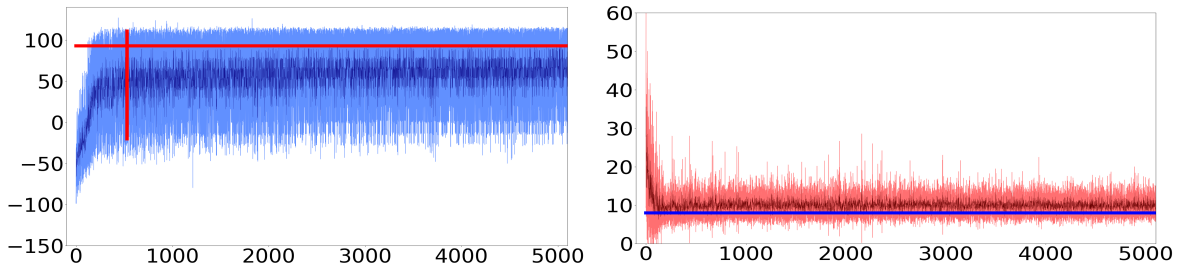
(a) ε-chamtivá (b) boltzmann (c) pursuit (d) N(s) (e) ucb-1

Obr. 4.1: Tepelná mapa návštev stavov.

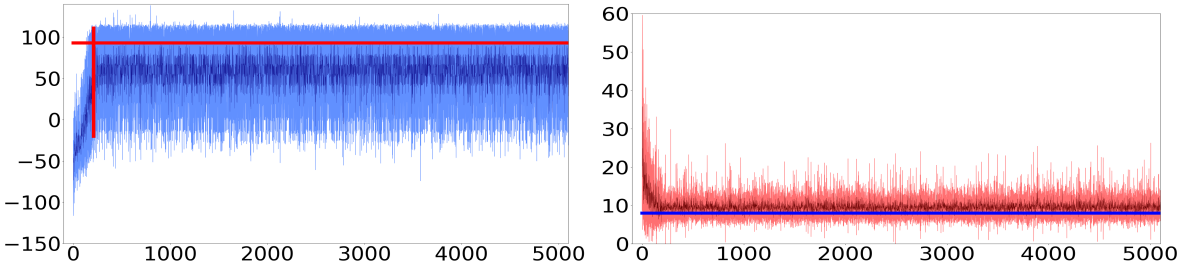
Pri tejto najjednoduchšej verzii experimentu všetky explorácie zvolili tú hodnotu optimálnych hyperparametrov, pri ktorých prostredie s najväčšou mierou exploitovali. Prostredie bolo deterministické s malou zložitosťou, takže všetky metódy ho dokázali veľmi rýchlo preskúmať. Hodnoty metrík jednotlivých explorácií nadobudli podobné hodnoty až na ε-chamtivú exploráciu, ktorá od iných metód dosiahla výrazne horšie výsledky. Možno by výsledky dopadli lepšie, ak by sme experimentovali s menšími hodnotami ε. Spomedzi všetkých metrík obstála najlepšie count-based explorácia $N(s)$. Pri ako

jedinej explorácii, agent počas tréningu zvažoval aj dolnú cestu do cieľa.

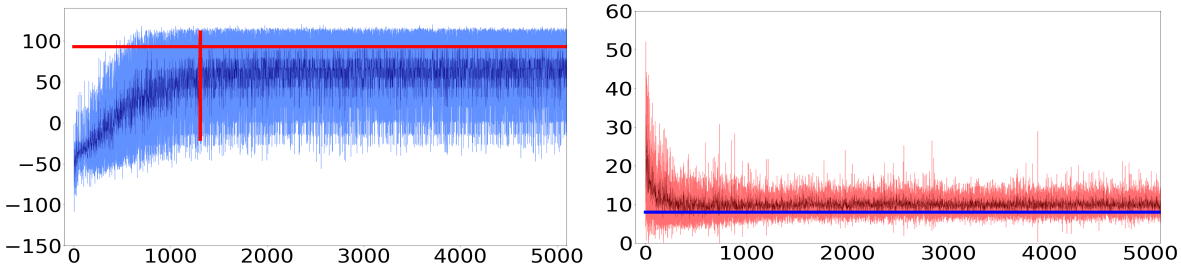
4.2 Stochastické prostredie 5×5



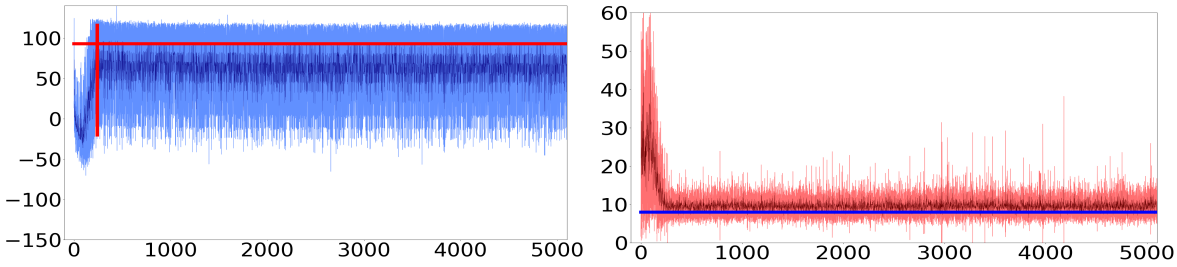
Obr. 4.2: Vývoj explorácie ε-chamtivá ε=0.1 s exponenciálnym poklesom.



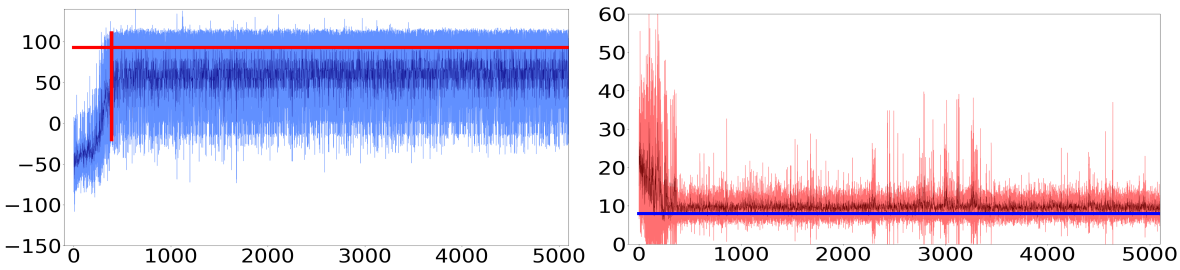
Obr. 4.3: Vývoj explorácie boltzmann T=0.3.



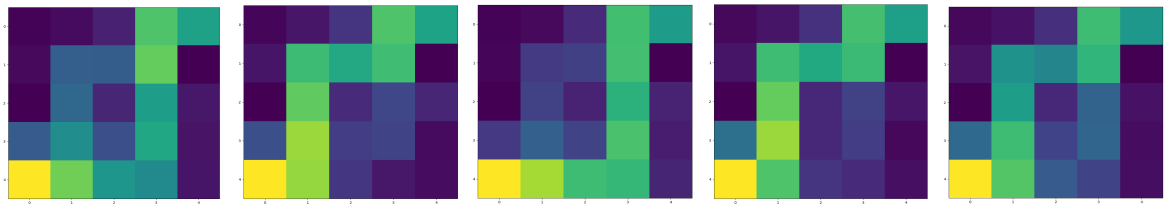
Obr. 4.4: Vývoj explorácie pursuit α=1e-04 β=0.1.



Obr. 4.5: Vývoj explorácie count-based $N(s)$ C=0.1.



Obr. 4.6: Vývoj explorácie UCB-1 C=0.2.



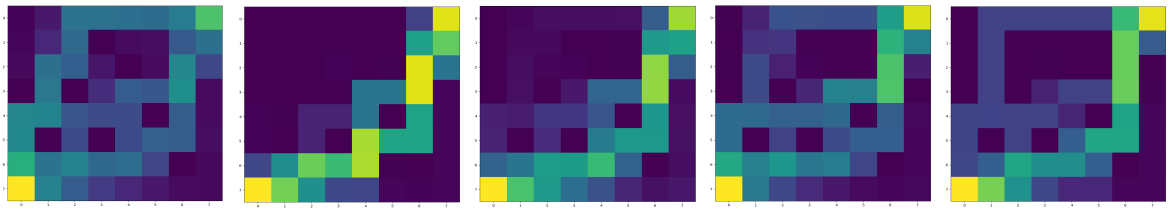
(a) ε-chamtivá (b) boltzmann (c) pursuit (d) $N(s)$ (e) ucb-1

Obr. 4.7: Tepelná mapa návštev stavov.

Pri stochastickom type prostredia máme k dispozícii aj krivky učenia, pričom tá vľavo vizualizuje $\{\mu_j\} \in (0, T)$. V grafe napravo napevno zobrazujeme jej priebeh iba pre interval $t \in [-100, 5100]$ pre mriežku 5×5 a $t \in [-100, 8000]$ pre mriežku 8×8, pretože pre väčšie t sa priebeh krivky už nemení. Horizontálna čiara v grafe ukazuje maximálnu možnú odmenu, ktorú agent môže získať a vertikálna čiara zobrazuje epizódu konvergencie vyrátanej pomocou vzťahu 3.6. Graf napravo vyjadruje celkový počet krokov za epizódu a je vyrátaná analogicky ako predchádzajúca. Modrá horizontálna čiara pri tomto type znamená minimálny počet krokov do cieľa. Bludisko s veľkosťou mriežky 5×5 má minimálny počet krokov 8 a pri mriežke 8×8 ich je 14.

Pri stochastickej verzii bludiska 5×5 zvíťazila opäť IM explorácia $N(s)$, avšak v ostatných metrikách najlepšia nebola. UCB-1 explorácia skončila až na poslenom mieste. Pursuit dosiahla najlepšiu stabilitu ale za cenu najpomalšej konvergencie, spomedzi ostatných explorácií. Môžeme si všimnúť, že ϵ-chamtivá explorácia s rovnakými hodnotami hyperparametrov ako pri predchádzajúcom experimente dosiahla až druhú priečku. Pri tejto povahe bludiska v porovnaní s deterministickým, vidíme aj na tepelných mapách výraznejší vplyv explorácie zapríčinený stochasticitou prostredia.

4.3 Deterministické prostredie 8×8

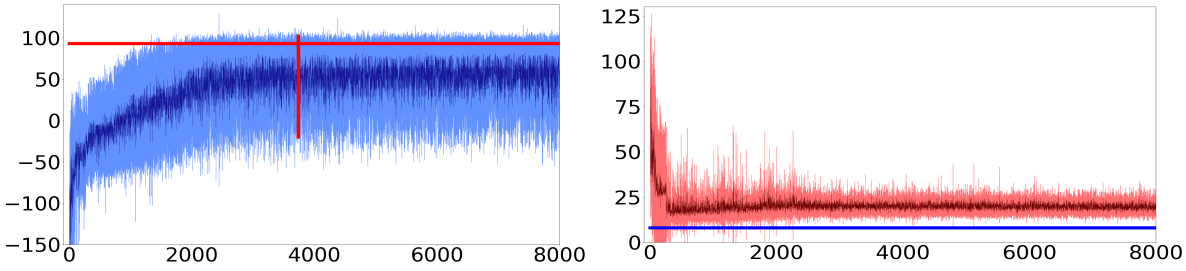


(a) ε-chamtivá (b) boltzmann (c) pursuit (d) $N(s)$ (e) ucb-1

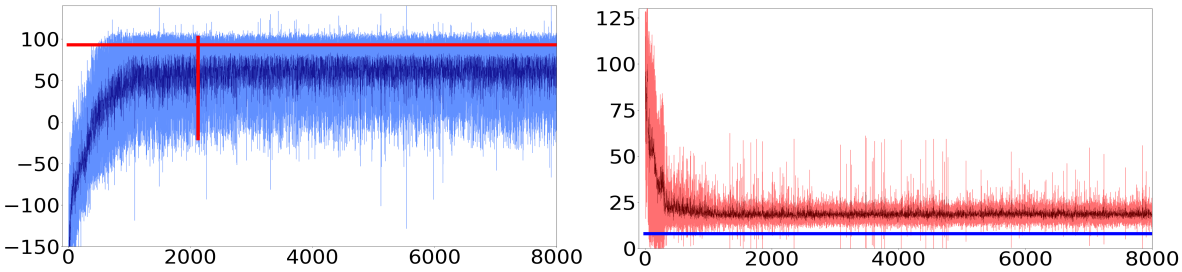
Obr. 4.8: Tepelná mapa návštev stavov.

Pri deterministickom prostredí veľkosti 8×8 explorácia UCB-1 dosiahla druhé umiestnenie. Zistili sme, že pri boltzmannovej metóde je hodnota hyperparametra *teploty* výrazne vyššia, až $T = 0.8$ oproti doterajším nízkym hodnotám 0.2 a 0.3. Explorácia ϵ-chamtivá dosiahla posledné umiestnenie a v porovnaní s ostatnými metódami mala veľmi nízku stabilitu. Môžeme to vidieť taktiež na tepelnej mape, kde ϵ-chamtivá výraznejšie neexploitoval prostredie. Explorácia $N(s)$ dosiahla najlepší asymptotický výkon a aj najrýchlejšie konvergovala.

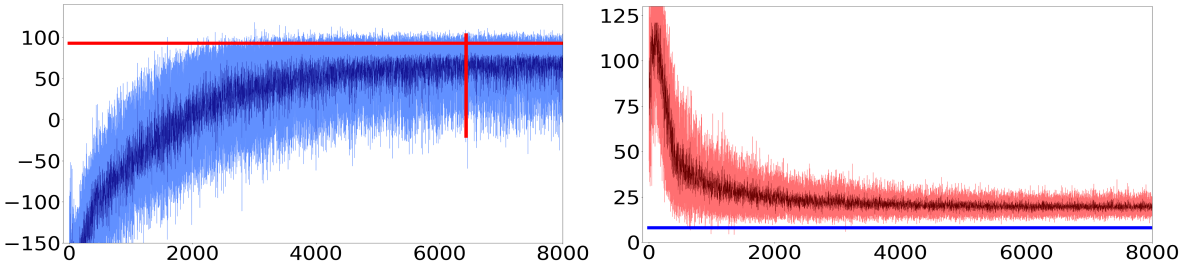
4.4 Stochastické prostredie 8×8



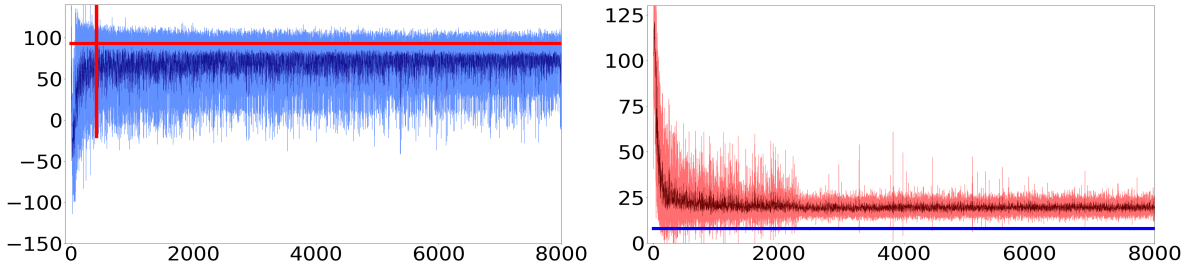
Obr. 4.9: Vývoj explorácie ε-chamtivá ε=0.9 s lineárnym poklesom.



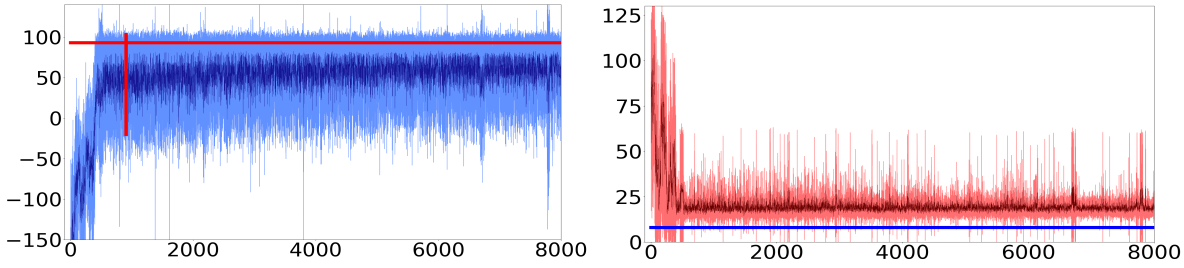
Obr. 4.10: Vývoj explorácie boltzmann T=1.0.



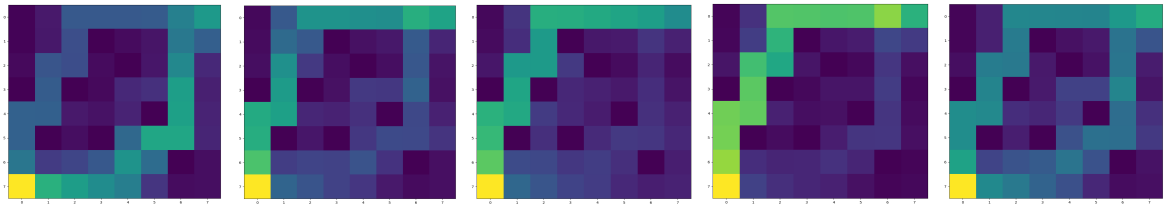
Obr. 4.11: Vývoj explorácie pursuit α=1e-05 β=0.1.



Obr. 4.12: Vývoj explorácie count-based $N(s)$ C=0.1.



Obr. 4.13: Vývoj explorácie UCB-1 C=0.2.



(a) ε-chamtivá (b) boltzmann (c) pursuit (d) $N(s)$ (e) ucb-1

Obr. 4.14: Tepelná mapa návštev stavov.

V prípade posledného experimentu dosiahla najlepšie výsledky opäť $N(s)$ a to spomedzi všetkých metrík. Ostatné explorácie dosiahli veľmi podobné výsledky. V prípade Boltzmannovej metódy explorácia dosiahla najlepšie hodnoty pre softmaxovú distribúciu, pretože hyperpatameter mal hodnotu 1. Výrazná zmena taktiež nastala pri ϵ-chamtivej stratégii, pretože pri tomto experimente epsilon klesal lineárne. Ako môžeme

vidieť na obrázku vľavo explorácia Pursuit konvergovala s podstatným oneskorením ako ostatnéexplorácie, avšak priebeh učenia mala hneď po $N(s)$ explorácií najstabilnejší. To sa rovnako týka aj grafu na pravej strane, krivky učenia pre počet krokov mala Pursuit s najmenšou odchýlkou. UCB-1 explorácia zase preukazovala nan grafe vpravo priebeh s najvyššiou odchýlkou. Je zaujímavé si všimnúť, že pri stochastickom bludisku sa Q-sieť v rámci všetkých explorácií naučila optimálnu stratégiu tej bezpečnejšej cesty (pravdepodobne kvôli políčkam priepasti (H)).

Záver

V práci sme sa venovali problematike exploračných stratégii v učení posilňovaním. Najskôr sme si uviedli všetky potrebné teoretické poznatky a princípy fungovania jednotlivých metód a ich vybraných algoritmov. Uviedli sme si taktiež ako zapadá do konceptu RL optimalizácia pomocou gradientových metód a na záver sme si ozrejmili akú rolu zohrávajú samotné exploračné stratégie a priblížli sme si ich niektoré typy. Poznatky sme využili pri zostrojení experimentou, ktorých cieľom bolo vyšetriť priebeh explorácií pri probléme hľadania najkratšej cesty v prostredí bludiska s dvoma úrovňami zložitosti. V rámci poslednej kapitoly venovanej prezentácii výsledkov experimentov, ponúkame viacero kvantitatívnych mier, pomocou ktorých vyhodnocujeme explorácie.

Ukázalo sa, že pomer explorácie a exploitácie je skutočne kľúčový pre efektívne plnenie úlohy agenta. Už aj pri nami skúmanom jednoduchom probléme hľadania najkratšej cesty v bludisku, potrebovali explorácie väčšinou špecifické hodnoty hyperparametrov zabezpečujúcich mieru explorácie, inak boli neefektívne.

Výsledky experimentov ukázali, že IM explorácia založená na jednoduchom počítaní stavov dávala lepšie výsledky ako ostatné explorácie a to vo všetkých typoch experimentov. Preukázala prevahu v stabilite aj v rýchlosti konvergencie. Dosiahla to v každom prípade s najnižšou hodnotou hyperparametra spomedzi nami skúmanej množiny hodnôt. UCB-1 však dopadlo výrazne horšie ako predchádzajúca metóda IM explorácie. Vo všetkých experimentoch si explorácie vystačili s hodnotou hyperparametra, ktorá zabezpečuje len malú mieru explorácie, avšak dostatočne postačujúcu. V prípade robustnosti explorácií najlepšie obstála opäť explorácia count-based $N(s)$, pretože jej hyperparameter škálovací faktor disponuje širokým spektrom hodnôt, ktoré dokázali efektívne nájsť optimálne stratégie. Robustnosť UCB-1, ktorá podobne škáluje pomocou hyperparametra vygenerovanú vnútornú odmenu, však obstála horšie

ako metóda $N(s)$. Explorácia pursuit, keďže bola aproximovaná neurónovou sieťou, jej nastavovanie hyperparametrov bolo najťažšie a najzdĺhavejšie. Explorácia ε-chamtivá je pomerne robustná, avšak kôli svojmu naivnému prístupu v prípade vyberania akcií z uniformného rozdelenia, môže opakovať tie isté chyby mnohokrát, v čom spočíva jej najväčšia nevýhoda. Boltzmannová metóda dokáže byť pomerne efektívnou v prípade nájdenia optimálnej hodnoty (alebo jej okolia) hyperparametra *teploty*.

V prípade budúceho výskumu problematiky sa ponúka široké spektrum možností smerovania. Záujem by sme však smerovali k výskumu metód explorácie pomocou vnútornej motivácie v spolupráci s generatívnymi modelmi. V oboch prípadoch vidíme potenciál v ich realizácií pomocou rôznych modelov neurónových sietí.

Literatúra

Baldi, P., & Hornik, K. (1989, 12). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, *2*, 53-58.

Bellemare, Srinivasan, Ostrovski, Schaul, Saxton, & Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. *CoRR*, *abs/1606.01868*. Retrieved from `http://arxiv.org/abs/1606.01868`

Chentanez, Barto, & Singh. (2005). Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems 17* (pp. 1281–1288). MIT Press.

Coggan, M. (2004). Exploration and exploitation in reinforcement learning..

Haykin, S. S. (2009). *Neural networks and learning machines* (3rd ed.). Upper Saddle River, NJ: Pearson Education.

Hester, T., Lopes, M., & Stone, P. (2013). Learning exploration strategies in model-based reinforcement learning. In *The twelfth international conference on autonomous agents and multiagent systems (aamas).*

LeCun, Bottou, L., Orr, G., & Müller, K. (2012). Efficient backprop. In *Neural networks: Tricks of the trade* (pp. 9–48).

McFarlane, R. (2003). A survey of exploration strategies in reinforcement learning..

Mnih, Kavukcuoglu, Silver, Graves, Antonoglou, Wierstra, & Riedmiller. (2013). Playing atari with deep reinforcement learning. *CoRR*, *abs/1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Hassabis, D. (2015, February). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Nielsen, M. (n.d.). *Neural networks and deep learning.* `http://neuralnetworksanddeeplearning.com`.

Ostrovski, Bellemare, Oord, & Munos. (2017). Count-based exploration with neural

density models. *CoRR*, *abs/1703.01310*. Retrieved from http://arxiv.org/abs/1703.01310

Pecháč, M. (2019). *Reinforcement learning for intrinsically motivated robot behavior.* Pisomná práca k dizertačnej skúške. FMFI Univerzita Komenského v Bratislave.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *CoRR*, *abs/1609.04747*. Retrieved from http://arxiv.org/abs/1609.04747

Semmler, M. (2017). *Exploration in deep reinforcement learning* (Unpublished master's thesis). Technische universitat Darmstadt, Germany.

Sutton, R., & Barto, A. (2017). *Reinforcement learning: An introduction* (2nd ed.) (No. zv. 1). A Bradford Book.

Tang, Houthooft, Foote, Stooke, Chen, Duan, ... Abbeel (2016). Exploration: A study of count-based exploration for deep reinforcement learning. *CoRR*, *abs/1611.04717*. Retrieved from http://arxiv.org/abs/1611.04717

Tijsma, A. D., Drugan, M. M., & Wiering, M. A. (2016). Comparing exploration strategies for Q-learning in random stochastic mazes. In *Ssci* (p. 1-8). IEEE.

van den Oord, A., Kalchbrenner, N., & Kavukcuoglu, K. (2016). Pixel recurrent neural networks. *CoRR*, *abs/1601.06759*. Retrieved from http://arxiv.org/abs/1601.06759

van Hasselt, H. (2012). Reinforcement learning in continuous state and action spaces. In (pp. 207–251). Springer Berlin Heidelberg.

Zafrany, S. (2019). *Deep reinforcement learning for maze solving.* https://www.samyzaf.com/ML/rl/qmaze.html.