

ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA RIADENIA A INFORMATIKY

DIZERTAČNÁ PRÁCA

Študijný odbor: **Aplikovaná informatika**

Ing. Michal Chovanec

**Aproximácia funkcie ohodnotení v
algoritmoch Q-learning
neurónovou sieťou**

Vedúci: **prof. Ing. Juraj Miček, PhD**

Apríl 2016

Abstrakt

MICHAL CHOVANEC: *Aproximácia funkcie ohodnotení v algoritmoch Q-learning neurónovou sieťou* [Dizertačná práca]

Žilinská Univerzita v Žiline, Fakulta riadenia a informatiky, Katedra technickej kybernetiky.

Vedúci: prof. Ing. Juraj Miček, PhD

FRI ŽU v Žiline, 2016

Práca sa zaoberá aproximáciou funkcie ohodnotení konania agenta, v algoritmoch Q-learning. V priestoroch s malým počtom stavov predstavuje vhodné riešenie tabuľka. Pre prípady veľkého počtu stavov je tabuľkové riešenie ťažko vypočítateľné. Je tak nutné použiť aproximáciu. Vhodným kandidátom je neurónová sieť. Tradičné riešenie doprednej siete je však nepoužiteľné z dôvodov nemožnosti takúto sieť učiť. V práci je preto venovaný priestor neurónovej sieti základných funkcií ktorú už je možné na daný problém trénovať iteráčnými metódami.

Abstract

MICHAL CHOVANEC: *Q-function approximation in Q-learning algorithms using neural network*

[Dissertation thesis]

University of Žilina, Faculty of Management Science and Informatics, Department of technical cybernetics.

Tutor: prof. Ing. Juraj Miček, PhD

FRI ŽU v Žiline, 2016

This thesis is focused on Q-value function approximation, in Q-learning algorithms. In state spaces with small numbers of states can be solution using table used well. In cases with large numbers of states is table solution difficult to solve. It is necessary to use approximation, where good candidate can be neural network. Common solution, using feed forward neural network can't be used, because impossible to learn this network. Thesis is focused to use basis functions neural network, which can be learned using iterations methods.

Prehlásenie

Prehlasujem, že som túto prácu napísal samostatne a že som uviedol väčšinu použitých prameňov a literatúry, z ktorých som čerpal.

V Žiline, dňa 22.4.2016

Michal Chovanec

Obsah

1	Ciele práce	4
2	Teoretické východiská učiacich sa systémov na báze odmeňovania	7
2.1	Úvod	7
2.2	Markovove rozhodovacie procesy	10
2.3	Princípy učenia s odmeňovaním	12
2.4	Ohodnocovanie vykonaných akcií	14
2.5	Jednostavový systém	17
2.5.1	Výsledky experimentu	18
3	Q-larning algoritmus	27
3.1	Definícia algoritmu	27
3.2	Výber akcie	31
3.3	Problémy výpočtu $Q(s,a)$	32
3.4	Tabuľka	34
3.5	Dopredná neurónová sieť	34
3.6	Kohonenova neurónová sieť	37
3.7	Neurónová sieť bázičických funkcií	40
3.7.1	Určenie parametrov α	42
3.7.2	Určenie parametrov β	43
3.7.3	Určenie váhových parametrov w	44
3.7.4	Hybridný variant	44

	3
4 Experimentálna časť	47
4.1 Ciele experimentu	47
4.1.1 Divergencia riešenia	48
4.2 Riešenie aproximácie	52
4.3 Návrh experimentu	52
4.4 Výsledky experimentu	56
4.5 Priemerné výsledky experimentu	62
4.6 Doplnkový experiment	64
5 Záver	70
Literatúra	72

Kapitola 1

Ciele práce

Agent ako jednotka schopná konať rozhodnutia (akcie) v prostredí danom Markovovim [22] rozhodovacím procesom hľadá optimálnu stratégiu v zmysle rovnice 2.1. Cieľom agenta je teda nájsť optimálnu stratégiu a maximalizovať tak odmenu. Pre veľký počet stavov je hľadanie optima metódou počítania pravdepodobností prechodov medzi stavmi $P(s, s')$ ťažko vypočítateľné.

Východiskom sú napríklad algoritmy Q-learning, alebo SARSA. Tieto algoritmy počítajú ohodnotenie akcie v danom stave $Q(s(n), a(n))$, ktoré číselne vyjadruje vhodnosť danej akcie. Využitie môžu nájsť [38], [39], [40] napríklad pri plánovných rozhodnutiach v

1. robotike
2. virtuálnych agentových systémoch
3. počítačové hry

Vo všeobecnosti riešia uvedené algoritmy problémy umelej inteligencie, kedy nie je možné zostaviť tréningové dáta v tvare vstup, požadovaný výstup a aplikácia je obmedzená na udeľovanie odmen agentovi za vykonanie zvolenej stratégie [43], [44]. Na rozdiel od evolučných algoritmov (genetické algoritmy, diferenciálna evolúcia, simulované žíhanie), kedy je daná kritériálna funkcia, umožňujú algoritmy Q-learning, alebo SARSA postupne zlepšovať riešenie na princípe hľadania optimálnej stratégie z niekoľkých optimálnych podstratégií - už nájdené optimálne riešenie podstratégie sa nemení. V prípade evolučných algoritmov je ty-

pická zmena všetkých hľadaných parametrov. Nie sú teda vhodné na úlohy kde sa požaduje generovanie postupnosti akcií.

Pre algoritmus Q-learning je zaručená konvergencia k optimálnemu ohodnoteniu (v zmysle 2.1) [41] pre ľubovoľnú metódu výberu akcií - postačuje aby každá akcia mala nenulovú pravdpodobnosť vykonania v prislúchajúcom stave. V prípade SARSA táto konvergencia nie je zaručená pre všetky metódy výberu akcií. Oba algoritmy pracujú v diskretnom čase.

Pre problémy s rádovo stovkami stavov, ktoré sú diskrétné, môže byť funkcia $Q(s(n), a(n))$ realizovaná formou tabuľky. Konvergencia k optimálnemu riešeniu je v tomto prípade zaručená. Pre problémy kde je počet stavov veľmi veľký (tisíce a viac), alebo stavy nenadobúdajú diskrétné hodnoty je potrebné zvoliť aproximáciu tejto funkcie. Konvergencia v tomto prípade už nie je zaručená.

Prístupov ako aproximovať túto funkciu je niekoľko [31], [32], [33], [34]. Najčastejšie používané

1. Diskretizácia stavov spojitých hodnôt tabuľkou
2. Lineárna kombinácia príznakov
3. Dopredná neurónová sieť
4. Neurónová sieť bázičných funkcií

Prvý spôsob predstavuje triviálne riešenie problému redukciou nekonečného počtu stavov na konečný.

Druhý spôsob spočíva v pevne definovaných príznakoch, ktoré závisia od typu problému. Tieto príznaky tvoria súbor funkcií $f_i(s(n), a(n))$. Hodnota $Q(s(n), a(n))$ je daná lineárnou kombináciou týchto príznakov. Hľadá sa teda vektor váh w pre ktorý $Q_b(s(n), a(n), w) = \sum_{i=0}^I w_i f_i(s(n), a(n))$ má minimálnu veľkosť chyby e , definovaná je ako $e(w) = \sum_{s,a} (Q(s(n), a(n)) - Q_b(s(n), a(n), w))^2$ Problematická zostáva voľba príznakových funkcií - ich tvar aj počet.

Tretí spôsob spočíva v použití doprednej neurónovej siete ako univerzálny aproximátor funkcie. Schopnosť aproximovať funkciu doprednou neurónovou sieťou je veľmi dobre známa aj preskúmaná. Pre úlohy Q-learning algoritmu je však nepoužiteľná [42], z dôvo-

dov nemožnosti túto sieť naučiť doteraz dostupnými prostriedkami. Hoci existuje niekoľko prípadov kde sa učenie dá uskutočniť, vo všeobecnosti sú v protiklade dva požiadavky :

1. Učenie siete na požadovanú hodnotu
2. Generovanie požadovanej hodnoty

Sieť teda musí zároveň poskytovať správny výstup pre minulé stavy a zároveň sa učiť na súčasný stav bez toho, aby sa hodnoty z minulých stavov zmenili.

Štvrtý spôsob je využívať lineárnu kombináciu bázičných funkcií. Bázičné funkcie sú dané vopred, avšak ich parametre sa menia v priebehu učenia, podobne ako vektor váh lineárnej kombinácie w . Nech sú ich parametre označené ako v . Cieľom je nájsť také w a v pre ktoré chyba $e(v, w) = \sum_{s,a} (Q(s(n), a(n)) - Q_b(s(n), a(n), v, w))^2$ je minimálna. Kde $Q_b(s(n), a(n), v, w) = \sum_{i=0}^I w_i f_i(s(n), a(n), v_i)$.

Cieľom práce je overiť možnosti aproximácie funkcie $Q(s(n), a(n))$ uvedenými metódami. Vzhľadom na už prebehnutý výskum a problémy dopredných neurónových sietí, sa problematika sústreďuje najmä na hľadanie vhodných bázičných funkcií. Práve v tejto oblasti je venovaný výskumu najväčší priestor. Tieto funkcie by mali byť volené tak, aby zmena parametrov v_i jednej funkcie, neovplyvnila výsledok inde ako pre žiadané $s(n)$ a $a(n)$. Použitie riešenie je potom možné využiť vo veľkých stavových priestoroch, kde možnosti použiť tabuľku zlyhávajú z dôvodov

1. Veľké pamäťové nároky
2. Nutnosť navštíviť a správne spočítať Q pre všetky $s(n), a(n)$

Prvý problém nepredstavuje pre súčasné počítače až tak veľký nedostatok tabuľkového riešenia. Horšia je situácia v prípade vyplňania korektných hodnôt v tabuľke. Práve rekurentnou povahou algoritmov Q-learning a SARSA je časovo veľmi náročné vyplniť tieto hodnoty - mnohonásobne treba navštíviť všetky stavy a vykonať v nich všetky akcie. Práve to je primárny dôvod aproximovať funkciu $Q(s(n), a(n))$.

Kapitola 2

Teoretické východiská učiacich sa systémov na báze odmeňovania

V tejto kapitole budú stručne predstavené učiace sa systémy založené na odmeňovaní. Kapitola si kladie za cieľ ukázať princípy, matematické detaily budú rozobraté v ďalšej kapitole.

2.1 Úvod

Strojové učenie predstavuje ďalší logický krok v počítačovej vede. Programovanie založené na pevne danom správaní prestáva pre mnohé úlohy stačiť. Na softvérové produkty sú kladené čoraz väčšie nároky, a v dohl'adnej dobe sa dá očakávať dosiahnutie hranice tradičného prístupu. Dôvodom je najmä veľmi rozsiahly stavový priestor úloh. Možným východiskom sú adaptívne a učiace sa systémy.

Samotné učenie, alebo adaptácia, predstavuje zmenu parametrov systému. V súčasnosti sú známe tri najpoužívanjšie spôsoby učenia

1. minimalizácia chyby, ktorá je definovaná v každom kroku
2. vyhľadávanie podobných príznakov a zhluková analýza
3. odmeňovanie alebo trestanie vykonaných rozhodnutí

Prvý spôsob zahŕňa systémy učenie na princípe predkladania dvojíc vstup, požadovaný výstup. Učenie sú napríklad metódou najmenších štvorcov [1] [2] [3], alebo často používané gradientové metódy učenia neurónových sietí [4] [5] [6] [27] [9], prípadne iteratívne učenie regulátory [7] [8], používané napr. v polohovacích mechanizmoch. Sú to typické systémy učenia s učiteľom.

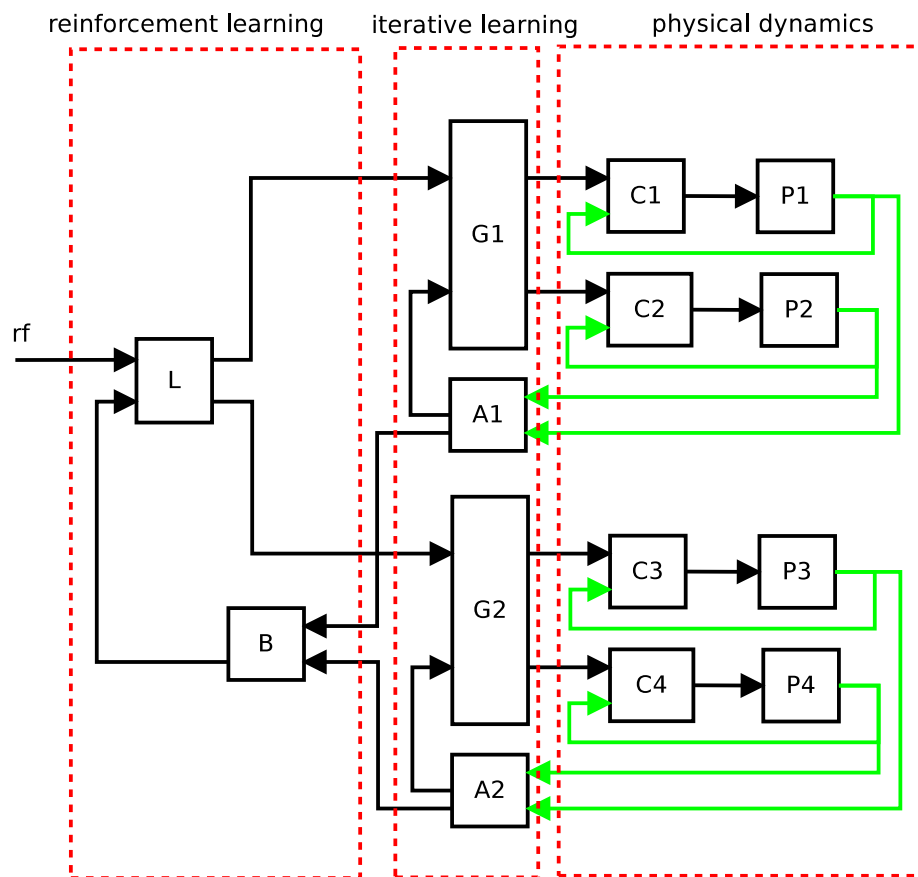
Druhý spôsob je typický zástupca učenia bez učiteľa. Vstupy do systému sú triedené podľa príznakov. Poborné vstupy sú zatriedené do rovnakej skupiny. Definícia podobnosti sa líši podľa aplikácie. Je to vlastne problém o zavedení vhodnej metriky pre danú úlohu. Niekedy postačuje euklidova metrika, pre rôznorodé vlastnosti príznakov je často potrebné ováňovať jednotlivé príznaky. Typickými zástupcami sú Kohonenové siete [18] [19] [20] ich špeciálny prípad je K-means.

Tretí spôsob je použiteľný v situáciách, kedy je možné stanoviť rozdiel výstupu a požadovaného výstupu len v niekoľkých prípadoch, pričom dosiahnutie výstupu vyžaduje niekoľko krokov. Počas týchto krokov môžu byť systému poskytované odmeny, na základe ktorých prehodnocuje svoje rozhodnutia. Použiteľný je teda v situáciách, kde je potrebných viac krokov na dosiahnutie cieľa, ale ich postupnosť nie je známa. To je podstatný rozdiel oproti napr. iteratívne učenými regulátormi, kde sa síce tiež dosahuje požadovaná hodnota niekoľkými krokmi, avšak je v každom z nich známa chyba oproti požadovanému správaniu. Práve tejto kategórii je venovaná dizertačná práca.

Hybridné hierarchické systémy predstavujú kombináciu niekoľkých princípov. V prípade riadenie, je obvykle dynamika stroja známa - je možné stanoviť jeho fyzikálny model, prípadne použiť adaptívny systém (je možné ukázať, že pre množstvo mechanických sústav je PID regulátor vyhovujúce riešenie). Samotný blok riadenia regulátorom však nevypovedá o žiadanej hodnote - snaží sa na ňu systém dostať, ale nevie ju stanoviť. V jednoduchých situáciách to nepredstavuje problém.

Pre komplexné riadenie je nevyhnutné systém rozdeliť do niekoľkých vrstiev, obrázok 2.1.

Obrázok je možné vysvetliť na príklade robotického ramena, so 4 stupňami voľnosti. Cieľom je riadiť 4 sústavy $P1$ až $P4$, Na úrovni fyzikálnej vrstvy je možné použiť adaptívne PID regulátory, do ktorých vstupuje výstup sústavy a požadovaná hodnota. V prípade ramena je požadovanou hodnotou natočenie jednotlivých kĺbov. Tieto uhly sú spočítané blokmi $G1$ a



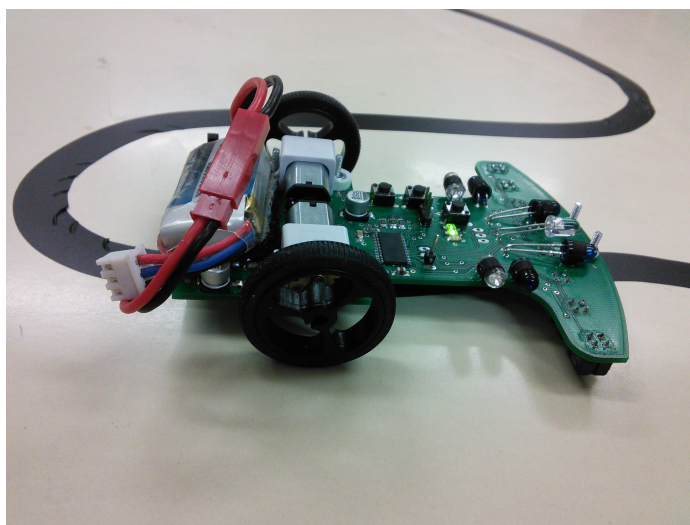
Obr. 2.1: Hierarchický systém riadenia

G2. Vstupom do regulátorov na fyzikálnej úrovni je tak postupnosť uhlov generovaná blokmi G1 a G2. Na tejto úrovni sú zavedené aj agregačné bloky A1 a A2, ktoré poskytujú G-blokom informáciu o stave systému. Bloky G spolu s regulátormi C teda úspešne riešia problém presunu ramena po zadanej trajektórii. Veľa systémom to postačuje - operátor výroby ľahko určí požadované trajektórie, ktoré má robot vykonávať. V prípade komplexného problému, kedy požadované trajektórie nie sú známe, má zmysel použiť blok L - tretia úroveň, a nechať systém nech si trajektórie a ich poradie určí sám. Používateľ systému do tohto procesu zasahuje odmeňovaním alebo trestaním výsledného riešenia. Príkladom môže byť niekoľko kroková montáž výrobku. Ktoré diely zložiť ako prvé a ktoré ako posledné, aby bol výrobok zložený v čo najkratšom čase

triviálny príklad : osádzanie plošného spoja, osadiť najprv veľké alebo najprv malé súčiastky? Druhý príklad : V hre šachy je hráč postavený pred rozhodnutie či najprv vybudovať

obranu a stratiť niekoľko t'ahov, alebo sa radšej sústrediť na útok a riskovať slabinu vo vlastnej obrane?

V experimentálnej časti bude ako doplnková ukážka uvedený riadiaci mechanizmus robota Motko Aftermath, sledujúceho čiaru, kde sa požadovaná hodnota určuje predikciou pomocou neurónovej siete a následne vstupuje do konvenčného regulátora 2.2.

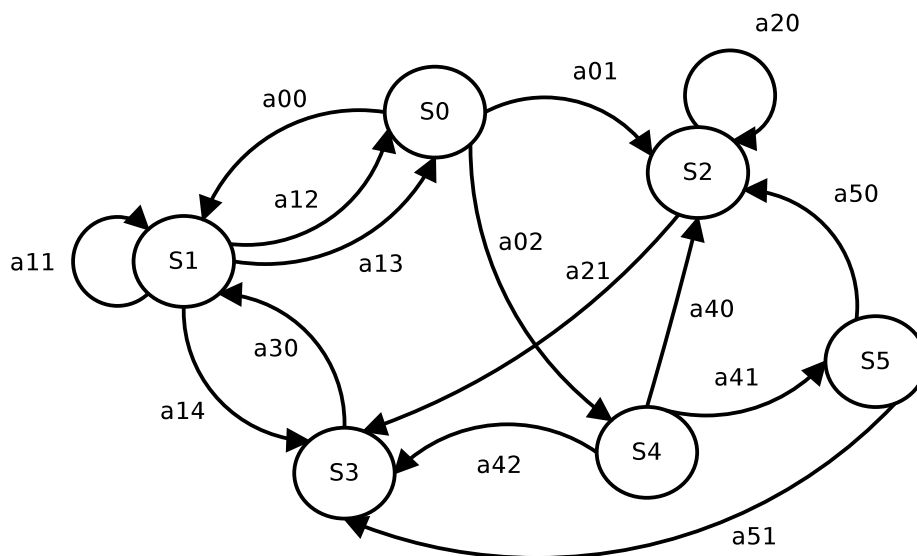


Obr. 2.2: Ranná verzia robota

2.2 Markovove rozhodovacie procesy

Množstvo úloh je možné popísať množinou stavov \mathbb{S} . Pre každý stav je ďalej definovaná množina akcií $\mathbb{A}_{\mathbb{S}}$ [21] [22]. Pre každý stav a každú akciu, ktorú je v ňom možné vykonať existuje pravdepodobnostná funkcia prechodu do ďalšieho stavu $P(s, s')$ a za uskutočnené prechody je daná funkcia odmien $R(s, s')$. Formálne je teda Markovov proces päťica $(\mathbb{S}, \mathbb{A}_{\mathbb{S}}, P(s, s'), R(s, s'), \gamma)$, kde $\gamma \in (0, 1)$ je faktor zabúdania a volí sa ako parameter procesu. Jeho význam bude vysvetlený v ďalšej kapitole. Príklad Markovovho rozhodovacieho procesu je na obrázku 2.3. Prechody v grafe znamenajú jednotlivé akcie.

V priestore stavov \mathbb{S} existuje výkonná jednotka - agent. Je to modul, ktorý na základe vstupnej informácie - stave vyberie jednu z akcií. To s akou pravdepodobnosťou sa daná akcia vykoná je dané prostredím (popísané Markovovým rozhodovacím procesom).



Obr. 2.3: Markovov rozhodovací proces

Je potrebné poznamenať, že $P(s, s')$ nie je agentovi známa. Agent teda nemôže vedieť kam sa vykonaním vybranej akcie dostane. Tento fakt značne komplikuje prehľadávanie priestoru. Ďalším typickým rysom je, že odmeňovacia funkcia $R(s, s')$, ktorá je zadaná, nadobúda pre väčšinu prechodov z s do s' hodnotu 0. Je to dôsledok prirodzeného spôsobu riešenia problémov delením na čiastkové kroky a nie je okamžite známa veľkosť úžitku po vykonaní čiastkového kroku. Často je táto skutočnosť známa len pre pár vybraných prechodov. Dobrým príkladom tejto situácie je hra Go.



Obr. 2.4: Hra Go

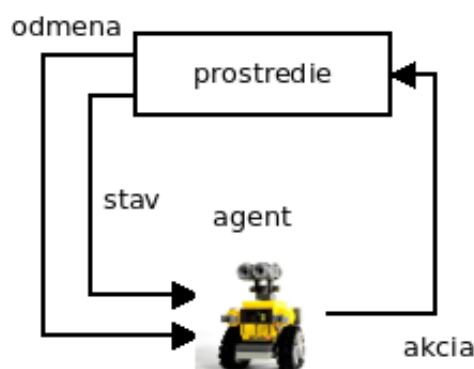
Položením kameňa na goban nie okamžite známe ako to ovplyvní priebeh hry. Je bežné,

že sa to prejaví až po mnohých krokoch.

2.3 Princípy učenia s odmeňovaním

V systémoch založených na predkladaní dvojíc vstup - požadovaný výstup je možné stanoviť chybu a tú vhodnými metódami minimalizovať. V prípade systému s odmeňovaním sa požaduje od výstupu postupnosť akcií, ktoré maximalizujú celkovú odmenu. Príkladom môže byť rozhodovanie robota, ktorý má splniť cieľ pozostávajúci z niekoľkých elementárnych úkonov, ale postupnosť týchto elementárnych úkonov nie je známa - nie je teda definovaná požadovaná hodnota výstupu.

Riešením tohto problému je zavedenie systému odmeňovania agenta (robota) 2.5.



Obr. 2.5: Učenie s odmeňovaním

Táto metóda sa nazýva **reinforcement learning**. V súčasnosti predstavuje najmocnejší nástroj strojového učenia v oblasti umelej inteligencie. Umožňuje riešiť veľmi komplexné problémy a uplatnenie nachádza nielen v počítačových hrách, ale napr. aj v robotike. Jej princíp je možné zhrnúť do týchto bodov :

1. Zistenie stavu
2. Výber akcie
3. Vykonanie akcie
4. Prechod do ďalšieho stavu

5. Získanie odmeny alebo trestu

6. Učenie sa zo získanej skúsenosti

Zistenie stavu je umožnené vnímaním prostredia, napr. pomocou senzorov. V najjednoduchšom prípade si je možné ako stav predstaviť polohu. Vo všeobecnosti to ale nemusia byť ani priamo dáta zo vstupov, ale len príznaky - robot je na križovatke $p_0 = 0.9$, robot je v uličke $p_1 = 0.3$. Sada príznakov umožňuje zredukovať dimenziu stavového priestoru (tá musí byť pre ďalej rozoberané algoritmy konečná, vyplynie to z prezentovaných rovníc).

Výber akcie je uskutočnený na základe ohodnotení akcií získaných v minulosti. Tieto ohodnotenia sú samozrejme závislé od stavu v ktorom sa systém nachádza - tá ista množina akcií má iné ohodnotenie keď je robot na križovatke a iné keď je v uličke. Vo fáze prieskumu prostredia môže agent vyberať tie akcie, ktoré boli v minulosti málo často vykonané, alebo môže vyberať náhodne. Definovaním veľkosti zmeny ohodnotenia akcie môže uprednostniť tie akcie, ktorých ohodnotenia vykazujú veľkú zmenu - pretože je pravdepodobné že sa tým dostane do nepreskúmaných stavov. Vo fáze kedy agent už nepreskúma prostredie, môže vyberať len najlepšie ohodnotenú akciu. Prípadne môžu existovať v prostredí s viacerími agentami prieskumníci a vykonávatelia. Navzájom si môžu vymieňať skúsenosti.

Vykonanie akcie v deterministickom prostredí sa vybraná akcia vykoná vždy rovnako. V nedeterministickom prostredí sa vykonanie sa akcie riadi nejakou pravdepodobnostnou funkciou - napr. robotovi môže prešmyknúť koleso. Výber agenta vykonať nejakú akciu, teda ešte nutne neznamena že sa táto akcia aj naozaj vykoná. Táto sutočnosť je daná charakterom prostredia.

Prechod do ďalšieho stavu je obvyklým dôsledkom vykonania akcie. Samozrejme, že v danom stave môže existovať aj taká akcia ktorá nespôsobí zmenu stavu - napr. spomínané prešmyknutie kolesa - robot sa nepohne.

Získanie odmeny alebo trestu po vykonaní akcie, existuje odmeňovací mechanizmus zadany tvorcom systému. Preto sa nedá hovoriť o umelej inteligencii - správanie sa agenta je v deterministickom prostredí plne určené týmto mechanizmom - len nie je známe. Obvykle platí, že kladné hodnoty dosahuje odmena a záporné trest. Nulová hodnota symbolizuje neznalosť zadavateľa, a nevie teda určiť či dané konanie bolo dobré alebo zlé. Nesmierne dô-

ležitý je fakt, že pre veľkú väčšinu rozhodnutí je výsledkom odmeny práve nula. Príkladom môže byť japonská hra Go - pre úspešné vedenie partie nestačí ohodnotiť len aktuálny stav na gobane, je potrebné uvažovať viac ťahov dopredu - úspešnosť ťahu sa často ukáže až po vykonaní množstva ďalších ťahov, kedy je odmena už nenulová.

Práve dominancia nulových odmien je unikátom pre reinforcement learning algoritmy. Od zadávateľa sa vyžaduje minimum znalostí, je dokonca možné ukázať, že stačí odmena len v cieľovom stave (i keď proces učenia bude v tomto prípade trvať veľmi dlho).

2.4 Ohodnocovanie vykonaných akcií

Odmena je získaná z prostredia po vykonaní akcie. Ohodnotenie akcie v danom stave je tvorené predošlými skúsenosťami z vykonania predošlých akcií a získania odmien. Podstatou učenia je teda ohodnotenie vykonaných akcií v danom stave aby bolo možné v každom stave rozhodnúť, ktorá akcia je najlepšia - vyberá sa teda postupnosť akcií π pre ktorú je funkcia

$$\Lambda(\pi) = \sum_{n=0}^{L(\pi)} \gamma^n P_{\pi(n)}(s(n), s(n-1)) \quad (2.1)$$

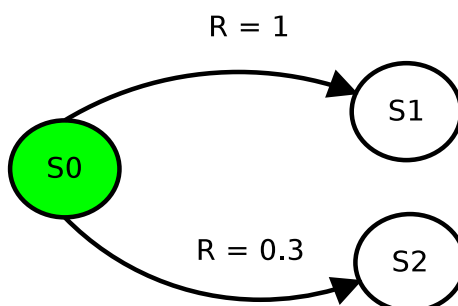
maximálna. Kde $\gamma \in \langle 0, 1 \rangle$ je koeficient zabúdania, $P_{\pi(n)}(s(n), s(n-1))$ je odmeňovacia funkcia po prechode zo stavu $s(n-1)$ do stavu $s(n)$ vykonaním $\pi(n)$ a $L(\pi)$ je dĺžka postupnosti π

Hľadanie mechanizmu nájdenia maxima $\Lambda(\pi)$ bude vysvetlené na nasledujúcich príkladoch. Odmena bude známa v každom prechode a je definovaná ako $R(s, a) \in \mathbb{R}$

Pre ilustráciu bude postupnosť prechodov zapísaná priamo ako parameter funkcie Λ , napr : $\Lambda(\{S_0, S_7, S_5\})$ predstavuje prechody z S_0 do S_7 a z S_7 do S_5 . Ďalej sa predpokladá že $\gamma = 1$. Je daný systém s tromi stavmi, a dvoma prechodmi 2.6. Pre ilustráciu je systém tak jednoduchý, že každá akcia má známu odmenu.

V tomto systéme je ohodnotenie $\Lambda(S_a, S_b)$ naozaj triviálne.

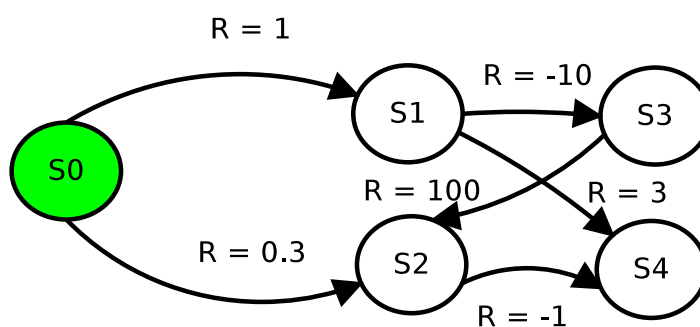
1. $\Lambda(\{S_0, S_1\}) = 1$
2. $\Lambda(\{S_0, S_2\}) = 0.3$



Obr. 2.6: Ohodnocovanie akcií v trojstavovom systéme

Najlepšia cesta je potom $\{S_0, S_1\}$.

V prípade systému s viacerými stavmi 2.7, ale známimi ohodnoteniami v každom prechode bude situácia nasledovná.



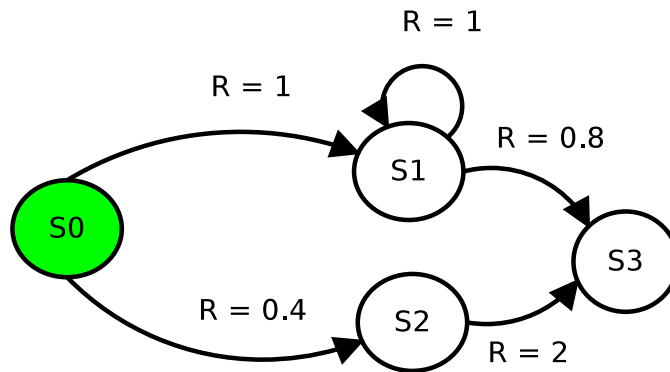
Obr. 2.7: Ohodnocovanie vo viacstavovom systéme

Ohodnotenie ciest :

1. $\Lambda(\{S_0, S_1, S_3\}) = 1 + (-10) = -9$
2. $\Lambda(\{S_0, S_1, S_4\}) = 1 + 3 = 4$
3. $\Lambda(\{S_0, S_2, S_4\}) = 0.3 + (-1) = -0.7$
4. $\Lambda(\{S_0, S_1, S_3, S_2, S_4\}) = 1 + (-10) + 100 + (-1) = 90$
5. ...

Jednoduchým sčítaním ohodnotení rôznych ciest je tak možné nájsť optimálnu postupnosť akcií.

Ak sa v systéme nachádza cyklus 2.8 (na obrázku je triviálny prípad) potom ohodnotenie bude divergovať. Agent bude mať možnosť vykonávať akciu, ktorá neustále pripočítava kladnú odmenu, nevyhnutne tak vyberie túto stratégiu, pretože tak získa nekonečne veľkú odmenu.



Obr. 2.8: Ohodnocovanie s cyklom

Ohodnotenie niektorých ciest :

1. $\Lambda(\{S_0, S_2, S_3\}) = 0.4 + 2 = 2.4$
2. $\Lambda(\{S_0, S_1, S_3\}) = 1 + 0.8 = 1.8$
3. $\Lambda(\{S_0, S_1, S_1, S_1\}) = 1 + 1 + 1 = 3$
4. $\Lambda(\{S_0, S_1, S_1, S_1, S_1\}) = 1 + 1 + 1 + 1 = 4$
5. $\Lambda(\{S_0, S_1, S_1, S_1, S_1, S_1, \dots\}) = 1 + 1 + 1 + 1 + 1 + \dots + 1 = \infty !!!!$

Riešením je práve zavedenie faktora zabúdania γ . Ilustračne bola zvolená $\gamma = 0.9$. Agent tak síce urobí niekoľko cyklov S_1, S_1 , po čase ale hodnota odmeny konverguje a minulé príspevky dávajú čoraz menší a menší úžitok a nevyhnutne tak po niekoľkých cykloch zmení rozhodnutie a prejde z S_1 do S_3 .

Ohodnotenie niektorých ciest :

1. $\Lambda(\{S_0, S_2, S_3\}) = 0.9 * 0.4 + 2 = 2.36$
2. $\Lambda(\{S_0, S_1, S_3\}) = 0.9 * 1 + 0.8 = 1.7$

3. $\Lambda(\{S_0, S_1, S_1, S_1\}) = 1 + 0.9 * (1 + 0.9 * 1) = 2.71$
4. $\Lambda(\{S_0, S_1, S_1, S_1, S_1\}) = 1 + 0.9 * (1 + 0.9 * (1 + 0.9 * 1)) = 3.439$
5. $\Lambda(\{S_0, S_1, S_1, S_1, S_1, S_1, \dots\}) = 10 < \text{---}$
6. $\Lambda(\{S_0, S_1, S_1, S_1, S_1, S_1, \dots, S_3\}) = 10.8 < \text{---}$

Cyklovanie v S_1 teda prinesie celkovú odmenu 10, ale pri prechode do S_3 je celková odmena 11.

Formálny prepis uvedených úvah vedie na Q-learning algoritmus, ktorý zohľadňuje Bellmanov princíp optimality.

2.5 Jednostavový systém

Prv než bude uvedené úplne znenie Q-learning algoritmu je vhodné rozobrať ešte jeden príklad, a to systém s jedným stavom 2.9. Tento problém vznikol z nasledujúcej úvahy : je daný robot, ktorý má jeden senzor vzdialenosti od cieľa (nevie teda určiť smer (napr. senzor intenzity osvetlenia)) a možnosti pohybu o elementárny krok vpred, vľavo, vpravo, vzad. Ako zvolit' postupnosť akcií aby robot došiel do cieľa - minimalizoval vzdialenosť? Navyše sa predpokladá, že senzor je nekvalitný a záludný zároveň : poskytuje informáciu len o tom či sa situácia zlepšila alebo zhoršila (oproti predošlému meraniu), a s určitou pravdepodobnosťou generuje náhodnú hodnotu - náhodne zmení výsledok (šum).

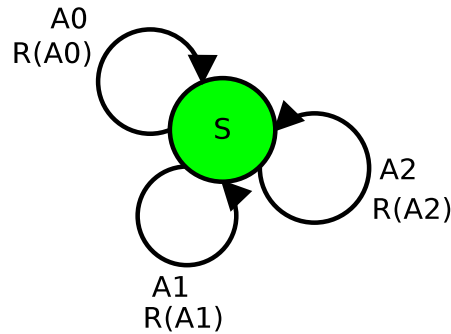
To vedie na zostavenie odmeňovacej funkcie ako

$$R(n) = \begin{cases} k & \text{ak } d(n) - d(n-1) < 0 \\ -k & \text{inak} \end{cases} \quad (2.2)$$

kde $d(n)$ je zmeraná vzdialenosť a na $d(n) - d(n-1) < 0$ sa pozerá ako uzavretú časť - vstupom do algoritmu teda nie je samotné $d(n)$ ale až $R(n)$. Pre k platí

$$k(n) = \begin{cases} -1 & \text{ak } rnd(0, 1) < p \\ 1 & \text{inak} \end{cases} \quad (2.3)$$

kde p je pravdepodobnosť zmeny nameranej hodnoty $p \in \langle 0, 1 \rangle$ a $rnd(0, 1)$ generuje náhodné číslo z intervalu $\langle 0, 1 \rangle$.



Obr. 2.9: Jednostavový systém s troma akciami

Kľúčový je výber akcie $a(n)$ z konečnej množiny akcií \mathbb{A} , vychádza sa z predstavy : ak má vykonaná akcia dobré ohodnotenie, vykoná sa znova, ak má zlé ohodnotenie, vyberie sa iná, náhodná akcia. Formálne zapísané

$$a(n) = \begin{cases} a(n-1) & \text{ak } Q_{n-1}(a(n-1)) > 0 \\ \text{random}(a(n-1)) & \text{inak} \end{cases} \quad (2.4)$$

Samotné Q predstavujúce ohodnotenie sa potom vypočíta ako

$$Q_n(A(n)) = R(n) + \gamma \max_{a'(n-1) \in \mathbb{A}} Q_{n-1}(a'(n-1)) \quad (2.5)$$

Algoritmus je teda veľmi jednoduchý a možno ho použiť na plánovanie pohybu robota. Robotovi tak nie je vnútené správanie programátorom, ale sám sa naučí ktoré akcie vyberať.

2.5.1 Výsledky experimentu

Overenie prebehlo s 32 virtuálnymi robotmi. Cieľom bolo naháňať pohyblivý cieľ, ktorý sa pohyboval po kružnici s rovnicou

$$X(n) = (0.7\cos(ndt), 0.7\sin(ndt)) \quad (2.6)$$

kde $dt = 0.001$. Roboti mohli vyberať zo 4 akcií ktorými menili svoju polohu

$$\mathbb{A} = ((0, 1), (0, -1), (1, 0), (-1, 0)) \quad (2.7)$$

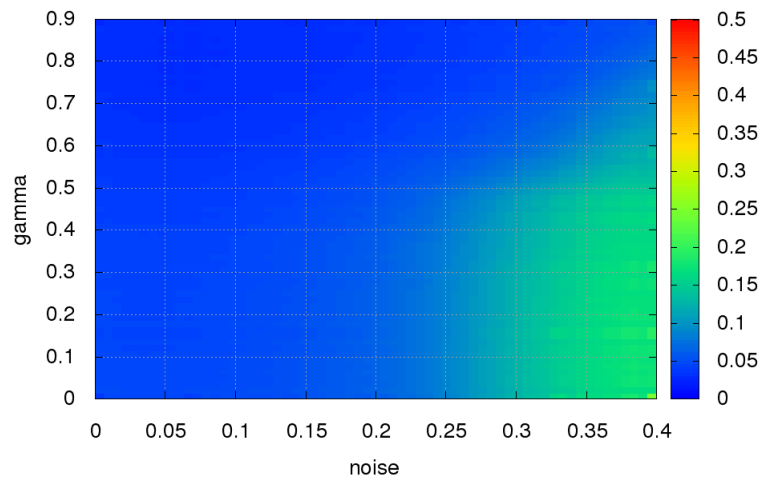
a zmena poloha $R(n)$ robota pri výbere k -tej akcie je potom

$$R(n) = R(n-1) + A(k)dt \quad (2.8)$$

Počiatkové polohy robotov $R(0)$ boli náhodné. Celý experiment prebiehal v dvojrozmernom priestore obmedzenom na rozsah polohy $\langle -1, 1 \rangle$. Metrika vzdialenosti bola Euklidovská.

Po 25000 iteráciach sa počítala priemerná chyba z 32 robotov. Vyšetrovala sa závislosť tejto chyby od zmeny parametrov γ a p (na grafe ako gamma a noise). Pre vybrané prípady boli exportované aj dráhy prvých 8 robotov.

Prostredie sa tak postupne mení z plne deterministického bez šumu až po hodnotu 0.4, čo predstavuje 40% pravdepodobnosť zmeny hodnoty senzora. Graf závislosti priemernej chyby od parametrov γ a p je na obrázku 2.10.

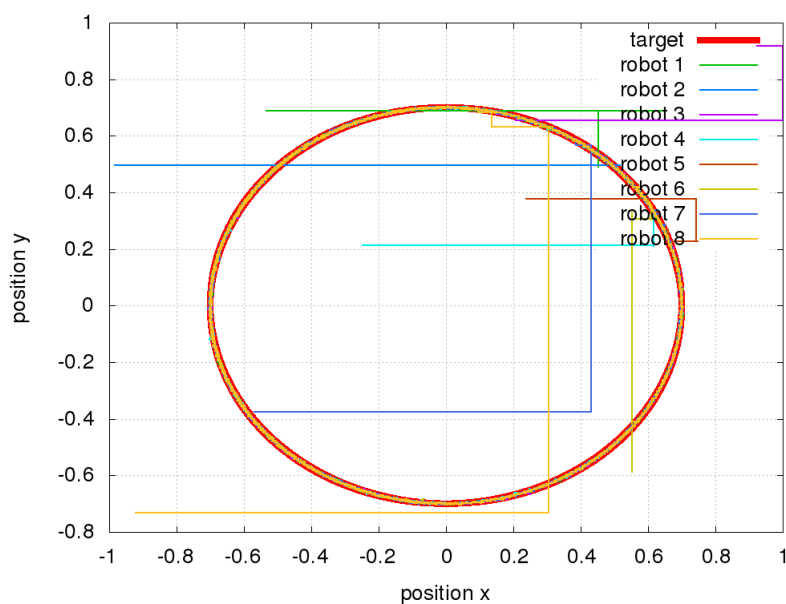


Obr. 2.10: Priemerná chyba 32 robotov po 25000 iteráciach

Dráhy robotov pre niektoré zvolené parametre sú uvedené v tabuľke 2.1

γ	p	graf dráhy	graf vývoja chyby
0.7	0.0	2.11	2.12
0.7	0.3	2.13	2.14
0.0	0.4	2.15	2.16
0.7	0.4	2.17	2.18
0.9	0.4	2.19	2.20
0.98	0.4	2.21	2.22

Tabuľka 2.1: Vybrané parametre experimentu a odkaz na grafické znázornenie výsledku

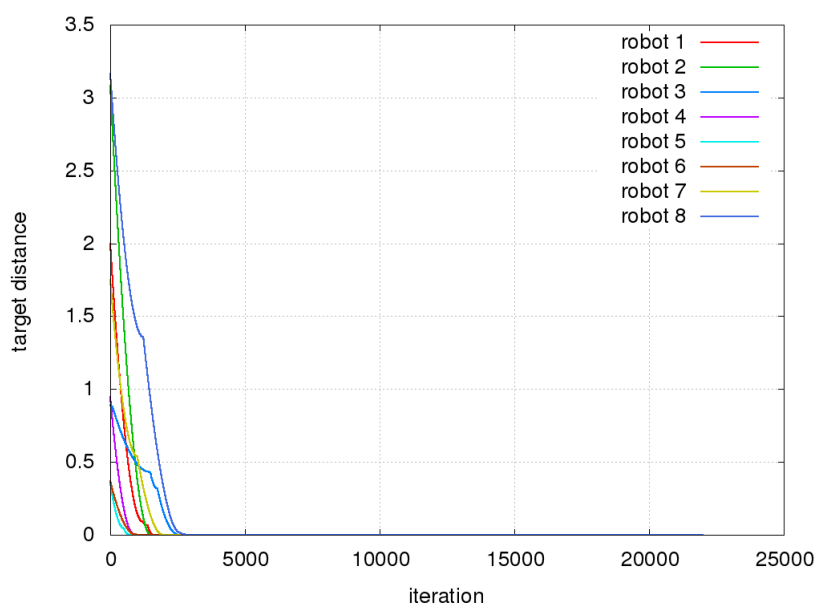


Obr. 2.11: Dráha robotov pre $\gamma = 0.7$ $p = 0.0$

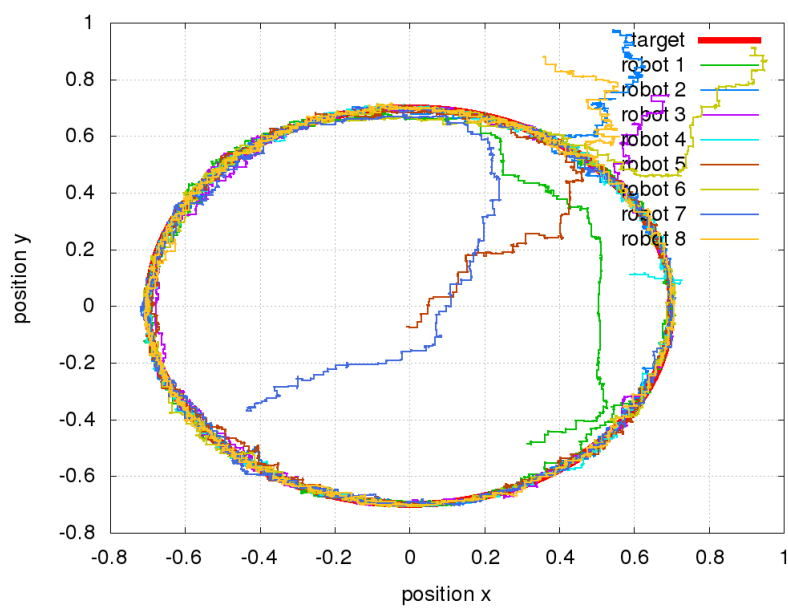
Z experimentov je zrejmé, že pre málo zašumené prostredie nemá hodnota parametra γ veľký význam, obrázky : 2.11 2.12, 2.13 2.14.

S postupným nárastom šumu však pri nevhodne zvolenej hodnote γ roboti vykazujú veľkú chybu. Je preto potrebné náležite zväčšovať parameter γ , tento priebeh zlepšovania výsledku zvyšovaním parametra γ je zrejmý z obrázkov : 2.15 2.16, 2.17 2.18, 2.19 2.20, 2.21 2.22.

Algoritmus bol pomenovaný nanoQ a je pod licenciou GNU GPL dostupný na [23]. Je napísaný v jazyku C len s využitím pevnej rádovej čiarky. To umožňuje jeho implementáciu aj do menej výkonných mikrokontrolérov, s jadrami napr. Cortex M0 alebo MSP430. Učenie

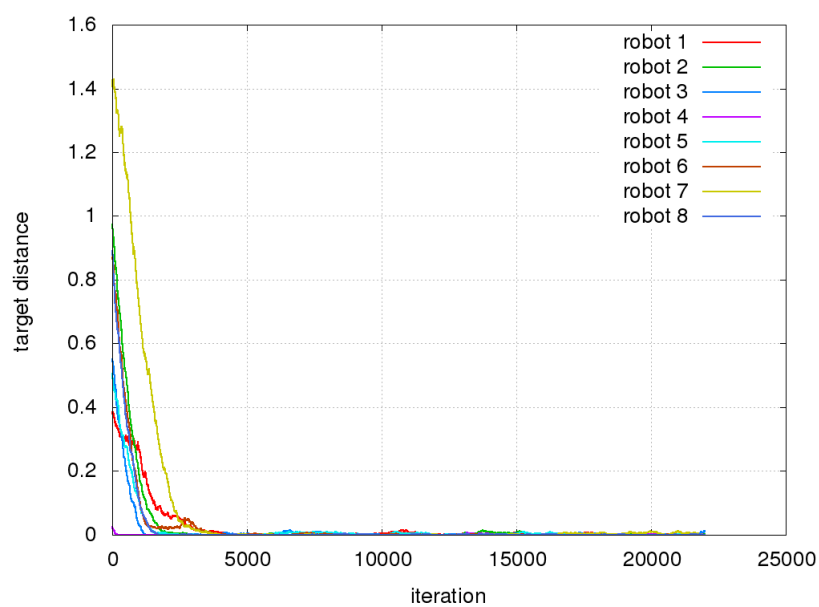


Obr. 2.12: Vzdialenosť robotov od cieľa pre $\gamma = 0.7$ $p = 0.0$

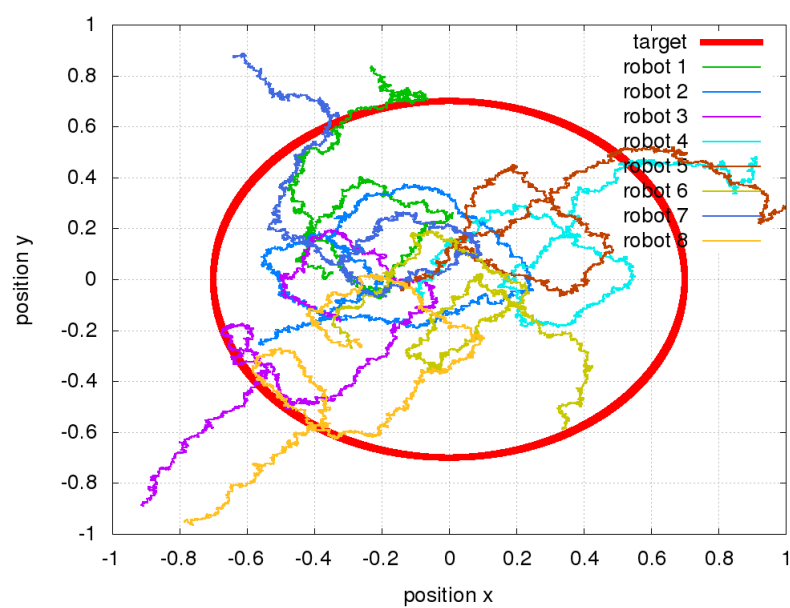


Obr. 2.13: Dráha robotov pre $\gamma = 0.7$ $p = 0.3$

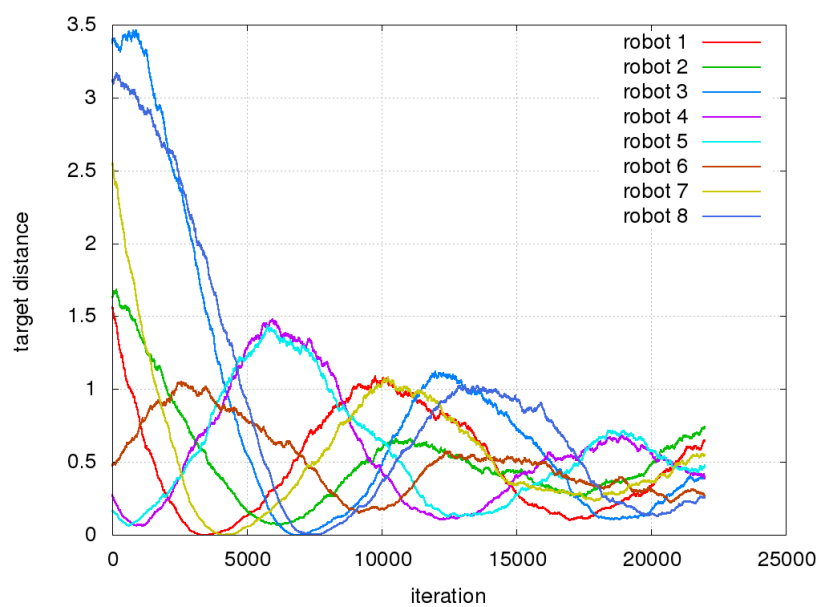
prebieha v reálnom čase a nevyžaduje veľký výpočtový výkon. Podobne, aj pamäťové nároky rastú lineárne s počtom akcií. Pre veľký počet akcií je vždy možné použiť aproximáciu, ktorá bude rozobratá v ďalších častiach práce.



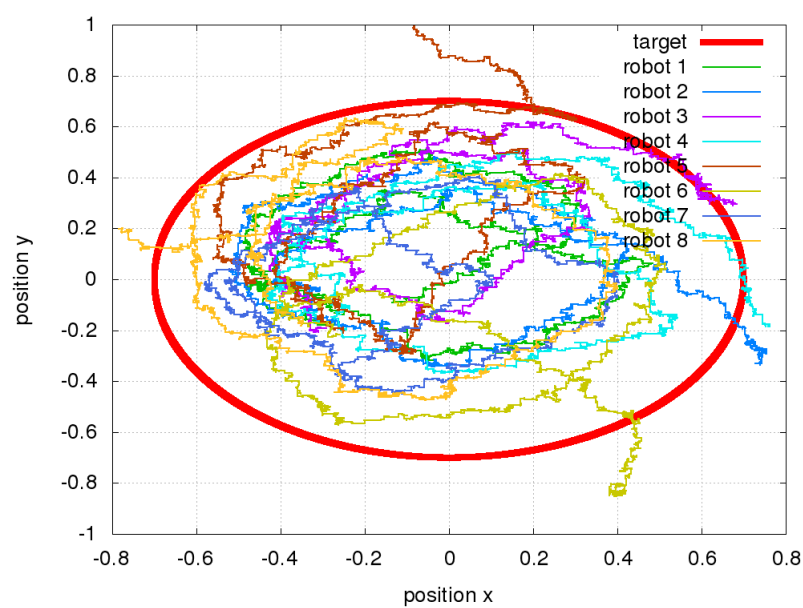
Obr. 2.14: Vzďialenosť robotov od cieľa pre $\gamma = 0.7p = 0.3$



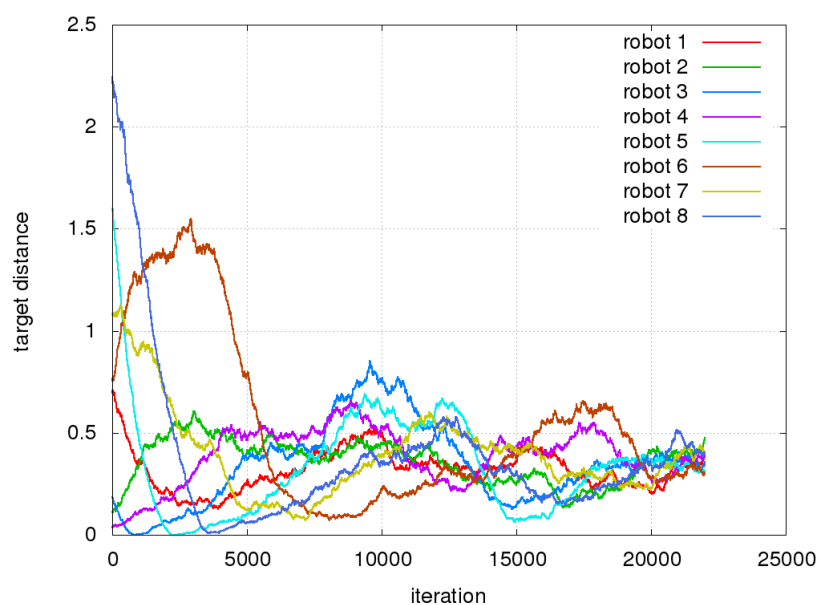
Obr. 2.15: Dráha robotov pre $\gamma = 0.0p = 0.4$



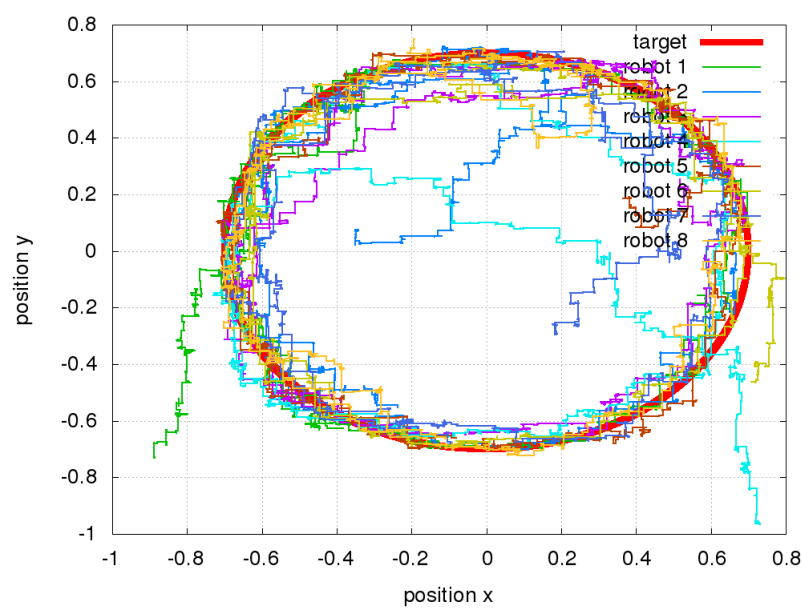
Obr. 2.16: Vzďialenosť robotov od cieľa pre $\gamma = 0.0$ $p = 0.4$



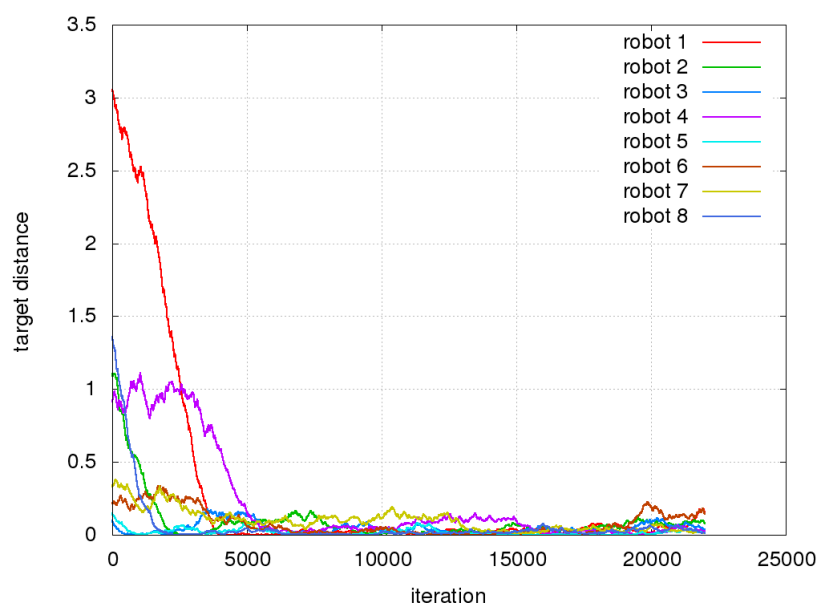
Obr. 2.17: Dráha robotov pre $\gamma = 0.7$ $p = 0.4$



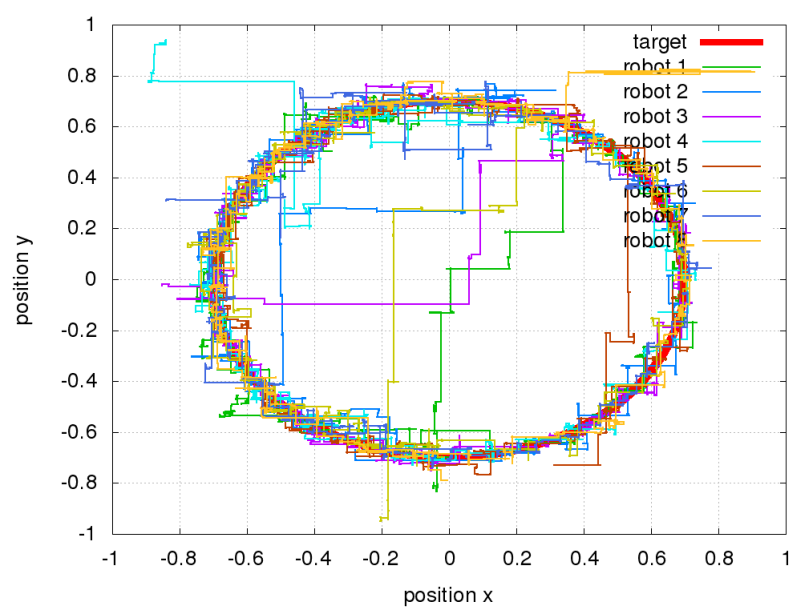
Obr. 2.18: Vzďialenosť robotov od cieľa pre $\gamma = 0.7p = 0.4$



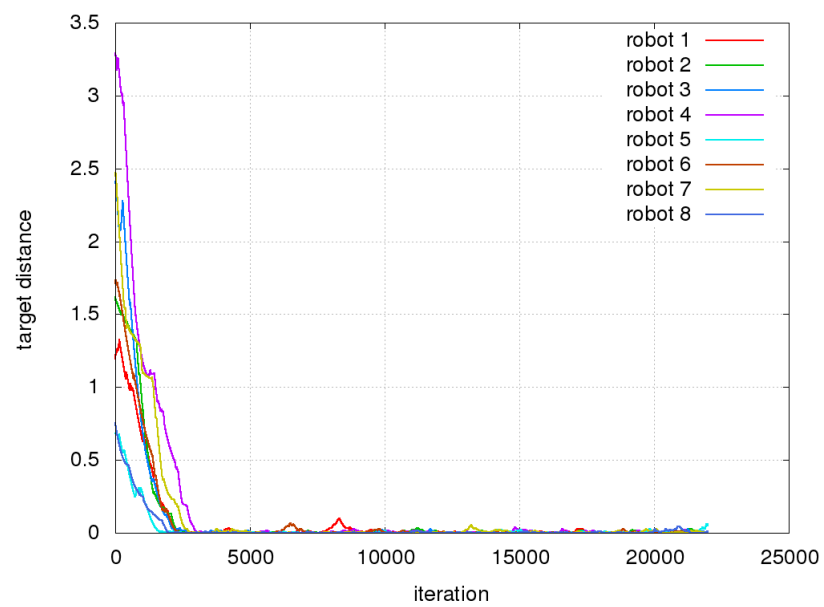
Obr. 2.19: Dráha robotov pre $\gamma = 0.9p = 0.4$



Obr. 2.20: Vzďialenosť robotov od cieľa pre $\gamma = 0.9p = 0.4$



Obr. 2.21: Dráha robotov pre $\gamma = 0.98p = 0.4$



Obr. 2.22: Vzdialenosť robotov od cieľa pre $\gamma = 0.98$ $p = 0.4$

Kapitola 3

Q-learning algoritmus

Q-learning algoritmus je definovaný pre časovo diskkrétne systémy. Agent ktorý prechádza stavový priestor vykonaním niektorej z vopred daných akcií získava za tieto prechody odmeny. Cieľom algoritmu je ohodnotiť všetky akcie v jednotlivých stavoch, tak aby bol dosiahnutý ustálený stav a v každom stave bolo možno vybrať akciu prinášajúcu najväčšiu odmenu, v zmysle s 2.1.

3.1 Definícia algoritmu

Autorom Q-learning algoritmu je Christopher J.C.H. Watkins, v roku 1992 publikoval článok kde tento algoritmus predstavil [10] a niekoľko ďalších vysvetlení tohto algoritmu je možné nájsť v [11] alebo [12]. Dôkazy o konvergencii k optimálnemu riešeniu (v zmysle s 2.1) sú k dispozícii [13], [14], [15], [16].

Daná je množina stavov \mathbb{S} a akcií \mathbb{A} , kde $\mathbb{S} \in \mathbb{R}^{n_s}$ a $\mathbb{A} \in \mathbb{R}^{n_a}$, kde n_s a n_a sú počty prvkov stavového vektora a vektora akcií.

Existuje prechodová funkcia

$$s(n+1) = \lambda(s(n), a(n)) \quad (3.1)$$

zo stavu $s(n) \in \mathbb{S}$ použitím akcie $a(n) \in \mathbb{A}$, táto funkcia je ale algoritmu neznáma, v reálnom prostredí je jej výsledok obvykle stochastický.

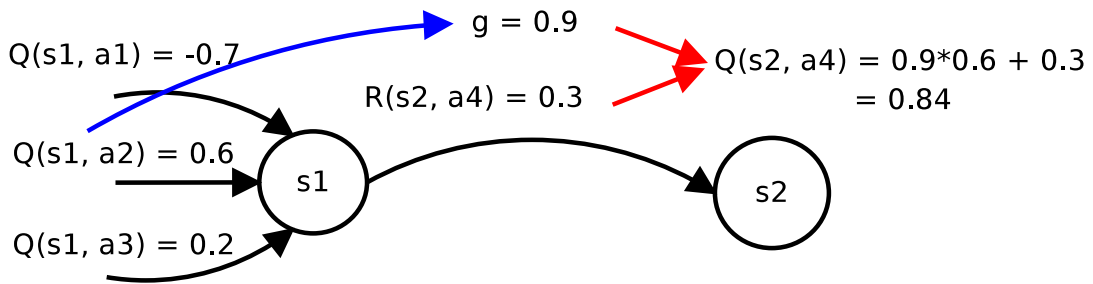
Ďalej je daná odmeňovacia funkcia $R(s(n), a(n))$, ktorá vyjadruje okamžité ohodnotenie konania agenta v $s(n)$ a $a(n)$. V reálnych aplikáciach táto funkcia nadobúda takmer v každom $s(n)$ a $a(n)$ hodnotu 0. Pre správnu funkciu algoritmu, musí byť aspoň jedna hodnota nenulová - napr. ohodnotenie dosiahnutia cieľového stavu (samotná existencia cieľového stavu však pre algoritmus nie je potrebná).

Funkcia ohodnotení je definovaná ako

$$Q_n(s(n), a(n)) = R(s(n), a(n)) + \gamma \max_{a(n-1) \in \mathbb{A}} Q_{n-1}(s(n-1), a(n-1)) \quad (3.2)$$

- $R(s(n), a(n))$ je odmeňovacia funkcia
- $Q_{n-1}(s(n-1), a(n-1))$ je funkcia ohodnotení v stave $s(n-1)$ pre akciu $a(n-1)$
- γ je odmeňovacia konštanta a platí $\gamma \in (0, 1)$.

Funkcia 3.2 definuje ohodnotenie akcií vo všetkých stavoch t.j. agent, ktorý sa dostal do stavu $s(n)$ vykonaním akcie $a(n)$ zo stavu $s(n-1)$ získal odmenu $R(s(n), a(n))$ a zlomok najväčšieho možného ohodnotenia, ktoré mohol získať dostaním sa do stavu $s(n-1)$, situáciu ilustruje obrázok 3.1.



Obr. 3.1: Ilustrácia funkcie ohodnotení, pre $\gamma = 0.9$

Je potrebné poznamenať, že práve časť $\max_{a(n-1) \in \mathbb{A}} Q_{n-1}(s(n-1), a(n-1))$ zabezpečuje nezávislosť konvergencie k optimu bez ohľadu voľby stratégie výberu akcie - postačuje, aby každá akcia, v každom stave mala nenulovú pravdepodobnosť vykonania. Určitým variantom je algoritmus SARSA [17]

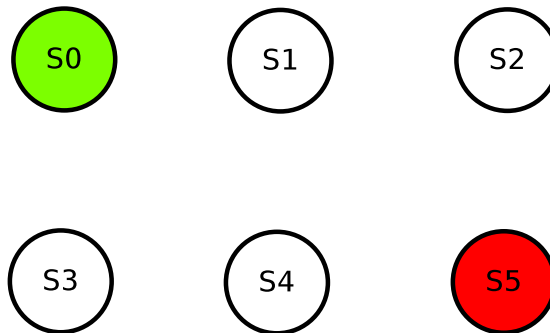
$$\begin{aligned}
Q_n(s(n), a(n)) = & \\
& (1 - \alpha)Q_{n-1}(s(n), a(n)) + \\
& \alpha(R(s(n), a(n)) + \gamma Q_{n-1}(s(n-1), a(n-1)))
\end{aligned} \tag{3.3}$$

kde $\alpha \in (0, 1)$, hodnota $Q_n(s(n), a(n))$ sa teda ustáli na strednej hodnote a závisí od stratégie výberu akcií. Q-learning teda vychádza z toho, čo najlepšie sa mohlo stať a SARSA z toho čo sa naozaj stalo.

Nasledujúce obrázky ilustrujú beh algoritmu pre systém so 6 stavmi pre $\gamma = 0.8$. Na začiatku nie sú známe ani samotné prechody medzi stavmi (Obr. 3.2), bol definovaný 1 cieľový stav $S5$, agent začína v stave $S0$ (môže však v ľubovoľnom inom). Ďalej sa pre jednoduchosť predpokladá že

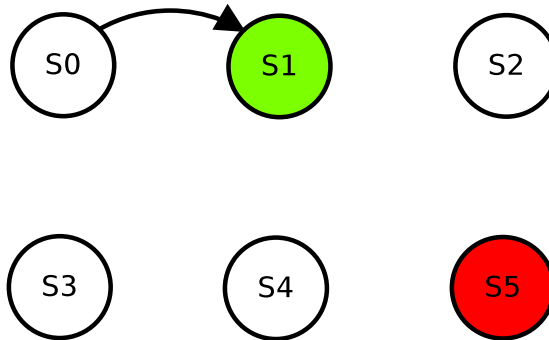
$$R(s(n), a(n)) = \begin{cases} 1 & \text{ak } s(n) = S5 \\ -0.5 & \text{ak } s(n) = S4 \wedge a(n) = A_y \\ 0 & \text{inak} \end{cases} \tag{3.4}$$

t.j. odmeňovacia funkcia nadobúda hodnotu 1 len ak sa agent dostal do stavu $S5$ a pre ilustráciu je definovaná aj jedna záporná odmena pri prechode z $S4$ do $S3$ akciou A_y .

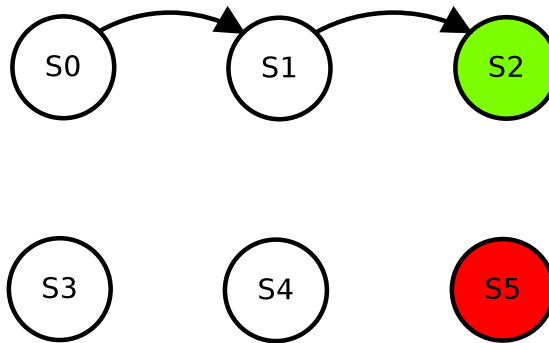


Obr. 3.2: Inicializácia

Agent v každom stave náhodne vyberá akcie (na výbere nezáleží, dôležité je aby každá akcia mala nenulovú pravdepodobnosť výberu, a rovnako bola nenulová pravdepodobnosť dosiahnutia ľubovoľného stavu). Obrázky Obr. 3.3 a Obr. 3.4 ilustrujú jednu z možných ciest.



Obr. 3.3: Prechod do stavu S1



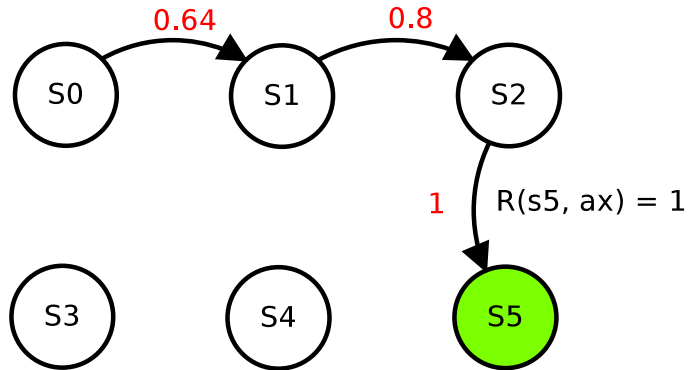
Obr. 3.4: Prechod do stavu S2

Po dosiahnutí cieľového stavu Obr. 3.5 je na základe 3.4 možné spočítať podľa 3.2 ohodnotenia doteraz vykonaných akcií - agent získal nenulovú odmenu $R(s_5, a_x) = 1$ (kde a_x značí ľubovoľnú akciu), ktorú rekurentne spočíta pre všetky doteraz vykonané akcie.

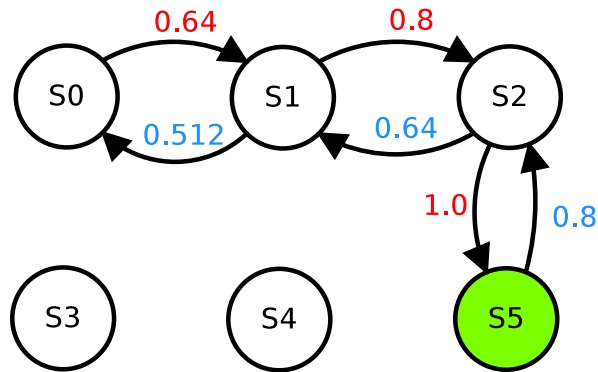
Pre zjednodušenú variantu 3.2 by bolo možné pamätať si len jeden predošlý stav a nepostupovať v ohodnocovaní rekurentne. V praktickej aplikácii je potrebné obmedziť hĺbku rekurzie, a pamätať si len posledných P stavov a vnich vykonaných rozhodnutiach. V tomto jednoduchom príklade však nie sú nutné tieto obmedzenia, je teda možné pamätať si celú cestu.

Agent môže pokračovať v ceste ďalej, napr. späť 3.6 a približne počítat' ohodnotenia. Prechod z s_5 do s_2 je ohodnotený ako 0.8 - vybralo sa najlepšie možné ohodnotenie ako sa dostať do s_5 (1) násobené γ , $R(s_2, a_x) = 0$ (podľa 3.4).

Po prejdení celého grafu, kedy agent vykonal všetky možné akcie dosiahne funkcia $Q(s(n), a(n))$ konečný, ustálený stav Obr. 3.7, teda



Obr. 3.5: Prechod do stavu S3



Obr. 3.6: Ďalšie prechody agenta

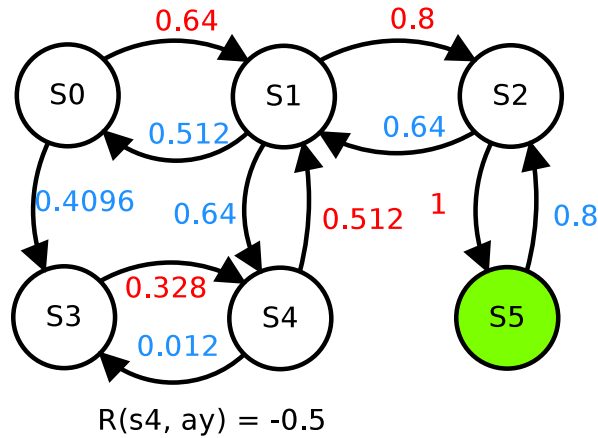
$$\forall s(n), \forall a(n), \forall \varepsilon > 0 \exists Q_n : |Q_n(s(n), a(n)) - Q_{n-1}(s(n), a(n))| < \varepsilon \quad (3.5)$$

hodnoty Q-funkcie sa teda pri pevne danom $R(s(n), a(n))$ už nemenia.

3.2 Výber akcie

Pre nájdenie konečných hodnôt funkcie ohodnotení podľa 3.5 stačí, aby každý prechod mal nenulovú pravdepodobnosť vykonania. Pre ďalšie vyšetřovanie konania agenta je daná pravdepodobnosť výberu akcie ako

$$P(s(n), a(s)) = \frac{e^{kQ(s(n), a(s))}}{\sum_{i=1}^{C_a} e^{kQ(s(n), a(i))}} \quad (3.6)$$



Obr. 3.7: Konečný stav

kde

s je zvolená akcia

C_a je počet akcií

k je konštanta a platí $k \geq 0$

agent ktorý vyberá všetky akcie s rovnakou pravdepodobnosťou má teda $k = 0$. Pre vysoké hodnoty k : $\lim_{k \rightarrow \infty}$ bude agent vyberať len najlepšie dostupné akcie. Pre učenie agenta je teda vhodné zvoliť malé k .

Je možné definovať ľubovoľné iné možnosti výberu akcie, napr. uprednostňovať menej často vykonané akcie, prípadne podľa zmeny $|Q_n(s(n), a(n)) - Q_{n-1}(s(n), a(n))|$ uprednostňovať prechody s veľkou hodnotou zmeny. Metóda výberu akcie nie je pre konečný výsledok $Q(s(n), a(n))$ v prípade Q-learning algoritmu dôležitá, preto sa jej práca nevenuje. V prípade požiadavku na učenie agenta v reálnom čase má kl'účový význam práve spôsob výberu akcie.

3.3 Problémy výpočtu $Q(s, a)$

Algoritmus je definovaný pre diskretnú množinu stavov. Ďalej sa predpokladá, že $s(n) \in \langle -1, 1 \rangle$ a podobne $a(n) \in \langle -1, 1 \rangle$

Pre počty prvkov stavového vektora a vektora akcií (n_s, n_a) je možné definovať delenie ich hodnôt na diskretný počet d_s a d_a , potom je možné vyjadriť celkový počet hodnôt $Q(s(n), a(n))$ ako

$$C = d_s^{n_s} d_a^{n_a} \quad (3.7)$$

Samotný počet hodnôt ktoré treba spočítať teda exponenciálne narastá s rastom počtu prvkov stavového vektora a vektora akcií.

Pre úlohy kde do systému vstupuje mnoho nezávislých vstupov sa stáva implementácia $Q(s(n), a(n))$ problémom najmä z dôvodov :

- veľké pamäťové nároky
- o nenavštívených prechodoch nevie agent povedať nič

Vhodným riešením sa ukazuje aproximácia Q-funkcie. Nech je aproximovaná funkcia označená $Q'_n(s(n), a(n))$ a presné riešenie ako $Q_n(s(n), a(n))$.

Dané sú postuláty o tejto aproximácii

Postulát 1 *Neobmedzená prenosť aproximácie* : Pre všetky stavy $s(n)$ a akcie $a(n)$ musí platiť $|Q_n(s(n), a(n)) - Q'_n(s(n), a(n))| < \varepsilon$. Kde $\varepsilon > 0$ a určuje kvalitu aproximácie. Zlepšením vlastností $Q'_n(s(n), a(n))$ je možné ľubovoľne znižovať ε .

Postulát 2 *Lokálna zmena* : Lokálna zmena hodnoty $\delta = |Q'_n(s(n), a(n)) - Q'_{n-1}(s(n), a(n))|$ neovplyvní hodnotu funkcie v inom bode o viac ako $\forall s(n') \forall a(n'), n \neq n' : \delta < \kappa$. Znižovaním hodnoty κ sa funkcia stáva menej závislá na okolí bodu $[s(n), a(n)]$.

Funkciu $Q(s(n), a(n))$ je možné aproximovať niekoľkými spôsobmi. Tie najbežnejšie sú

- tabuľka
- neurónová sieť
 - dopredná neurónová sieť
 - Kohonenova mapa
 - neurónová sieť báziických funkcií

3.4 Tabuľka

Priamočiarím prístupom je ukladať hodnoty $Q(s(n), a(n))$ do tabuľky. Pre diskretnú a konečnú množinu akcií možno tabuľku rozdeliť na viac častí, čím sa urýchli vyhľadávanie. Počet stavov v reálnych aplikáciach však môže byť vysoký (vo všeobecnosti tvoria stavy spojitú množinu).

Stav agenta aj vykonaná akcia sú obvykle vektory normované do $\langle -1, 1 \rangle$, pre implementáciu tabuľky je nutné ich prepočítat na celočíselné indexy

$$I_s(n) = \lceil \sum_{i=1}^{n_s} \left(d_s \frac{s_i(n) + 1}{2} \right)^i \rceil \quad (3.8)$$

$$I_a(n) = \lceil \sum_{i=1}^{n_a} \left(d_a \frac{a_i(n) + 1}{2} \right)^i \rceil \quad (3.9)$$

kde

$I_s(n)$ je index stavu

$I_a(n)$ je index akcie

$s_i(n)$ je i -ty prvok vektora stavu $s(n)$

$a_i(n)$ je i -ty prvok vektora akcií $a(n)$

Pre diskretný počet stavov a akcií je možné definovať tabuľkovú interpretáciu ako $Q^t(I_s(n), I_a(n))$. Pre $\lim_{d_s \rightarrow \infty}$ a $\lim_{d_a \rightarrow \infty}$ je možné považovať tabuľku za presné riešenie pretože spĺňa postuláty 1 aj 2.

3.5 Dopredná neurónová sieť

Pre aproximáciu funkcie ohodnotení je možné použiť dobrednú neurónovú sieť ako univerzálny aproximátor. Podľa Kolmogorovho teorému je možné neurónovú sieť takto použiť [24], [25] a [26]. Samotný teorém však nerieši problém učenia takejto siete. Na učenie siete existuje niekoľko algoritmov, medzi najčastejšie patria [27], [28], [29] :

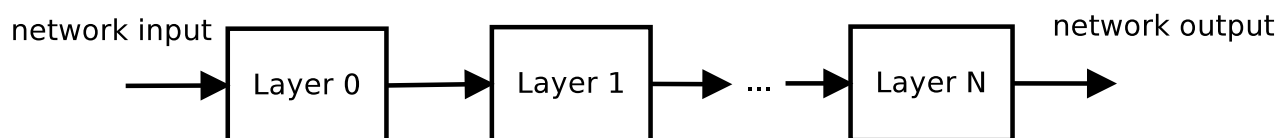
1. Backpropagation

2. Genetické algoritmy

3. Simulované žihanie

Najpoužívanejší Backpropagation má množstvo problémov, najmä postupný pokles gradientu v hlbokých neurónových sieťach. Kedy sú vnorené vrstvy prakticky neučené. Ďalší závažný problém je uviaznutie v lokálnom minime, čo sa autori snažia vyriešiť zavedným zotrvačností do zmeny váh siete. Istým nádejším východiskom sa zdá byť simulované žihanie [30]. Pre problém Q-learning algoritmu je však potrebný prístup ktorý postupne zlepšuje riešenie, a to je jedine niektorá z gradientových metód.

Samotná dopredná sieť je tvorená niekoľkými prepojenými vrstvami neurónov 3.8. Pre popis vrstvy je však vhodné použiť maticový zápis, nakoľko prenos neurónu má obvykle vo vrstve rovnakú aktivačnú funkciu.



Obr. 3.8: Dopredná neuronová sieť

Správanie sa jednej vrstvy je možné popísať ako funkciu vstupného vektora, ktorej výstupom je tiež vektor (počty prvkov týchto vektorov však môžu byť rôzne, vždy však konečné). Je daný vstupný vektor

$$I(n) = (s(n), a(n)) \quad (3.10)$$

výstupná hodnota neurónovej siete ako $y_{nn}(n)$ a požadovaná hodnota ako $y_r(n)$.

Ďalej je definovaná chyba ako

$$e(n) = y_r(n) - y_{nn}(n) \quad (3.11)$$

Vrstva l doprednej siete je definovaná ako

$$y^l(n) = f^l \left(W^l(n) I^l(n) \right) = f^l \left(\begin{pmatrix} w_{1,1}^l(n) & w_{1,2}^l(n) & \cdots & w_{1,n'}^l(n) \\ w_{2,1}^l(n) & w_{2,2}^l(n) & \cdots & w_{2,n'}^l(n) \\ \vdots & \vdots & \ddots & \vdots \\ w_{m',1}^l(n) & w_{m',2}^l(n) & \cdots & w_{m',n'}^l(n) \end{pmatrix} \begin{pmatrix} i_{1,1}^l(n) & i_{1,2}^l(n) & \cdots & i_{1,n'}^l(n) \end{pmatrix} \right) \quad (3.12)$$

kde

n' je počet prvkov vstupného vektora

m' je počet prvkov výstupného vektora

$f(X)$ je aktivačná funkcia

$W^l(n)$ je matica váh.

Najčastejšie používané aktivačné funkcie sú sigmoida, hyperbolický tangens, lineárna, usmerňovač a skoková funkcia. Ich predpisy sú

$$y_1(x) = \frac{1}{1 + e^{-x}} \quad (3.13)$$

$$y_2(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3.14)$$

$$y_3(x) = x \quad (3.15)$$

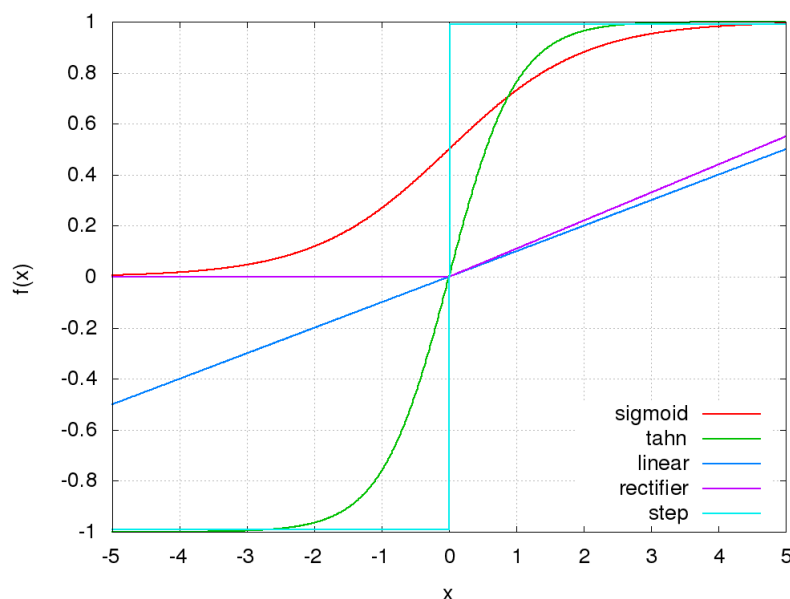
$$y_3(x) = \begin{cases} x & \text{ak } x > 0 \\ 0 & \text{inak} \end{cases} \quad (3.16)$$

$$y_4(x) = \begin{cases} 1 & \text{ak } x > 0 \\ -1 & \text{inak} \end{cases} \quad (3.17)$$

Ich priebehy sú znázornené na obrázku 3.9.

Zoradením niekoľkých vrstiev za sebou, tak že výstup predošlej je vstupom do aktuálnej vrstvy je možné získať doprednú neurónovú sieť. Takáto sieť je vhodná na riešenie klasifikačných aj aproximačných problémov.

Dopredná sieť je známa nelokálnosťou učenia: pri trénovaní na podmnožinu množiny požadovaných výstupov sa mení hodnota aj mimo túto podmnožinu. Sieť teda veľmi proble-



Obr. 3.9: Grafické znázornenie priebehov aktivačných funkcií

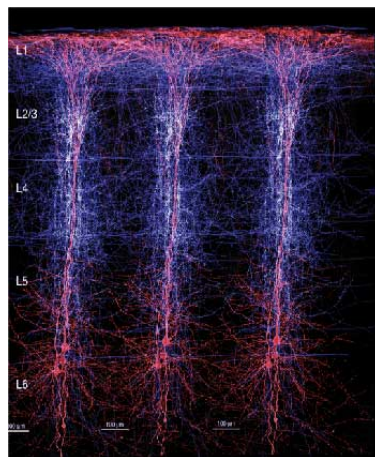
maticky spĺňa postulát 2. Dobré však spĺňa postulát 1, vhodnou voľbou počtu vrstiev a počtu neurónov je možné dosiahnuť ľubovoľnú presnosť aproximácie.

3.6 Kohonenova neurónová sieť

Kohonenova neurónová sieť je inšpirovaná rovinnými štruktúrami v mozgovej kôre, kde výrazne dominujú prepojenia v rámci vrstvy a prepojenia medzi vrstvami je podstatne menej 3.10. Napriek týmto rovinatým štruktúram, mozog bežne spracúva mnohodomenzionálne problémy - vstupom sú podnety z miliónov receptorov.

Pôvodný návrh Kohonenovej siete neriešil aproximáciu funkcie, ale klasifikáciu vstupov do zhlukov. Pričom podobné vstupy patria do rovnakého zhluku. Počet zhlukov je voliteľný a je rovný počtu použitých neurónov. Takáto sieť sa vie učiť bez definovania chyby, t.j. nie je potrebné stanoviť do ktorého zhluku dáta patria. Samotné zhluky sa utvárajú postupne, tak ako sú do siete predkladané dáta.

Každému zhluku je však možné priradiť požadovanú výstupnú hodnotu. Sieť tak môže pracovať podobne ako tabuľka - vstupné dáta priradí do najbližšieho zhluku a výstupom je hodnota prisluchajúca k tomuto zhluku. V tomto prípade už treba stanoviť chybu, ako rozdiel



Obr. 3.10: Neokortexový stĺpec

požadovanej hodnoty a skutočnej hodnoty výstupu siete. Samotné zhľuky sa však tvoria bez učiteľa. Algoritmus má niekoľko krokov :

Najpr sa spočítajú vzdialenosti od vstupného vektora

$$d_j(n) = \sum_{i=1}^N (I_i(n) - w_{ji}(n))^2 \quad (3.18)$$

kde $w(n) \in \mathbb{R}$ je matica váh, na začiatku sa volí náhodná, tak aby rovnomerne pokrila stavový priestor vstupných veličín (Obvykle sa používa rovnomerné rozdelenie. Štatistickou analýzou vstupných dát je ale možné určiť iné, vhodnejšie a urýchliť tak konvergenciu hodnôt váh).

Výťazný neurón v je definovaný ako

$$v : \forall j : d_v(n) \leq d_j(n) \quad (3.19)$$

je to neurón ktorý má najmenšiu vzdialenosť od predloženého vstupu.

A pre každý neurón existuje priradená výstupná hodnota $y_j(n) \in \mathbb{R}$, výstupom siete je teda hodnota priradená výťaznému neurónu $y_{nn}(n) = y_v(n)$.

Učenie siete prebieha v dvoch krokoch

1) **zmena váh** $w(n)$ zmenia sa váhy výťazného neurónu, pretože najlepšie zodpovedajú požadovaným váham tak že sa priblížia hodnote vstupného vektora

$$w_{ji}(n+1) = (1 - \eta_1(j))w_{ji}(n) + \eta_1 I_i(n) \quad (3.20)$$

kde $\eta_1(j) \in (0, 1)$ je krok učenia a závisí od polohy neurónu v sieti. V najjednoduchšom prípade

$$\eta_1(j) = \begin{cases} \eta & \text{ak } j = v \\ 0 & \text{inak} \end{cases} \quad (3.21)$$

k zmene váh teda dôjde len pri víťaznom neuróne. Ďalší často používaný tvar funkcie postupne zmenšuje hodnotu $\eta_1(j)$ podľa $d_j(n)$ a to ako

$$\eta_1(j, n) = \eta e^{-kd_j(n)} \quad (3.22)$$

kde $k \in (0, \infty)$. Krok učenia $\eta_1(j, n)$ je teda premenný a závisí aj od predloženého vzoru podľa n .

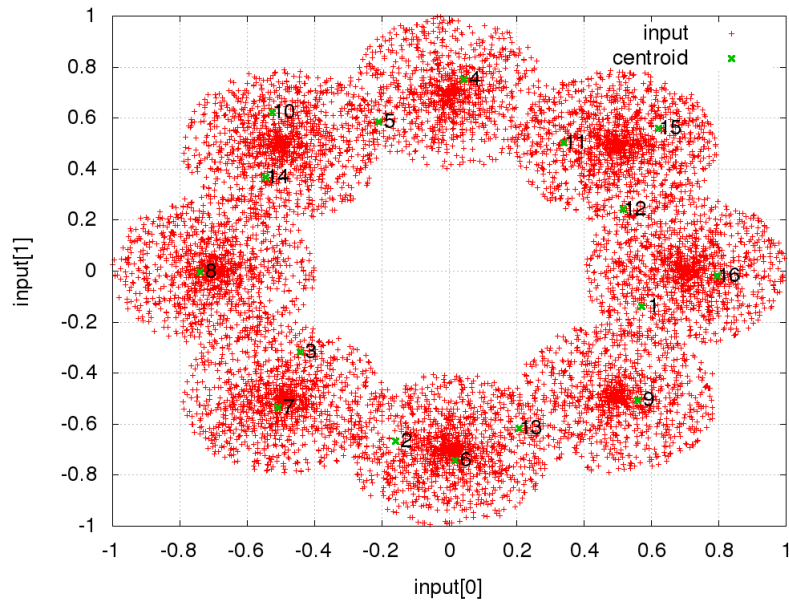
Po dostatočnom počte iterácií sa hodnoty váh ustália na hodnotách tak aby rozdelili množinu vstupných vektorov na lokálne oblasti. Tento stav je znázornený na 3.11. Vstupný proces generoval 8 zhlukov dát ktoré sieť klasifikovala použitím 16 neurónov. Každému neurónu je možné priradiť požadovanú hodnotu výstupu.

2) upraví sa výstupná hodnota $y_v(n)$

$$y_v(n+1) = y_v(n) + \eta_2 e(n) \quad (3.23)$$

kde $e(n)$ je chyba podľa 3.11. Takáto Kohonenová sieť má veľmi dobré predpoklady na aprixmáciu funkcie, ktorá nadobúda málo hodnôt, predkladaním vzorov z blízkosti jedného zhuku (dátá majú malú vzdialenosť v zmysle 3.18) sa dá očakávať zmena parametrov len jedného neurónu. Sieť teda dobre spĺňa predpoklady 1, 2. Počet neurónov však musí rásť do nekonečna.

Je vhodné poznamenať, že výstupná hodnota nemusí byť len číselná hodnota, ale aj funkcia, ktorej vstupom je opäť $I_i(n)$. Výber víťazného neurónu tak funguje ako výber vhodnej funkcie - prepínač.



Obr. 3.11: Znázornenie váhových parametrov w pre dvojrozmerný priestor

3.7 Neurónová sieť bázických funkcií

Samotný prenos neurónu nemusí byť obmedzený len na množinu funkcií z 3.17. Vhodnú funkciu je možné zmenou parametrov upraviť do tvaru, aby na zvolený vstup $I_0(n)$ dosahovala požadovanú hodnotu a postupným zväčšovaním vzdialenosti $|I_0(n) - I_i(n)|$ klesala jej hodnota k nule.

Najjednoduchším príkladom takýchto funkcií je

$$f_j(X(n)) = \begin{cases} k_j & \text{ak } X(n) = X_0^j \\ 0 & \text{inak} \end{cases} \quad (3.24)$$

kde k_j je hodnota požadovaná v bode X_0^j . Výstupom siete potom je

$$y(X) = \sum_{j=1} f_j(X(n)) \quad (3.25)$$

Z charakteru Q-learning algoritmu majú hodnoty $Q(s(n), a(n))$ charakter postupne klesajúcich hodnôt. Je teda potrebné vybrať iné funkcie.

Nasledujú preto definície funkcií s ktorými boli urobené experimenty.

Dané sú bázické funkcie $f_j^x(s(n), a(n))$, kde x je typ bázickej funkcie. Požadovaná hodnota $Q^x(s(n), a(n))$ je potom lineárnou kombináciou týchto funkcií typu x .

Z charakteru Q-learning algoritmu 3.2 je možné určiť požiadavky na tieto funkcie :

1. predpis 3.2 je tvorený klesajúcou exponenciálou - podobný charakter by mala mať aj bázická funkcia
2. existencia jedného globálneho maxima a zmenou parametrov určovať polohu tohto bodu
3. možnosť ľubovoľne meniť strmosť funkcie v okolí maxima
4. funkcia by mala byť zhora aj z dola ohraničená

Cieľom je mať možnosť nezávisle nastaviť maximá funkcií do oblastí, ktoré zodpovedajú nenulovým hodnotám $R(s(n), a(n))$ - bod 2. Ak ohodnotenie spĺňa podmienku najlepšej možnej akcie v danom stave, dá sa očakávať že bude mať menšiu strmosť, naopak, ak funkcia popisuje bod kde $R(s(n), a(n))$ dosahuje malé hodnoty (obvykle záporné), bude požadovaná vysoká strmosť tejto funkcie - obe požiadavky sú zhrnuté v bode 3. Bod 4 umožňuje rozumne ohraničiť rozsah funkcie.

Niektoré tvary bázických funkcií ktoré možno uvažovať pre problém aproximácie

$$f_j^1(s(n), a(n)) = e^{-\sum_{i=1}^{n_s} \beta_{aji}(n)(s_i(n) - \alpha_{aji}(n))^2} \quad (3.26)$$

$$f_j^2(s(n), a(n)) = \frac{1}{1 + \sum_{i=1}^{n_s} \beta_{aji}(n)(s_i(n) - \alpha_{aji}(n))^2} \quad (3.27)$$

$$f_j^3(s(n), a(n)) = e^{-\sum_{i=1}^{n_s} \beta_{aji}(n)|s_i(n) - \alpha_{aji}(n)|} \quad (3.28)$$

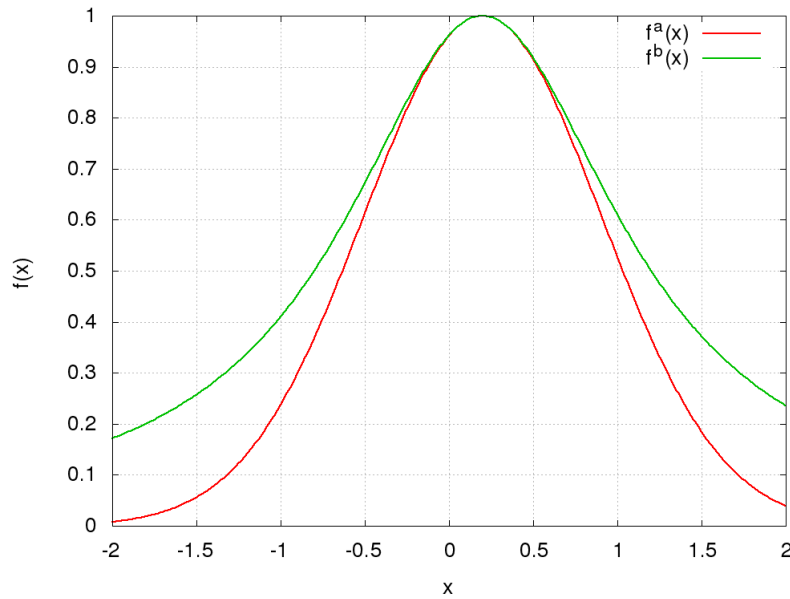
kde

$\alpha_{aji}(n) \in \langle -1, 1 \rangle$ určuje polohu maxima funkcie

$\beta_{aji}(n) \in (0, \infty)$ určuje strmosť funkcie.

Ich priebehy pre prvé dve uvedené sú na obrázku 3.12.

Pre symetrické prechody medzi stavmi ich možno zjednodušiť na



Obr. 3.12: Znážornenie priebehov bázických funkcií

$$f_j^1(s(n), a(n)) = e^{-\beta_{aj} \sum_{i=1}^{n_s} (s_i(n) - \alpha_{aji})^2} \quad (3.29)$$

$$f_j^2(s(n), a(n)) = \frac{1}{1 + \beta_{aj} \sum_{i=1}^{n_s} (s_i(n) - \alpha_{aji})^2} \quad (3.30)$$

$$f_j^3(s(n), a(n)) = e^{-\beta_{aj} \sum_{i=1}^{n_s} |s_i(n) - \alpha_{aji}|} \quad (3.31)$$

Aproximovaná funkcia ohodnotení pre l bázických funkcií je potom

$$Q^x(s(n), a(n)) = \sum_{j=1}^l w(n)_j^x f_j^x(s(n), a(n)) \quad (3.32)$$

kde $w(n)_j^x$ sú váhy bázických funkcií.

Je teda potrebné stanoviť celkovo 3 sady parametrov : α β w .

3.7.1 Určenie parametrov α

Parameter α určuje posunutie maxima funkcie a postupuje sa podobne ako v prípade 3.20. Treba zohľadniť fakt, že pre konečný výsledok je dôležité pokryť všetky oblasti s nenulovým $R(s(n), a(n))$, vrchol krivky bude ležať nad bodom $[s(n), a(n)]$.

Zmena parametrov α prebieha v piatich krokoch.

- na začiatku sa zvolia $\alpha_{jia}(n)$ náhodne, ze $\langle -1, 1 \rangle$
- spočítajú sa vzdialenosti od predloženého vstupu $d_{ja}(n) = |s(n) - \alpha_{ja}(n)|$
- nájde sa také ka kde pre $\forall j : d_{ka}(n) \leq d_{ja}(n)$
- spočíta sa krok učenia $\eta'_a(n) = \eta_1 |Q_r(s(n), a(n))|$
- upravia sa parametre $\alpha_{aki}(n+1) = (1 - \eta')\alpha_{aki}(n) + \eta' s_i(n)$

kde

$Q_r(s(n), a(n))$ je požadovaný výstup

η_1 je konštanta učenia

Krok učenia teda závisí od veľkosti požadovanej hodnoty, tým sa zabezpečí aby maximum krivky naozaj ležalo nad bodom $[s(n), a(n)]$.

3.7.2 Určenie parametrov β

Parameter β určuje strmosť krivky. Ak boli k dizpozícii naraz všetky požadované výstupy, bolo by možné spočítať tento parameter z rozptylu. Požadované hodnoty však prichádzajú postupne, strmosť krivky sa preto upravuje priebežne, podľa toho či požadovaná hodnota leží nad, alebo pod krivkou.

- stanoví sa chyba $e(n) = Q_r(s(n), a(n)) - Q(s(n), a(n))$
- pre každú bázičku funkciu $\beta_{ja}(n+1) = \beta_{ja}(n) + \eta_2 e(n) w_{ja}(n)$
- skontroluje sa $\beta_{ja}(n) \in (0, \infty)$

kde

$Q_r(s(n), a(n))$ je požadovaný výstup

η_2 je konštanta učenia

3.7.3 Určenie váhových parametrov w

Nakoniec sa gradientovou metódou určia váhové parametre. Pre presné riešenie by bolo možné použiť metódu najmenších štvorcov, tá je však pre veľký počet bázcikých funkcií ťažko vypočítateľná. Zmena parametrov je potom daná nasledujúcim postupom

- stanoví sa chyba $e(n) = Q_r(s(n), a(n)) - Q(s(n), a(n))$
- pre každé $w_{ja} : w_{ja}(n+1) = w_{ja}(n) + \eta_3 e(n) y_j(n)$
- skontroluje sa $w_{ja}(n) \in (-r, r)$

kde

η_3 je konštanta učenia

r je maximálny rozsah váh

3.7.4 Hybridný variant

Ak by funkcia $R(s(n), a(n))$ mala len jednu kladnú hodnotu a ostatné by boli nulové, aproximáciu $Q(s(n), a(n))$ by veľmi dobre popísala Gaussova krivka 3.31. Ak by funkcia $R(s(n), a(n))$ mala len záporné hodnoty a ostatné by boli nulové, funkcia $Q(s(n), a(n))$ by s ohľadnutím na 3.2 si boli rovné. Vo funkcií $Q(s(n), a(n))$ by sa tak objavilo niekoľko záporných hodnôt, ostro ohraničených.

Vyjduc z týchto úvah, je možné skombinovať výhody oboch : Gaussova krivka ktorá dokáže pokryť nenulovými hodnotami celý definyčný obor a má možnosť tak šíriť hodnoty Q na ďalšie stavy a funkcie 3.24. Funkcia 3.24 predstavuje vlastne tabuľku, ktorá nadobúda nenulové hodnoty vo vybraných bodoch - tvorí tak adaptívnu tabuľku.

Je teda možné skombinovať funkciu 3.24 s niektorou z 3.31, čo vedie na vzťahy

$$P_i(s(n), a(n)) = \begin{cases} r_{ai} & \text{if } s(n) = \alpha_i^1 \\ 0 & \text{inak} \end{cases} \quad (3.33)$$

$$H_j(s(n), a(n)) = w_{aj} e^{-\beta_{aj} \sum_{i=1}^{n_s} (s_i(n) - \alpha_{aji}^2)^2} \quad (3.34)$$

$$Q(s(n), a(n)) = \sum_{i=1}^I P_i(s(n), a(n)) + \sum_{j=1}^J H_j(s(n), a(n)) \quad (3.35)$$

kde

α_j^1 sú oblasti kde $H_j(s(n))$ nadobúda nenulové hodnoty

α_j^2 sú oblasti pre ktoré $f_j(s(n), a(n))$ nadobúda maximum

r_{ai} je hodnota zápornej odmeny $R(s(n), a(n))$

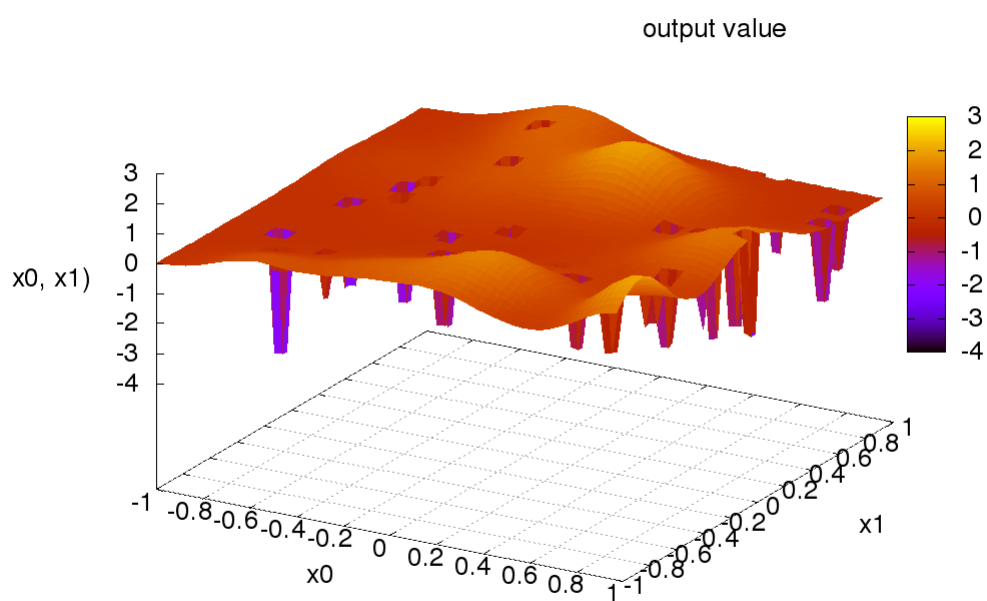
w_{aj} je váha a zobovedá veľkosti maxima resp. minima pre fukciu

β_{aj} je strmosť, a platí $\beta > 0$

I a J sú počty bázičkých funkcií

Označenia P a H vznikli z tvaru funkcií : peak a hill. Funkcia bude na ďalších grafoch označená ako Gauss + AT : kombinácia Gaussovej krivky a adaptívnej tabuľky. Mechanizmus učenia zostáva rovnaký ako pre bázičké funkcie v predošlej časti. Ukážka priebehu funkcie pre dve premenné je na obrázku 3.13. Počet funkcií $P_i(s(n), a(n))$ bol zvolený 30 a počet funkcií $H_j(s(n), a(n))$ 20. Pre názornosť boli parametre r_{ai} zvolené záporné a parametre β_{aj} kladné.

Funkcia predstavuje nový tvar bázičkých funkcií pre aproximovanie funkcie ohodnotení $Q(s(n), a(n))$.



Obr. 3.13: Znázornenie predmetnej funkcie

Kapitola 4

Experimentálna časť

4.1 Ciele experimentu

V oblasti Q-learning algoritmov je možné pozorovať dva hlavné smery výskumu

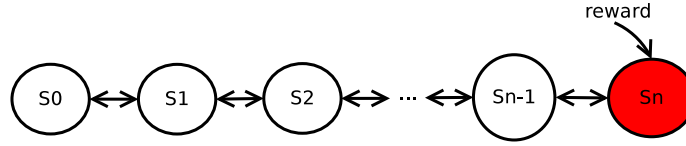
- aproximácia funkcie ohodnotení [31] [32] [33] [34]
- spôsob výberu akcie [35] [36] [37]

Obe majú široké pole diskusií v snahe vyriešiť niekoľko hlavných problémov Q-learning algoritmu a to najmä :

- veľký počet prechodov medzi stavmi
- malá zmena vo výpočte $Q(s(n), a(n))$ môže spôsobiť veľké zmeny v stratégii.

Cieľom práce je na danej množine odmeňovacích funkcií $R(s(n), a(n))$ overiť možnosti aproximácie $Q(s(n), a(n))$. Prvým intuitívnym spôsobom bola snaha aproximovať predmetnú funkciu doprednými neurónovými sieťami. Principiálne tomu nič nebráni, problém je ale nedokonalý algoritmus učenia, a to, že sa vplyvom rekurentnej povahy Q-learning algoritmu pokúša neurónová sieť zároveň predikovať správnu hodnotu a zároveň učiť na požadovanú hodnotu.

Postupne sa tak v sieti nabaľuje chyba. Tento problém ilustruje 4.1. Je daná postupnosť stavov a každom okrem východzieho a cieľového sú dve akcie. Odmena $R(s, a)$ je všade nulová, len po dosiahnutí cieľového stavu je rovná kladnej hodnote.



Obr. 4.1: Ilustrácia postupného nabal'ovania chyby

Pre korektné vyplnenie hodnôt v s_{n-1} sa vyžaduje korektná hodnota v s_n

$$Q(s(1), a(1)) = R(s(1), a(1)) + \gamma \max_{a(0) \in \mathbb{A}} Q(s(0), a(0))$$

$$Q(s(2), a(2)) = R(s(2), a(2)) + \gamma \max_{a(1) \in \mathbb{A}} Q(s(1), a(1))$$

...

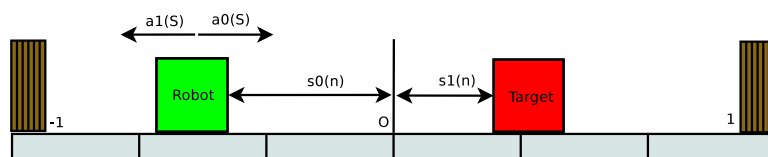
V prípade doprednej siete učenej algoritmom Backpropagation, zmena hodnoty v jednom bode $Q(s(n), a(n))$ spôsobí zmenu vo všetkých ostatných hodnotách a nikde nie je zaručené, že k správnej hodnote - v určitom štádiu učenia sa tak môže zdať, že hodnoty korektné konvergujú, a inom sa môžu vzd'alať. Práve preto sa pre klasické úlohy rozpoznávania predkladajú sieti vzory v náhodnom poradí a v mnohých opakovaniach. Vzory a požadované výstupy sú však nezávislé.

4.1.1 Divergencia riešenia

Tento efekt divergencie bol pozorovaný nie len vyššie uvedenými autormi, ale aj experimentálne overený v tejto práci. Usporiadanie experimentu je na obrázku 4.2. Robot má dve akcie, pohyb o pevne zvolený krkok vľavo alebo vpravo. Úlohou je dostať sa do cieľa, ktorý môže byť umiestnený kdekoľvek. Pre jednoduchosť bol vybraný dvoj rozmerný stavový priestor z rozsahu $s \in \langle -1, 1 \rangle$. Stav systému charakterizovaný vektorom s je poloha robota voči počiatku a poloha cieľa voči počiatku, takýto systém je aj dobre graficky znázorniteľný.

Z ostatných parametrov ktoré boli použité pre beh experimentu :

- počet iterácií = 10000000
- delenie stavového priestoru = 1/8.0



Obr. 4.2: Schéma experimentu pre doprednú neurónovú sieť

- $\gamma = 0.7$
- neurónová sieť :
 - počet skrytých vrstiev = 2
 - počet neurónov v skrytých vrstvách = 10
 - rozšaha váh = 4.0
 - krok učenia $\eta = 0.001$

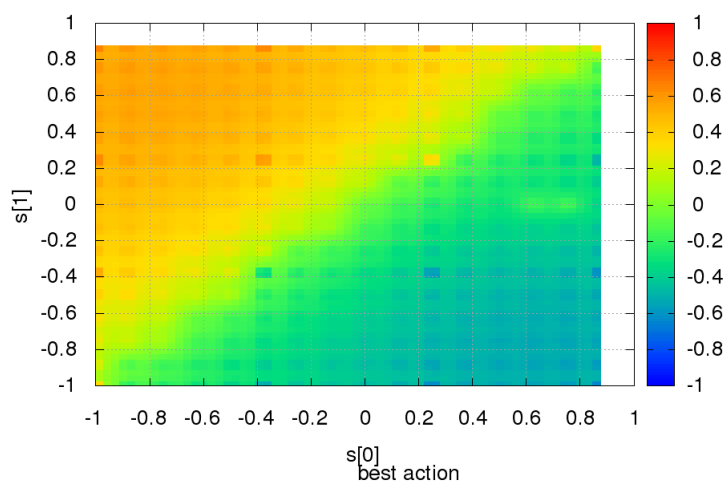
Najskôr bolo určené riešenie použitím tabuľky (ktoré bolo pre malý počet stavov možné spočítať). Najdôležitejší výstup je výber korektnej akcie, kde $+1$ znamená jeden smer a -1 smer opačný. Veľmi ľahko sa dá očakávať ostré rozdelenie stavového priestoru po diagonále : ak je robot naľavo od cieľa musí sa pohybovať doprava a naopak. Výsledok je na obrázku 4.5. Pre úplnosť, obrázok 4.6 znázorňuje hodnoty $\max_{a(n-1) \in \mathbb{A}} Q(s, a)$. Opäť sa dá ľahko očakávať že pre najmenšiu vzdialenosť bude táto hodnota najväčšia - hodnoty na diagonále.

Jedno z najlepších riešení dosiahnuté doprednou neurónovou sieťou učenou Backpropagation algoritmom je na obrázkoch 4.5 a 4.6.

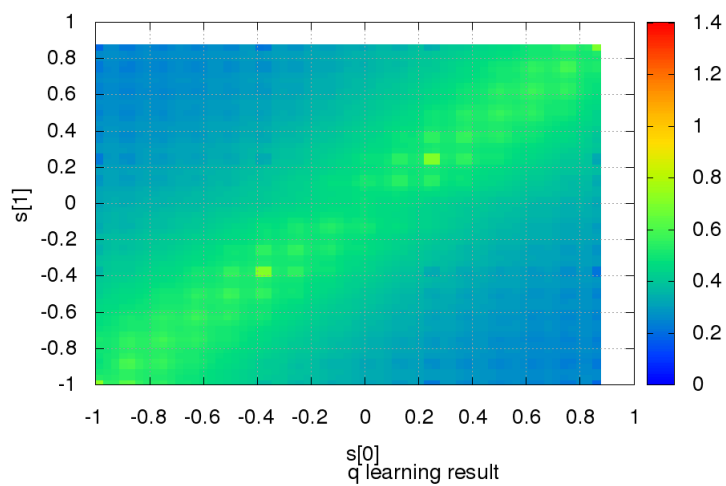
Napriek jednoduchej úlohe, nie je možné povedať že sieť úspešne aproximuje tento problém. Porovnaním výstupov najlepších akcií je možné vidieť určitý náznak podobnosti, ktorý je však vzhľadom na irelevantnosť úlohy bezpredmetný a dosahuje priveľkú chybu, najmä ak sa robot už blížil k cieľu.

Najlepšie výsledky dosahovala dopredná neurónová sieť s novo zavedeným modelom neurónu v tvare

$$y(x(n)) = \tanh\left(\sum_{i=0}^{N-1} w_i x_i(n) + \sum_{j=0}^{N-1} \sum_{i=0}^j v_{ij} x_i(n) x_j(n)\right) \quad (4.1)$$



Obr. 4.3: Najlepšia akcia pre riešenie s tabuľkou

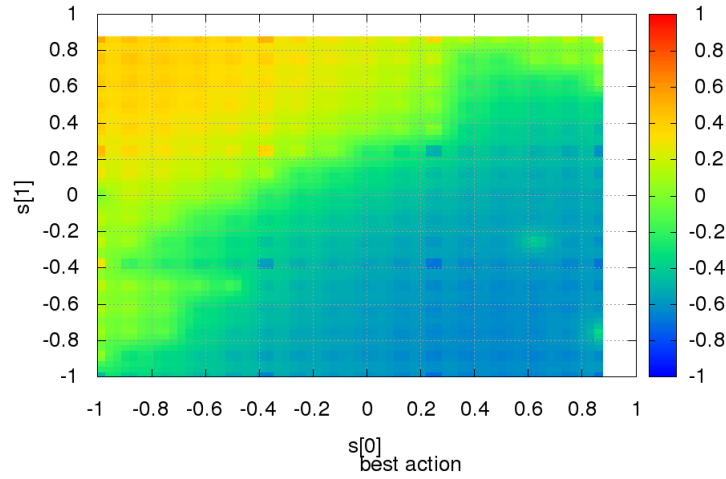


Obr. 4.4: Hodnoty $\max_{a(n-1) \in \mathbb{A}} Q(s, a)$ pre riešenie s tabuľkou

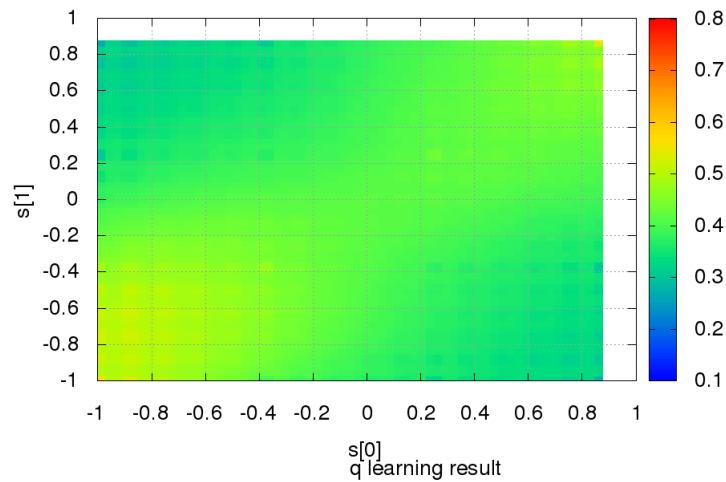
kde $x(n)$ je vstup do neurónu

w je vektor váh

v je matica váh



Obr. 4.5: Najlepšia akcia pre riešenie s neurónovou sieťou



Obr. 4.6: Hodnoty $\max_{a(n-1) \in \mathbb{A}} Q(s, a)$ pre riešenie s neurónovou sieťou

Takto definovaný model neurónu umožňuje okrem bežných funkcií McCulloh-Pittsovhého neurónu aj násobiť prvky vstupného vektora (časť $v_{ij}x_i(n)x_j(n)$). Dôsledkom toho je realizácia zložitých funkcií len s použitím jednej skrytej vrstvy - výrazne sa tak zjednoduší učenie. Medzi typické funkcie ktoré sa s McCulloh-Pittsovým neurónom a jendou skytou vrstvou ťažko

realizujú, možno uviesť napr : Fourierova transformácia, zmiešavanie signálov, riadenie toku dát na základe inej časti dát. Najmä posledne uvedená zvyšuje stupeň abstrakcie, kde neurónová sieť neaproximuje len jeden naučený druh funkcie, ale môže aproximovať viac, úplne rozdielných a medzi nimi vyberať. Uvedený model bol doteraz nepublikovaný v inej literatúre.

4.2 Riešenie aproximácie

Uvedení autori najčastejšie používajú tzv. príznaky (features) na aproximovanie $Q(s(n), a(n))$

$$Q(s(n), a(n)) = \sum_{j=1} w_j g_j(s(n), a(n)) \quad (4.2)$$

kde $g_j(s(n), a(n))$ sú funkcie príznakov, ktorých je konečný počet a w_j predstavuje váhy ich lineárnej kombinácie.

Príznaky sú funkcie, ktoré sú pevne zvolené a závisia od typu úlohy. Práve to predstavuje najväčší nedostatok. Cieľom navrhovaného experimentu je využiť príznaky ktorých parametre sa menia - bázické funkcie. Vzniká tak akýsi hybrid medzi neurónovou sieťou a lineárnou kombináciou pevne zvolených príznakov.

Z ohľadom na minimalizovanie vplyvu zmeny parametrov j -teho príznaku alebo váhy w_j na ostatné príznaky a váhy, je potrebné, aby ich bolo možné nastavovať nezávislé - aby vhodná séria príznakov pokryla svoju podmonožinu stavového priestoru. Toto je možné dosiahnuť ortogonalitou príznakou, stráca sa však možnosť generovať funkciu ako je lineárna kombinácia týchto ortogonálnych funkcií. Vhodným kompromisom sú preto funkcie uvedené v 3.28, alebo funkcia 3.7.4.

4.3 Návrh experimentu

V niekoľkých bodoch je možné postup určiť ako

- výber funkcií $R(s(n), a(n))$
- určenie presného riešenia, použitím tabuľky s veľkým počtom prvkov

- voľba aproximačnej metódy
- pre každú $R(s(n), a(n))$ spočítať niekoľko nezávislých behov
- výsledky porovnať s presným riešením, overiť a zosumarizovať

Funkcie $R(s(n), a(n))$ budú vybrané tak aby boli riedke a plne sa využil Q-learning - okamžité odmeny sú známe len v malom počte prípadov. Postupne sa obmenia pre rôzne počty nenulových prvkov.

Presné riešenie, aby bolo možné spočítať bude mať niekoľko tisíc diskretných stavov. Pre jednoduchosť, bude v každom stave rovnaká a presne definovaná množina akcií.

Vyberie sa niekoľko aproximačných metód, ktoré sa použijú na spočítanie $Q(s(n), a(n))$. Tu je nevyhnutné upozorniť na častú metodickú chybu : aj keď je možné $Q(s(n), a(n))$ spočítať presne, nesmie byť toto presné riešenie použité na stanovenie približného riešenia. Príkladom je dopredná neurónová sieť, ktorá sa dá veľmi ľahko natréňovať ak je množina požadovaných výstupov vopred známa. V prípade Q-learning algoritmu sa ale požadované hodnoty spočítavajú rekuretné, až počas behu.

Kedže voľba niektorých počiatočných parametrov aproximačných metód je náhodná, je nevyhnutné spočítať niekoľko nezávislých behov a overiť tak rozptyl, minimálnu, maximálnu a priemernu chybu.

Aby sa dalo kvalitatívne ohodnotiť použité riešenie, je nutné urobiť veľký počet experimentov. Aby bolo možné ľahko graficky znázorniť výsledok, bude stavový priestor dvojrozmerný a platí $s(n) \in \langle -1, 1 \rangle$. Agent si bude vyberať z pevne danej množiny akcií a bude sa tak v tomto priestore môcť pohybovať a to :

$$\mathbb{A} = [[0, 1], [0, -1], [1, 0], [-1, 0], [1, -1], [1, 1], [-1, -1], [-1, 1]]$$

prostredie umožní zmenu stavu vykonaním akcie $a(n) \in \mathbb{A}$, a to podľa

$$s(n+1) = s(n) + a(n)dt \quad (4.3)$$

Jednotlivé funkcie $R^k(s(n), a(n))$ predstavujú mapy odmien v ktorých sa agent pohybuje. Pre zjednodušenie bude platiť, že nezáleží ktorou akciou sa agent dostal do daného stavu - funkcia bude mať teda tvar $R^k(s(n))$ a predstavuje teda odmenu za to, že sa agent dostal na nejaké miesto.

Ako metódy aprximácie je zvolených 6 rôznych funkcií.

1. riedka tabuľka
2. Gaussova krivka $f_j^1(s(n), a(n))$ 3.31
3. Gaussova krivka $f_j^1(s(n), a(n))$ kombinovaná s riedkou tabuľkou
4. Modifikácia Kohonenovej neurónovej siete $f_j^2(s(n), a(n))$
5. Modifikácia Kohonenovej neurónovej siete $f_j^2(s(n), a(n))$ s riedkou tabuľkou
6. Guassova krivka a adaptívna tabuľka 3.7.4

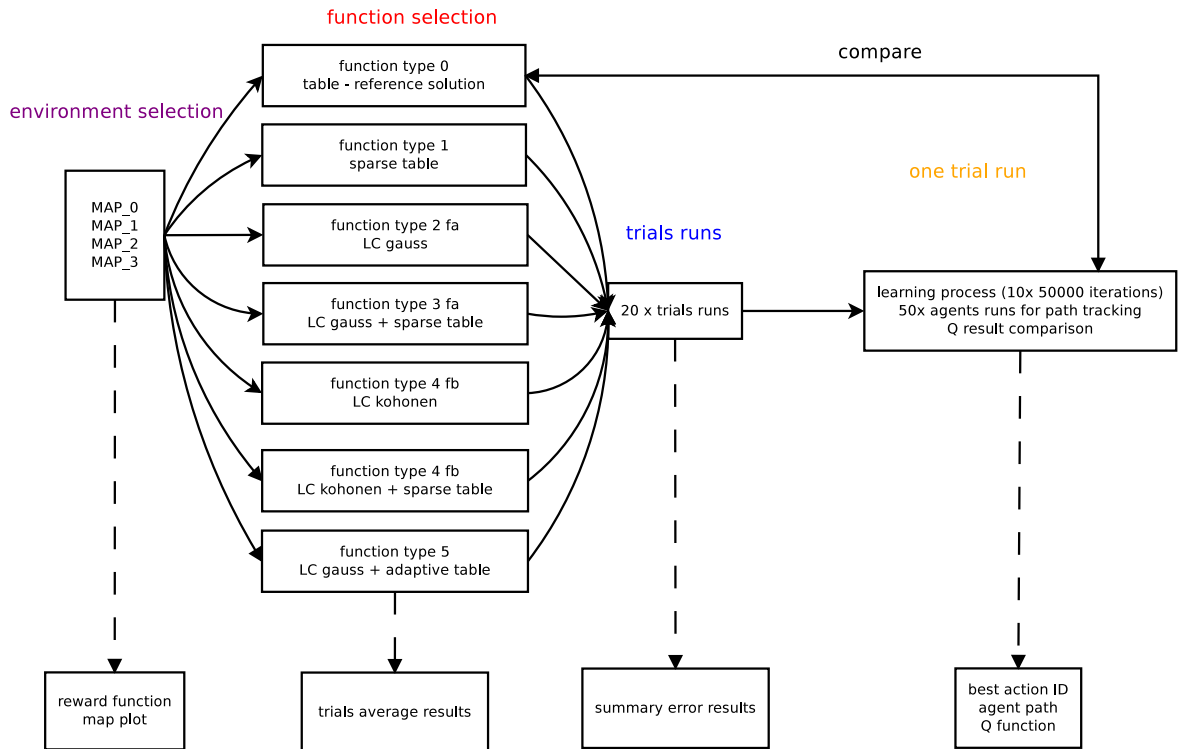
Pre každú z nich prebehne 20 trialov aby bolo možné urobiť štatistické vyhodnotenie. V každom trialu prebehne $10 \cdot 50000$ učiacich interácií aby bolo možné v 10 tich krokoch sledovať priebeh učenia. Na konci prebehne 50 behov agentov z náhodných východzieh stavov aby bolo možné sledovať ich cestu stavovým priestorom. Spolu teda prebehne 560 nezávislých experimentov a celkovo 280mil. behu algortimu.

Súhrnná schéma behu experimentov je na obrázku 4.7. Plné šípky predstavujú prepojenie úrovni metodológie. Čiarkované šípky znázorňujú výstupy v jednotlivých úrovniach. Presné riešenie je použité na porovnanie výslednej chyby.

- 50000 iterácií učenia
- rozmer s je $n_s = 2$, rozmer a je $n_a = 2$
- predpis funkcie ohodnotení

$$\begin{aligned}
 Q(s(n), a(n)) &= \\
 \alpha Q(s(n-1), a(n-1)) &+ \\
 (1 - \alpha)(R(s(n), a(n)) + \gamma \max_{a(n-1) \in \mathbb{A}} Q(s(n-1), a(n-1))) &
 \end{aligned}$$

- $R(s(n), a(n)) \in \langle -1, 1 \rangle$ náhodná mapa s 1 cieľovým stavom
- $\gamma = 0.98$ a $\alpha = 0.7$



Obr. 4.7: Schéma experimentu

- hustota referenčného riešenia = $1/32$ (4096 stavov)
- počet akcií v každom stave = 8
- hustota riedkej tabuľky = $1/8$ (1:16 pomer)
- počet bázičských funkcií $l = 64$
- rozsah parametrov

$$- \alpha_{ja}(n) \in \langle -1, 1 \rangle$$

$$- \beta_{ja}(n) \in \langle 0, 200 \rangle$$

$$- w_{ja}(n) \in \langle -4, 4 \rangle$$

$Q_{rt}(s(n), a(n))$ referenčná funkcia Q (funkcia 0), kde $t \in \langle 0, 19 \rangle$ je číslo trialu

$Q_{jt}(s(n), a(n))$ testované funkcie Q a $j \in \langle 1, 5 \rangle$.

Celková chyba behu trialu t je

$$e_{jt} = \sum_{s,a} (Q_{rt}(s,a) - Q_{jt}(s,a))^2$$

priemerná, minimálna, maximálna chyba a smerodajná odchylka

$$\bar{a}_j = \frac{1}{20} \sum_t e_{jt}$$

$$e_j^{\min} = \min_t e_{jt}$$

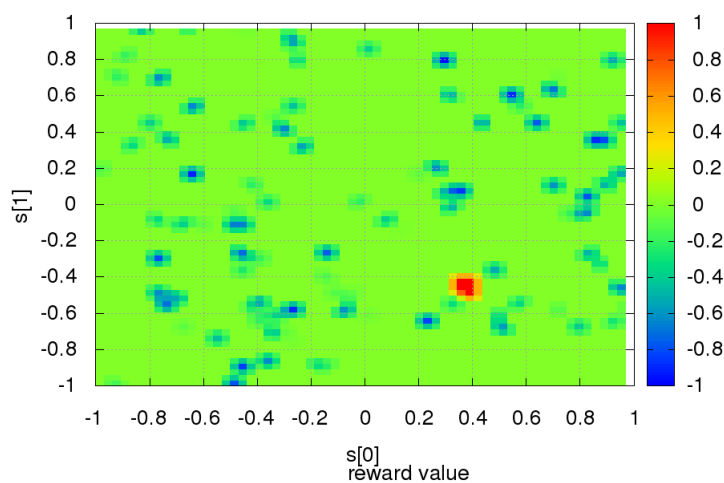
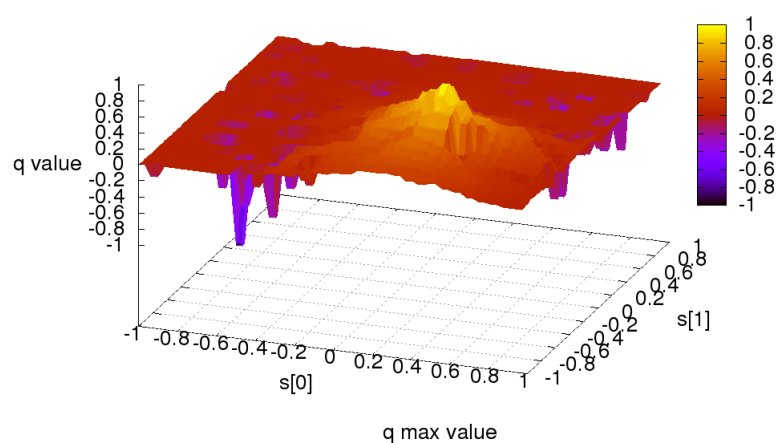
$$e_j^{\max} = \max_t e_{jt}$$

$$\sigma_j^2 = \frac{1}{20} \sum_t (\bar{a}_j - e_{jt})^2$$

4.4 Výsledky experimentu

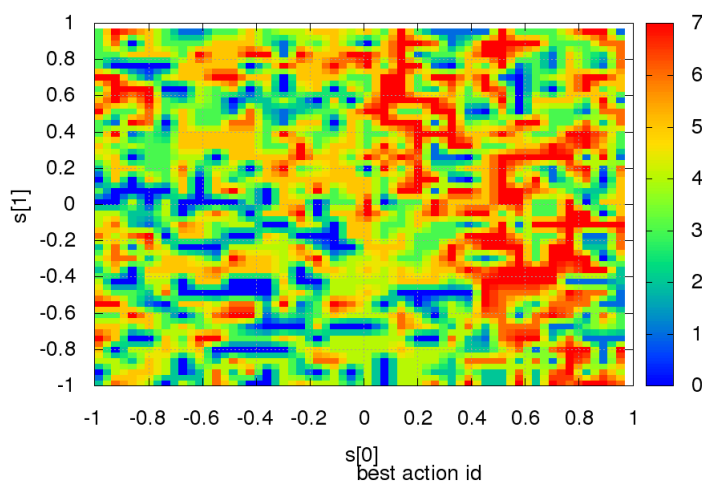
Experiment bol spočítaný pre 4 rôzne mapy - funkcie $R(s(n), a(n))$. Je potrebné poznamenať, že takto navrhnuté prostredie umožňuje agentovi aby nastala každý možný stav - to komplikuje možnosť redukcie počtu stavov. Ukážka mapy č. 2 je na obrázku 4.8. Pričom ako bolo v predošlej časti povedané, platí $R^k(s(n), a(n)) = R^k(s(n))$, t.j. odmena je rovnaká v každom prechode vedúcom do rovnakého stavu.

Pre riešenie Q funkcie s použitím tabuľky, ktoré bude vzhľadom na podmienky experimentu presným riešim je graf funkcie $\max_{a(n) \in \mathbb{A}} Q_n(s(n), a(n))$ na obrázku 4.9. Je možné ľahko pozorovať maximum v oblasti jediného kladného $R(s(n), a(n))$. Od tohto maxima sa šíria hodnoty na celý definičný obor podľa vzťahu 3.2. Ďalej je možné pozorovať záporné hodnoty, ktoré sa nešíria ďalej - predstavujú oblasti kde $R(s(n), a(n))$ nadobúda tiež záporné hodnoty. Na základe známeho $Q(s(n), a(n))$ je možné zostaviť mapu ktorú akciu číslovanú 0 až 7 má agent zvoliť - mapa najlepších akcií v danom stave je znázornená na obrázku 4.10.

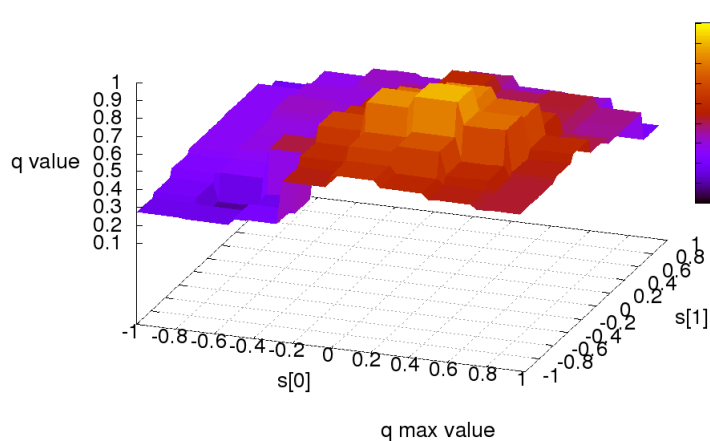
Obr. 4.8: Odmeňovacia funkcia $R(s(n), a(n))$, mapa 2

Obr. 4.9: Referenčné riešenie

Riešenie pre použitie riedkej tabuľky na aproximáciu je viditeľné na obrázku 4.11. Je vidieť nespojitú zmenu, a absenciu schopnosti aproximovať náhle záporné hodnoty požadované zápornou $R(s(n), a(n))$. Zo známeho $Q(s(n), a(n))$ je ďalej možné zostaviť mapu najlepších akcií (očíslované od 0..7). Závislosť čísla najlepšej akcie od stavu je na obrázku 4.12.

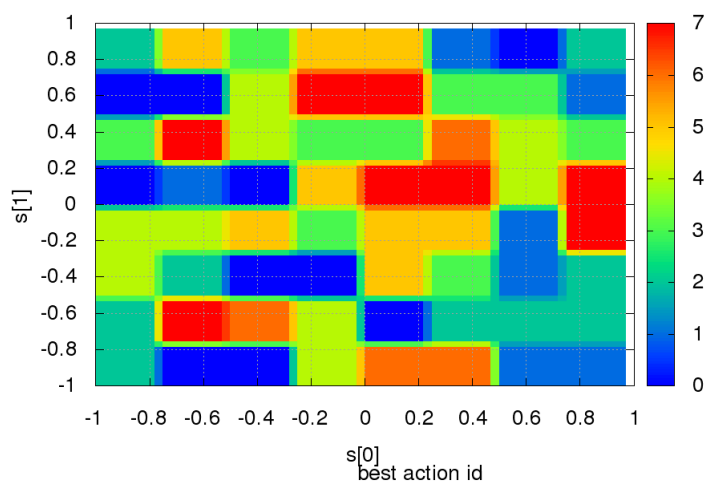


Obr. 4.10: Číslo najlepšej akcie použitím referenčného riešenia



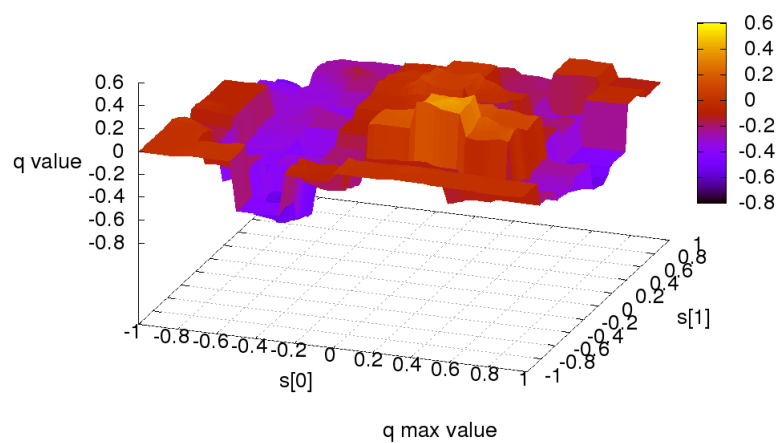
Obr. 4.11: Riešenie aproximácie použitím riedkej tabuľky

Riešenie pre bázičnú funkciu typu Gaussova krivka kombinovaná s riedkou tabuľkou (funkcia typu 3, obr. 4.7) je na obrázku 4.13. Je možné vidieť nepojité zmeny spôsobené riedkou tabuľkou aj vyhladené oblasti vďaka Gaussovým krivkám. Podobne ako v predošlom prípade je možné znázorniť závislosť najlepšej akcie od stavu na obrázku 4.14. Oproti riešeniu

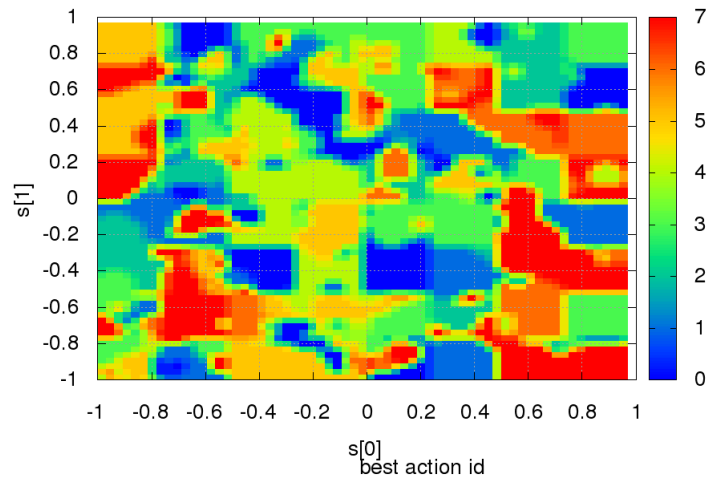


Obr. 4.12: Číslo Najlepšej akcie použitím riedkej tabuľky

s riedkou tabuľkou je možné pozorovať zjemnenie prechodov.

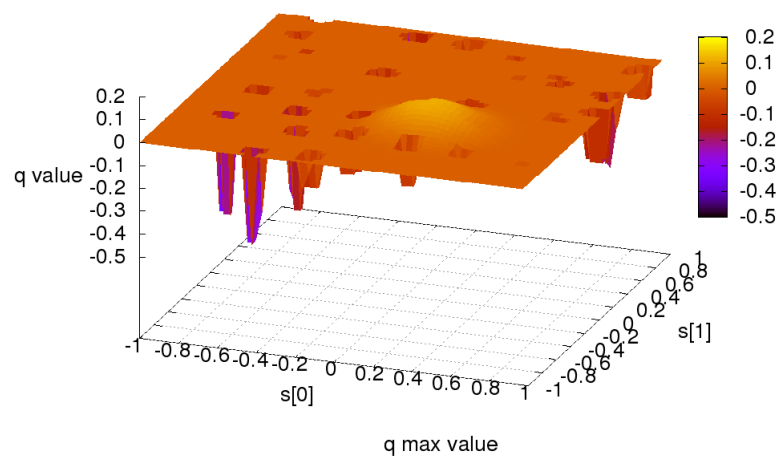


Obr. 4.13: Riešenie aproximácie použitím Gaussovej krivky kombinovanej s riedkou tabuľkou

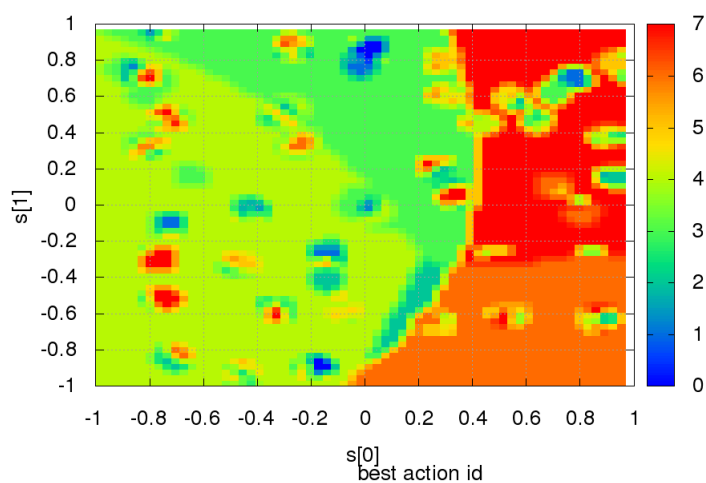


Obr. 4.14: Číslo najlepšej akcie použitím Gaussovej krivky kombinovanej s riedkou tabuľkou

Pre úplnosť, znázornenie riešenia pre novo zavedenú funkciu 3.7.4 - kombinujúc výhody hladkej Gaussovej krivky s tabuľkou, v bodoch kde sú záporné hodnoty $R(s(n), a(n))$. Výsledok pre $\max_{a(n) \in \mathbb{A}} Q_n(s(n), a(n))$ je možné pozrovať na obrázku 4.15 a mapa najlepších akcií je na obrázku 4.16.



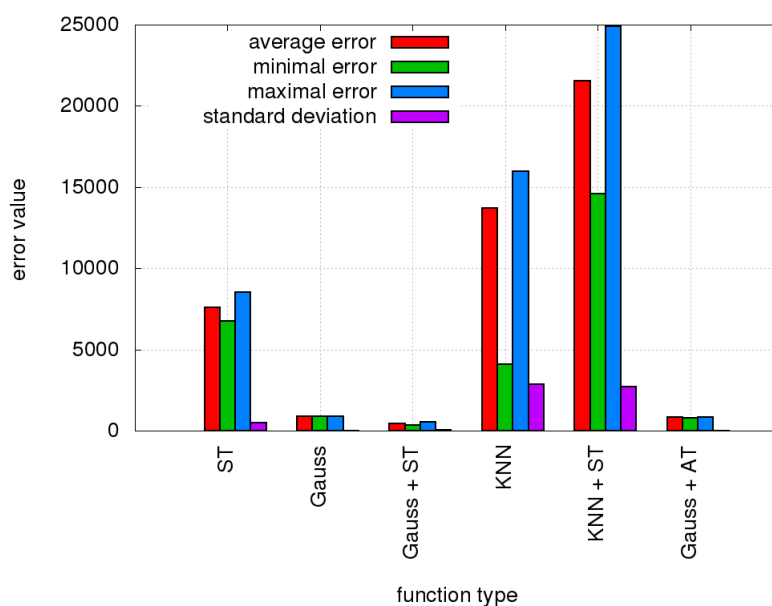
Obr. 4.15: Riešenie aproximácie použitím Gaussovej krivky kombinovanej s adaptívnou tabuľkou



Obr. 4.16: Číslo najlepšej akcie použitím Gaussovej krivky kombinovanej s adaptívnou tabuľkou

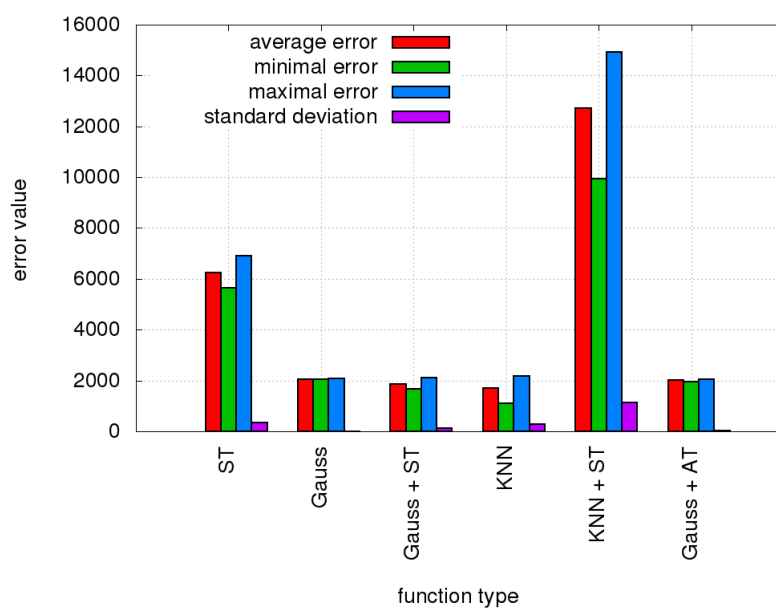
4.5 Priemerné výsledky experimentu

V predošlej časti uvedené výsledky zobrazujú výsledne riešenia pre 3 zvolené aproximačné metódy a jednu mapu - mapu 2. Celkové vyhodnotenie behu všetkých trialov a pre všetky štyri mapy je na nasledujúcich obrázkoch. Sledovali sa veličiny priemerná, maximálna, minimálna chyba a rozptyl chyby. Znázornenie výsledkov je na obrázkoch 4.17, 4.18, 4.19 a 4.20.

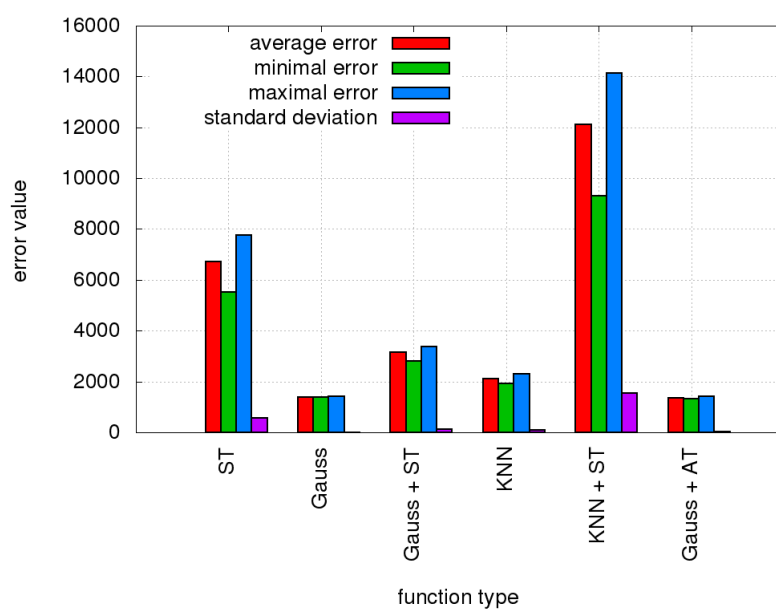


Obr. 4.17: Súhrnné výsledky pre všetky testovacie funkcie a mapu 0

Z výsledkov je možné vybrať troch najvhodnejších kandidátov na aproximáciu : Gaussova krivka, Gaussova krivka kombinovaná s riedkou tabuľkou, Gaussova krivka kombinovaná s adaptívnou tabuľkou. Výsledky sa môžu zdať vyrovnané, dôležité je však znázorniť pohyby jednotlivých virtuálnych robotov a podľa toho urobiť záver. Pohyby robotov pre mapu 2 a jednotlivé aproximačné metódy je možné sledovať na obrázkoch 4.21 4.22 4.23 a 4.24. Je zrejmé, že jedina vyhovujúca aproximačná metóda pre uvedené parametre experimentu je novo zavedená funkcia 3.7.4. Príčinou zlyhania ostatných je neschopnosť zabezpečiť potrebnú strmosť v oblastiach zápornej hodnoty $R(s(n), a(n))$. Samotná rekurentná povaha Q-learning algoritmu spôsobuje, že s rastúcou vzdialenosťou od jediného kladného $R(s(n), a(n))$ sa znižujú rozdiely $Q(s(n), a(n))$ pre jednotlivé akcie ktoré je možné v danom stave vykonať. To je obzvlášť nepríjemné pre experiment tak ako bol navrhnutý - malá zmena pohybu

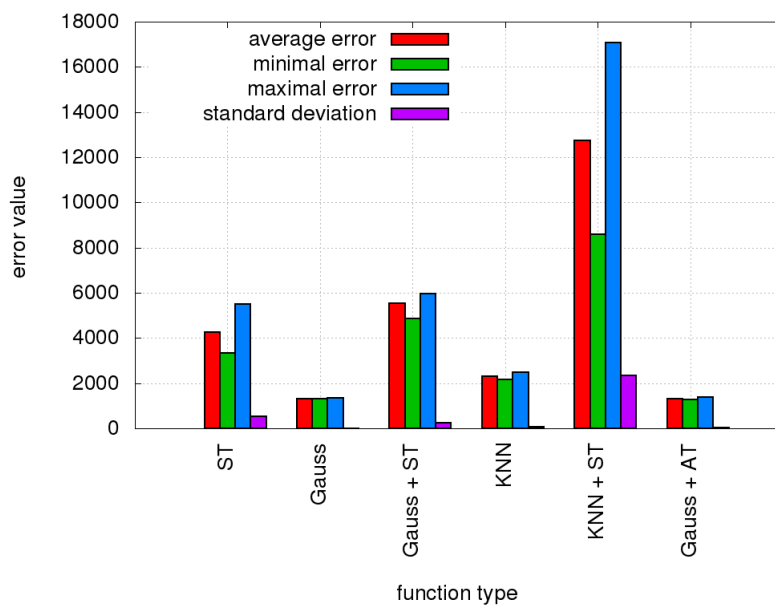


Obr. 4.18: Súhrnné výsledky pre všetky testovacie funkcie a mapu 1

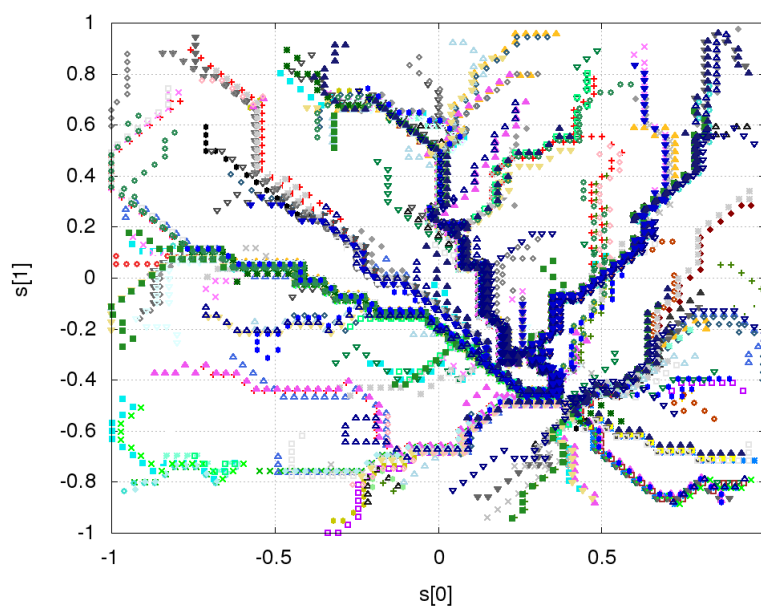


Obr. 4.19: Súhrnné výsledky pre všetky testovacie funkcie a mapu 2

robota znamená aj malú zmenu stavu a aproximačná funkcia tak veľmi ťažko zachytí zmenu s požadovanou presnosťou.



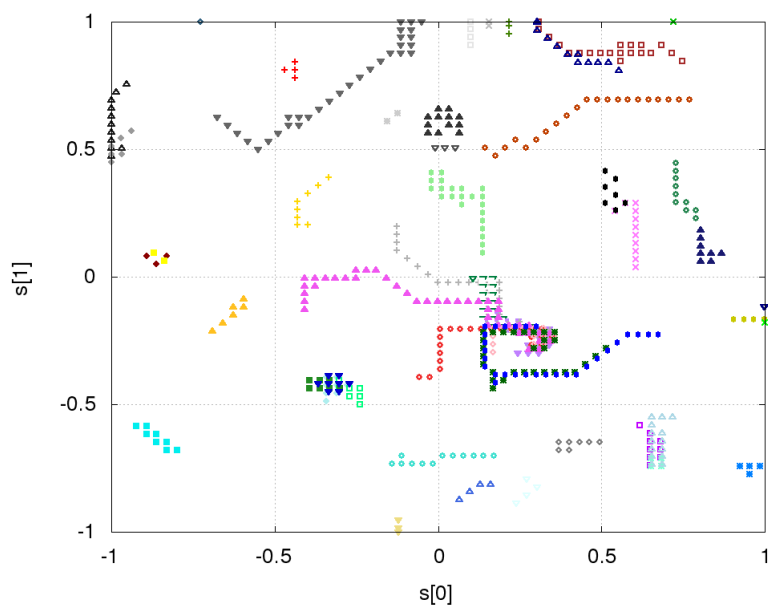
Obr. 4.20: Súhrnné výsledky pre všetky testovacie funkcie a mapu 3



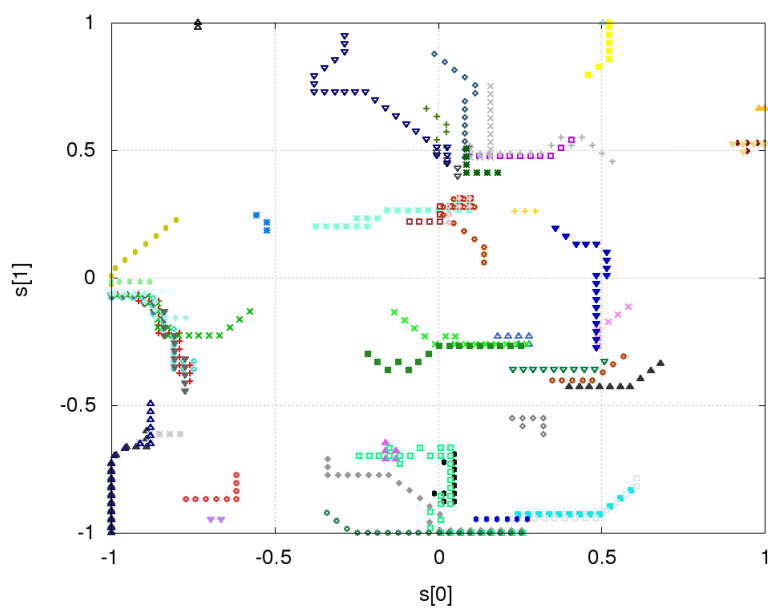
Obr. 4.21: Dráha robotov, referenčné riešenie

4.6 Doplnkový experiment

Zaujímavá aplikácia uvedených metód je robot sledujúci čiaru. Robot má dva motory, kde zmenou rýchlosti otáčania jednotlivých motorov je možné meniť jeho polohu. Cieľom je prejsť po čiare za čo najkratší čas. Vybavený je snímačmi polohy čiar typ APDS9950, mik-



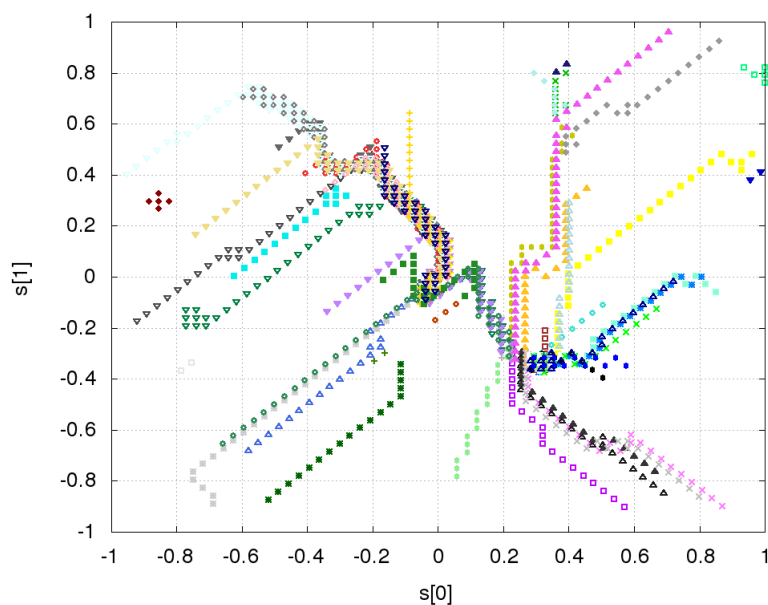
Obr. 4.22: Dráha robotov, aproximacia Gaussovou krivkou



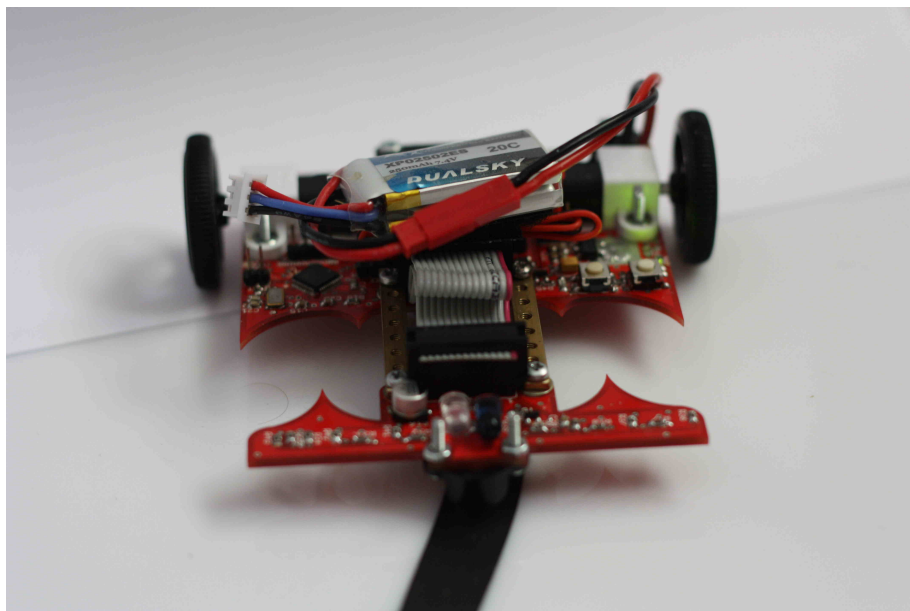
Obr. 4.23: Dráha robotov, aproximacia Gaussovou krivkou kombinovanou s riedkou tabuľkou

rokontrolérom STM32F303 (72MHz arm cortex M4), gyroskopom a dvoma jednosmernými motormi, fotografia robota je na obrázku 4.25.

Pre zjednodušený fyzikálny model je robota možné aproximovať kvádom o rozmeroch w , h , d a rozchode kolies b homogénne rozloženej hmoty na ktorú pôsobia dve sily pravého a

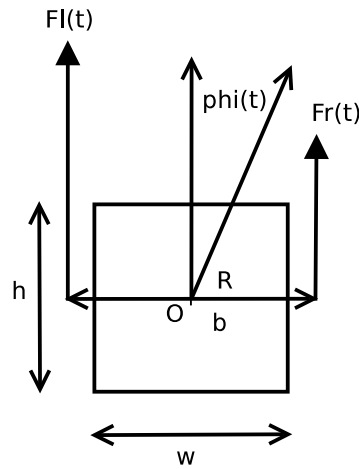


Obr. 4.24: Dráha robotov, aproximacia Gaussovou krivkou kombinovanou s adaptívnou tabuľkou



Obr. 4.25: Fotografia robota

Ľavého motora $F_r(t)$ a $F_l(t)$. Poloha robota je vzťahnutá na bod $R(x, y)$, na začiatku splývajúci s počiatkom súradnicovej sústavy O . Pôsobením týchto síl sa robot pohne a otočí na uhol $\phi(t)$, schématické znázornenie je na obrázku 4.26.



Obr. 4.26: Odvodenie fyzikálneho modelu robota

Ak budenie motorov bude $u_l(n)$ a $u_r(n)$ a moment zotrvačnosti rotora je zanedbateľný oproti momentu zotrvačnosti robota potom sily pôsobiace na ľavé a pravé koleso sú približne

$$F_l(t) = ku_l(t) \quad (4.4)$$

$$F_r(t) = ku_r(t) \quad (4.5)$$

$$M_f(t) = \frac{b}{2}(F_l(t) - F_r(t)) \quad (4.6)$$

kde k je konštanta úmernosti a $M_f(t)$ je výsledný točivý moment robota.

Pre rozmery w , h a hmotnosť m je moment zotrvačnosti robota

$$J = \frac{1}{12}m(w^2 + h^2) \quad (4.7)$$

Pre uhol robota je potom možné napísať

$$\frac{d^2\phi(t)}{dt^2} = \frac{1}{J}M_f(t) - \frac{b}{J}\frac{d\phi(t)}{dt} \quad (4.8)$$

Rovnica 4.8 aproximuje postupný nábeh rýchlosti otáčania robota prvým rádom. Za rýchlosť nábehu zodpovedá koeficient tlmenia b . Riadenie hodnoty uhla potom vedie na PD regulátor. Ten riadi rozdiel rýchlosti ľavého a pravého motora, $d(n)$.

Pre doprednú rýchlosť robota a koeficient tlmenia c platí

$$F(t) = F_l(t) + F_r(t) \quad (4.9)$$

$$\frac{dv(t)}{dt} = \frac{F(t)}{m} - cv(t) \quad (4.10)$$

po úpravach je podobne ako pre uhol možné navrhnuť regulátor, tentoraz ale PI regulátor - žiadaná hodnota je dopredná rýchlosť $v(n)$, nie poloha.

Riadenie ľavého $u_l(n)$ a pravého motora $u_r(n)$ je teda rozdelené na dve časti

$$u_l(n) = v(n) + d(n) \quad (4.11)$$

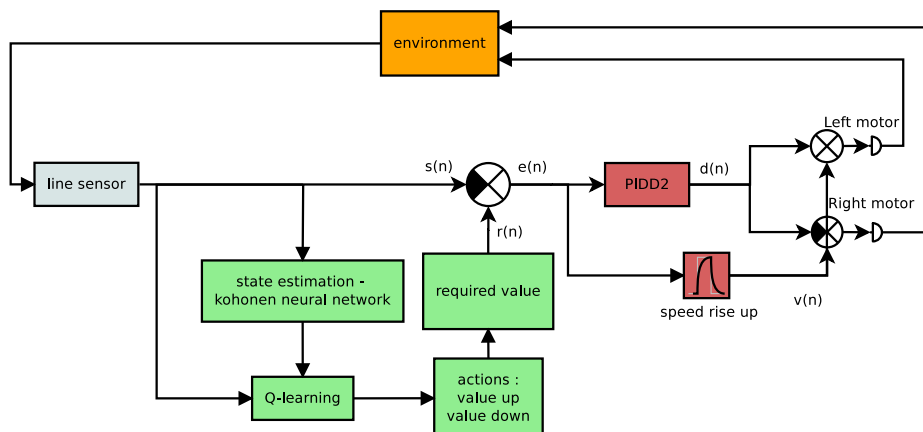
$$u_r(n) = v(n) - d(n) \quad (4.12)$$

$$(4.13)$$

$v(n)$ je dopredná rýchlosť a $d(n)$ je rozdiel rýchlostí motorov.

Poloha čiary je získaná zo snímača, nadobúda 256 diskretných hodnôt a na blokovej schéme 4.27 je označená ako $s(n)$. Vzorkovacia frekvencia je 200Hz.

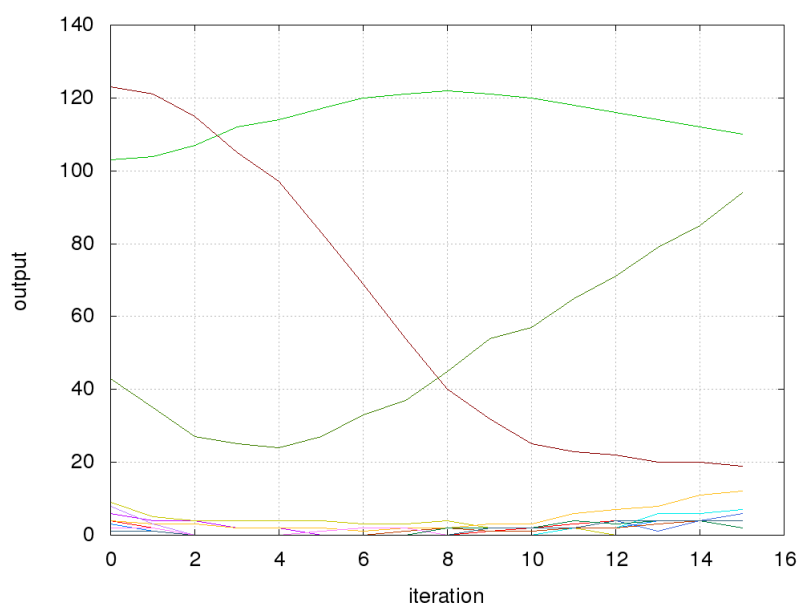
Tu je potrebné poznamenať, že požadovaná hodnota $r(n)$ nie je známa ani pre uhol natočenia ani pre rýchlosť. Dráha po ktorej robot prechádza má zákruty rôzneho tvaru aj rovné úseky. Triviálny prípad je možné získať pre $r(n) = 0$. Robot tak na základe chyby $e(n) = r(n) - s(n)$ môže upravovať rýchlosť aj natočenie.



Obr. 4.27: Bloková schéma riadiaceho systému robota

Práve tu sa ukazuje ako zaujímavé riešenie použiť Q-learning algoritmus na stanovenie požadovanej hodnoty $r(n)$. Stav robota je určený niekoľkými po sebe zameranými polohami

čiaru so senzora $s(n)$. V tomto konkrétnom prípade ide o 16 vzoriek. Keďže existuje veľmi veľký počet možných stavov (pre symetrické správanie sa v ľavo aj pravotočivých zákrutách je to 128^{16} bola použitá Kohonenová neurónová sieť na klasterizáciu stavov do 16 kategórií - predstavujú zákruty s rôznou strmosťou, obrázok 4.28. Na ose iteration vidno vstupnú vzorku v poradí $n..n - 16$ na ose output je hodnota snímača čiaru. Po niekoľkých prejdieniach tréningovej dráhy sa tvary zákrut ustálili tak ako sú na obrázku. Každému typu zákruty (stavu) je tak možné priradiť žiadanú hodnotu $r(n)$. Mechanizmus stanovenia týchto hodnôt je algoritmus Q-learning. Akcie boli zvýšiť rýchlosť pre daný typ zákruty alebo znížiť. Odmeňovacia funkcia je filtrovaná poloha čiaru - čím viac sa robot odchyľuje od požadovanej dráhy, tým väčší je trest. Robota v prevádzke je možné vidieť na videu [45].



Obr. 4.28: Klasterizácia rôznych zákrut

Kapitola 5

Záver

Práca rieši problematiku aproximovania funkcie ohodnotení v algoritmoch Q-learning. S pomoci najčastejšie používaných prístupov bola zvolená aproximácia neurónovou sieťou pomocou bázičných funkcií. Oproti bežne používanému prístupu lineárnej kombinácií príznakov (features) sa líši tým, že samotné tvary príznaky si algoritmus stanovuje sám, počas učenia. Zmenšuje sa teda potrebná znalosť programátora.

Vedecký prínos je možné nájsť v

- Ukážka nevhodnosti použitia doprednej siete v predloženom probléme učenou gradientovými metódami. Riešenie nekonvergovalo ani po miliónoch iteráciach v triviálnom experimente s dvoma akciami. Príčinou je nelokálnosť učenia siete - zmena hodnoty v jednom bode, zmení hodnoty v každom bode, a nie nutne k lepšiemu. Od siete sa súčasne požaduje generovanie správnej hodnoty aj učenie v nejakom inom bode.
- Uvedenie algoritmu nanoQ, ktorý vyšetruje systém s jedným stavom. Nepodarilo sa nájsť publikáciu ktorá by tento princíp využívala. Algoritmus môže nájsť uplatnenie v riešení pohybu jednoduchého robota.
- Uvedenie novej bázičkej funkcie, ktorá z testovaných najlepšie aproximuje funkciu ohodnotení. Táto funkcia môže byť učená lokálne, a vďaka časti $P(s(n), a(n))$ umožňuje zabezpečiť potrebnú strmosť, bez nutnosti širokého rozsahu parametrov β v časti $H(s(n), a(n))$ - ten môže zostať malý, a riešiť tak šírenie kladnej odmeny na ďalšie stavy v súlade s parametrom γ .

- Testovanie Q-learning algoritmu na reálnom robotovi, kde predstavuje druhú úroveň riadenia. Na spodnej vrstve sa pracuje s PID regulátormi, na druhej sa pomocou Q-learning algoritmu stanovujú žiadané hodnoty.

Napriek uvedeným skutočnostiam a súčasnemu stavu komerčnej sféry, autor práce nepredpokladá využitie Q-learning algoritmov v priemyselnej praxi. Medzi hlavné dôvody možno zaradiť konzervatívny prístup riadenia v priemysle, kde väčšinu úloh plnohodnotne vyrieši PID regulátor a nad ním postavená logika vetvenia (napr. rôzne stavové automaty). Práca tak predstavuje nepatrný prínos v teoretickej oblasti reinforcement learning algoritmov. Jediné možné využitie v blízkej dobe je možné nájsť v počítačových hrách. Všetky zdrojové súbory a podrobné výsledky experimentov (vrátane dát na ďalšie smerovanie) sú k dispozícii pod GNU GPL licenciou. Práca tak spadá do kategórie otvorenej vedy. Zdrojové súbory pre Q-learning experiment sú k dispozícii na autorovom gite [46]. Spolu je to cca 55648 súborov, z toho cca. 17000 pripadá na výsledky experimentov a cca 36000 na zdrojové súbory. Zdrojové súbory (vrátane podkladov na výrobu) pre robota Motoko sú k dispozícii na [47].

Literatúra

- [1] Nhan Nguyen, NASA Ames Research Center, Moffett Field, CA 94035 : Predictor-Model-Based Least-Squares Model-Reference Adaptive Control with Chebyshev Orthogonal Polynomial Approximation
- [2] Girish Chowdhary and Eric Johnson, Least Squares Based Modification for Adaptive Control http://web.mit.edu/girishc/www/publications/files/Chow_Joh_CDC_10_ls.pdf
- [3] Sun Pei, Noise Resistant Least Squares Based Adaptive Control, March 27, 2012, Stockholm, Sweden <http://www.diva-portal.org/smash/get/diva2:514116/FULLTEXT01.pdf>
- [4] Prof. Nathan L. Gibson Department of Mathematics, Gradient-based Methods for Optimization. Part I., 2011 <http://math.oregonstate.edu/~gibsonn/optpart1.pdf>
- [5] Antony Jameson, Department of Aeronautics and Astronautics Stanford University, Stanford, CA 94305-4035 Gradient Based Optimization Methods, <http://aero-comlab.stanford.edu/Papers/jameson.gbom.pdf>
- [6] L. Hasdorff, Gradient optimization and nonlinear control, ISBN 0471358703, https://books.google.cz/books?id=o_ZQAAAAMAAJ
- [7] Kevin L. Moore, Iterative Learning Control, <http://inside.mines.edu/~kmoore/survey.pdf>
- [8] Kevin L. Moore, An Introduction to Iterative Learning Control Theory, http://inside.mines.edu/~kmoore/504_ILC_Seminar-Save.pdf

- [9] Jeff Heaton, Introduction to Neural Networks with Java, Heaton Research, Inc., 2008, ISBN 1604390085
- [10] CHRISTOPHER J.C.H. WATKINS, PETER DAYAN : Technical Note Q-Learning, Machine Learning, 8,279-292 (1992) <http://www.gatsby.ucl.ac.uk/~dayan/papers/cjch.pdf>
- [11] Q-learning 1 <https://www-s.acm.illinois.edu/sigart/docs/QLearning.pdf>
- [12] Q-learning 2 <http://mnemstudio.org/path-finding-q-learning-tutorial.htm>
- [13] Francisco S. Melo Institute for Systems and Robotics, Instituto Superior Técnico, Lisboa, PORTUGAL : Convergence of Q-learning: a simple proof <http://users.isr.ist.utl.pt/~mtjspaen/readingGroup/ProofQlearning.pdf>
- [14] Eyal Even-Dar, Yishay Mansour : Convergence of optimistic and incremental Q-learning, <http://web.cs.iastate.edu/~honavar/rl-optimistic.pdf>
- [15] Carden, Stephen, "Convergence of a Reinforcement Learning Algorithm in Continuous Domains"(2014). All Dissertations. Paper 1325. http://tigerprints.clemson.edu/cgi/viewcontent.cgi?article=2326&context=all_dissertations
- [16] Francisco S. Melo and M. Isabel Ribeiro, Convergence of Q-learning with linear function approximation, Proceedings of the European Control Conference 2007 Kos, Greece, July 2-5, 2007, <http://gaips.inesc-id.pt/~fmelo/pub/melo07ecc.pdf>
- [17] Karan M. Gupta Department of Computer Science Texas TechUniversity Lubbock, TX 79409-3104 : Performance Comparison of Sarsa(λ) and Watkin's Q(λ) Algorithms, <http://www.karanmg.net/Computers/reinforcementLearning/finalProject/KaranComparisonOfSarsaWatkins.pdf>
- [18] R. Rojas: Neural Networks, Springer-Verlag, Berlin, 1996, Kohonen Networks <https://page.mi.fu-berlin.de/rojas/neural/chapter/K15.pdf>

- [19] Steven K. Rogers, Matthew Kabrisky SPIE Press, 1991, ISBN 0819405345 : An Introduction to Biological and Artificial Neural Networks for Pattern Recognition <https://books.google.cz/books?id=uo4Smk6QnTgC>
- [20] Teuvo Kohonen and Timo Honkela (2007), Scholarpedia, 2(1):1568 : Kohonen network http://www.scholarpedia.org/article/Kohonen_network
- [21] Markovove rozhodovacie procesy, stručne : Pieter Abbeel UC Berkeley EECS : Markov Decision Processes and Exact Solution Methods <http://www.cs.berkeley.edu/~pabbeel/cs287-fa12/slides/mdps-exact-methods.pdf>
- [22] Martin L. Puterman : Markov Decision Processes: Discrete Stochastic Dynamic Programming , isbn 9781118625873, rok 2014, <https://books.google.sk/books?id=VvBjBAAAQBAJ>
- [23] NanoQ learning zdrojové súbory https://github.com/michalnand/q_learning/tree/master/src/nano_q_learning
- [24] Fundamentals of Artificial Neural Networks Mohamad H. Hassoun, MIT Press, 1995
- [25] B. Irie Auditory & Visual Perception Res. Lab., ATR, Osaka, Japan, S. Miyake : Neural Networks, 1988., IEEE International Conference on, INSPEC 3350063
- [26] Kolomongorov teorém, stručne https://en.wikipedia.org/wiki/Universal_approximation_theorem
- [27] R. Rojas: Neural Networks, Springer-Verlag, Berlin, 1996, chap 7
- [28] Martin Riedmiller, Computer Standards & Interfaces Volume 16, Issue 3, July 1994, Pages 265-278 : Advanced supervised learning in multi-layer perceptrons — From backpropagation to adaptive learning algorithms
- [29] J. Leonard, M.A. Kramer, Computers & Chemical Engineering Volume 14, Issue 3, March 1990, Pages 337–341 Improvement of the backpropagation algorithm for training neural networks

- [30] Jonathan Engel, Norman Bridge Laboratory of Physics 161-33, California Institute of Technology, Pasadena, CA 91125, USA : Teaching Feed-Forward Neural Networks by Simulated Annealing
- [31] Francisco S. Melo, Sean P. Meyn, M. Isabel Ribeiro An Analysis of Reinforcement Learning with Function Approximation, Appearing in Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 2008 <http://www.machinelearning.org/archive/icml2008/papers/652.pdf>
- [32] David Silver : Lecture 6: Value Function Approximation http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/FA.pdf
- [33] Francisco S. Melo M. Isabel Ribeiro : Q-learning with linear function approximation <http://gaips.inesc-id.pt/~fmelo/pub/melo07tr-b.pdf>
- [34] Marina Irodova and Robert H. Sloan : Reinforcement Learning and Function Approximation, 2005, American Association for Artificial Intelligence (www.aaai.org) <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.81.7833&rep=rep1&type=pdf>
- [35] Punit Pandey, Dr. Shishir Kumar, Deepshikha Pandey, Reinforcement Learning by Comparing Immediate Reward, (IJCSIS) International Journal of Computer Science and Information Security, Vol. 8 , No. 5, August 2010 <http://arxiv.org/pdf/1009.2566.pdf>
- [36] Melanie Coggan, CRA-W DMP Project at McGill University (2004) : Exploration and Exploitation in Reinforcement Learning http://ftp.bstu.by/ai/To-dom/My_research/Papers-2.1-done/RL/0/FinalReport.pdf
- [37] Mark Humphrys Trinity Hall, University of Cambridge August 1996, <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.73.8309&rep=rep1&type=pdf>
- [38] Jinyi Yao Dept. of Comput. Sci. & Technol., Tsinghua Univ., Beijing, China Jiang Chen ; Zengqi Sun An application in RoboCup combining Q-learning with adversarial

- planning, 2002 http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1022159&abstractAccess=no&userType=inst
- [39] Asma Al-Tamimi, Frank L. Lewis , Murad Abu-Khalaf Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control, 2006 <http://www.sciencedirect.com/science/article/pii/S0005109806004249>
- [40] Asma Al-Tamimi, Frank L. Lewis , Murad Abu-Khalaf Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control, 2006 <http://www.sciencedirect.com/science/article/pii/S0005109806004249>
- [41] Christopher J. C. H. Watkins, Peter Dayan Q-learning, 1992 <http://link.springer.com/article/10.1007/BF00992698>
- [42] Mae L. Seto Springer Science & Business Media, 9. 12. 2012, Marine Robot Autonomy, ISBN 1461456592, chap 7.3.3.2
- [43] Peter Dayan, Christopher J.C.H. Watkins, Reinforcement Learning <http://www.gatsby.ucl.ac.uk/~dayan/papers/dw01.pdf>
- [44] Daniel Dewey, Oxford Martin Programme on the Impacts of Future Technology, Future of Humanity Institute : Reinforcement Learning and the Reward Engineering Principle <http://www.danieldewey.net/reward-engineering-principle.pdf>
- [45] video robota Motoko Aftermath Michal Chovanec, youtube https://www.youtube.com/watch?v=8sskJN_zuko
- [46] Michal Chovanec, Q-learning zdrojové súbory https://github.com/michalnand/q_learning
- [47] Michal Chovanec, Motoko robot zdrojové súbory https://github.com/michalnand/motoko_after_math_linefollower