UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

KATEDRA APLIKOVANEJ INFORMATIKY

OVLÁDANIE VIRTUÁLNEHO AUTOMOBILU POMOCOU NEURÓNOVEJ SIETE

(Diplomová práca)

MAREK CIFRA

Študijný odbor: 9.2.8 Umelá inteligencia a 3.1.9 Psychológia

Študijný program: Kognitívna veda

Školiteľ: Doc. Ing. Igor Farkaš, PhD.

BRATISLAVA 2009

Čestné vyhlásenie

Vyhlasujem, že som diplomovú prácu vypracoval samostatne a uviedol som všetku použitú literatúru.

……

vlastnoručný podpis študenta

Poďakovanie

Ďakujem Doc. Ing. Igorovi Farkašovi, PhD. za ochotu, cenné rady a usmernenie pri písaní záverečnej práce.

Abstrakt

CIFRA, Marek: *Ovládanie virtuálneho automobilu pomocou neurónovej siete.* [Diplomová práca] – Univerzita Komenského v Bratislave. Fakulta matematiky, fyziky a informatiky; Katedra aplikovanej informatiky. – Vedúci: Doc. Ing. Igor Farkaš, PhD. Bratislava: UK, 2009.

Cieľom tejto práce je vytvoriť systém na ovládanie automobilu, ktorý bude schopný udržať automobil na ceste na základe učenia sa pomocou interakcie s prostredím. Na tento účel využívame metódu učenia posilňovaním, ktorej princíp je na začiatku práce vysvetlený. Zároveň táto práca poukazuje na súvislosti medzi týmto typom adaptácie v strojom učení a biologickým učením v živých systémoch. Počas samotného návrhu takéhoto systému som dospel k jeho postupnému vylepšovaniu, čo vyústilo v použití neurónovej siete s učením posilňovaním. Správanie všetkých simulovaných modelov je ilustrované pomocou grafov.

Kľúčové slová: reinforcement learning, učenie posilňovaním, odmena, trest, neurónová sieť, algoritmus SARSA, automobil, agent

Abstract

The aim of this work is to create a system for driving a car, which will be able to keep the car on the road. This system must learn from environment through bilateral interaction. For this purpose we used reinforcement learning method, whose principle is explained. It also shows the correlation between machine learning and biological learning in living systems. Besides the designing of the system, some improvements are presented too, resulting in using neural network with reinforcement learning. The performance of all simulated models is graphically illustrated.

Predhovor

Táto práca bola inšpirovaná projektom Starfish, robotickej hviezdice, ktorá sa učila používať svoje končatiny na pohyb. To bolo prvýkrát, čo som sa stretol s pojmom učenia posilňovaním. V dobe začiatku písania tejto práce som teda nemal o učení posilňovaním žiadne vedomosti, pretože sa táto problematika na našej univerzite nevyučuje, no s postupom času a odsimulovaním jednotlivých modelov som začal plne chápať jej princíp. Na vysvetlenie problematiky učenia posilňovaním sme využili niektoré kapitoly z knihy Reinforcement Learning (Sutton a Barto, 1998), z ktorej som sám čerpal vedomosti a považujem túto knihu za „Bibliu“ učenia posilňovaním.

Vzhľadom na nedostatok materiálu pre robotiku a na náročnosť virtuálneho prostredia pri počítačovej podobnej simulácií ako je Starfish, som sa rozhodol vytvoriť iný systém využívajúci spomínanú metódu učenia, menej náročný na výpočtový a implementačný čas samotného prostredia, aby my zostalo viac času na samotný model.

Cieľom práce je teda vytvoriť systém pre ovládanie automobilu, schopným udržať automobil na ceste. Keďže som však s prvotným modelom nebol spokojný, začal som ho vylepšovať, čo tiež opisujem v tejto práci. Nakoniec som prešiel od tabuľkového systému k neurónovým sieťam, čo sa ukázalo ako veľmi dobrý krok. Posledný mnou navrhnutý model by mal byť schopný ovládať aj skutočné automobily, resp. roboty, čo bol aj cieľ tejto práce.

Bc. Marek Cifra

Obsah

Zoznam ilustrácií a tabuliek

1 Úvod

V dnešnej dobe ponúkajú luxusné autá parkovacieho asistenta, systémy pre čítanie dopravných značiek, udržiavanie jazdného pruhu a vzdialenosti od auta pred sebou a mnohé iné technické „zázraky“. Riešenie týchto systémov, je založené na štatistike, resp. sú to napevno programované inštrukcie. To ale znamená, že systémy sa neučia.

V tejto diplomovej práci sa budem venovať vytvoreniu takého systému, ktorý sa bude schopný učiť a využívať svoje znalosti na udržanie automobilu na ceste, čo je obdoba systému pre udržiavanie jazdného pruhu. Následne budem tento systém zdokonaľovať.

V procese učenia použijem metódu učenia posilňovaním, ktorá vracia odmenu za vykonanú akciu a jej princípom je maximalizovať túto odmenu. Aby som však dobu potrebnú na učenie skrátil, využijem techniku, ktorú som nazval faktor skreslenia. Tú neskôr nahradím neurónovou sieťou pre schopnosť interpolácie. A na koniec celú túto architektúru zoptimalizujem, čím dostaneme efektívny systém s postačujúcimi schopnosťami na udržanie automobilu na ceste. Takýto systém budem v tejto práci nazývať slovom agent. Aj keď ide len o virtuálneho agenta, resp. automobil, mnou navrhnutý systém by mal byť schopný ovládať aj skutočného robota, ak by dostal na vstup obraz z jeho kamery.

Podobný systém sa snažili vyvinúť už v roku 1989 v projekte ALVINN (POMERLEAU, 1989), no nakoniec bol pozastavený. ALVINN využíval neurónovú sieť, kde na vstup dostával obraz z kamery a výstupom bola akcia otočenia volantom do vybraného smeru. Rýchlosť auta určoval človek. ALVINN využíval na učenie metódu učenia s učiteľom, kde sledoval správanie človeka riadiaceho automobil. Môj systém s učením posilňovaním je na hranici medzi učením s učiteľom a bez neho, ako si neskôr ukážeme. Nakoniec, učenie posilňovaním súvisí viac s biologickým učením sa ľudí, ovládať napríklad svoju motoriku, ako učenie s učiteľom.

2 Učenie posilňovaním

Učenie posilňovaním [reinforcement learning] (RL), známe aj ako učenie pomocou odmeny a trestu, je skupina metód strojového učenia založená na vracaní odmeny. Pokojne by sme ho mohli tiež nazvať cieľovo-orientované učenie pomocou z interakcie. Ako dojčatá sme sa učili pohybovať končatinami a zároveň sme dostávali spätnú väzbu jednak v podobe vizuálnych signálov (čiže cez zrak) a jednak v podobe hmatových signálov pri dotyku s nejakým predmetom. Pri dotyku sme videli, ako reaguje, napríklad v loptička sa posunula smerom udaným našim pohybom končatiny. Takže na základe týchto interakcií s okolitým svetom sme sa naučili koordinovať nielen naše pohyby, ale aj správanie a všetko s nami spojené.

Existuje veľa problémov, ktoré sa dajú riešiť pomocou RL, napríklad pohyb v bludisku, balansovanie s paličkou a iné. No napriek tomu nebola ešte vytvorená taká verzia, ktorá by bola schopná riešiť všetky problémy bez dodatočných úprav. Jednou z množných oblastí, kde možno účinne testovať a vylepšovať RL sú určite hry, pretože väčšina z nich sú cieľovo-orientované sekvenčné rozhodovacie problémy, kde každé rozhodnutie môže mať dlhotrvajúci efekt (Szita, 2007).

Základným princípom RL je maximalizovať *odmenu*, resp. odpoveď z prostredia, v ktorom sme aplikovali akciu. Agent však nevie, aká je maximálna možná odmena, čo v podstate znamená, že aj po tom, ako agent dosiahne maximum, skúša ďalšie možnosti. Možno teda povedať, že agent získava skúsenosti a zisťuje, aké maximá môže v ktorej situácií dosiahnuť. Keďže agentovi nikto presne nepovie, ktorá voľba je správna, spadá RL do kategórie *učenia bez učiteľa*[1].

V skutočnom svete sme zaplavovaní veľkým množstvom informácií (signálov), s ktorými sa musí náš mozog vysporiadať, najčastejšie odfiltrovaním zbytočných vstupných signálov. Podobne agent môže čeliť takejto záplave, ak ide o komplexný

[1] Znalý čitateľ by mohol namietať, že RL patrí do skupiny učenia s učiteľom a mal by tiež pravdu. Všetko totiž závisí na definícií učiteľa. V našom prípade je učiteľ niečo alebo niekto, kto agentovi presne povie akú akciu mal vykonať a teda v tomto zmysle RL nemá učiteľa.

svet, v ktorom sa pohybuje. Úlohou agenta je teda zachytiť tie najdôležitejšie aspekty prostredia, ktorému čelí, vytvoriť z nich stav okolitého prostredia, vyhodnotiť tento stav a následne vykonať príslušné úkony, aby dosiahol svoj cieľ. Agent teda musí byť schopný, do určitej miery, ovplyvniť svoje okolie a úkony, ktoré vykonáva, slúžia na dosiahnutie čiastkových cieľov, na konci ktorých sa nachádza primárny cieľ. Tiež možno povedať, že sa agent musí rozhodovať aj v *neistom* prostredí, teda jeho rozhodnutie je založené na čiastkových informáciách.

Spomenul som, že agent získava skúsenosti, z ktorých neskôr ťaží pri rozhodovaní. Získavanie skúseností poznáme aj z reálneho sveta. Pokiaľ človek čelí celkom novej udalosti, nemôže sa rozhodnúť na základe predchádzajúcich skúseností. V tej chvíli začína proces skúmania a človek získava prvé skúsenosti s daným javom, ktoré využije, keď sa s ním opätovne stretne. Podobne ako umelý agent nevie či vybraná voľba, resp. akcia, docieli maximálnu odmenu, ani človek nevie, či to čo dosiahol je to najlepšie, čo mohol dosiahnuť a či nie je lepšie skúsiť inú voľbu. Ide teda o kompromis medzi *skúmaním* [exploration] nového a *využívaním* [exploitation] poznaného, čo je najjasnejší rozdiel oproti učeniu s učiteľom, ktorý tento kompromis nepozná, pretože učiteľ vždy žiakovi povie čo mal urobiť.

Ďalším aspektom RL je, že sa na problém pozerá ako na celok a nerozbíja ho na menšie časti. Samozrejme, mohli by sme namietať na efektívnosť rozhodovacieho procesu, keďže mnohokrát je naozaj lepšie a rýchlejšie rozdeliť si problém na podproblémy a následne ich jednotlivo vyriešiť. Toto nám však nezaručuje, že pôvodný problém bude vôbec vyriešený a že bude vyriešený najlepšie možne. Jednotlivé riešenie si totiž môžu navzájom odporovať, a tým sa celé riešenie stáva bezpredmetným.

Žiaľ komplexnosť niektorých problémov nás núti rozdeliť ich na menšie podproblémy. Zoberme si príklad s telefonovaním počas šoférovania auta. Všetci vieme, že je zakázané telefonovať pri riadení motorového vozidla bez použitia hands-free sady. Mohlo by sa zdať, že je to postačujúca podmienka pre bezpečné šoférovanie, ale nie je to mu tak. Problémom je, že človek nedokáže zamerať svoju pozornosť na dve činnosti naraz. V podstate musí prepínať medzi plnohodnotným vedením rozhovoru po telefóne a šoférovaním. Keď sa na to pozrieme z globálneho

hľadiska, človek rozdelil problém šoférovania s telefonovaním na dva podproblémy, ktoré spolu nesúvisia a aktívne prepína medzi nimi. Možno povedať, že ak by sme tento problém nerozdelili, úspešné riešenie by nikdy nebolo dosiahnuté. Preto v tomto prípade by agent mal mať dva subsystémy, kde každý z nich by sa zaoberal iba jednou činnosťou. Ak sa pozrieme na funkčnú stavbu mozgu človeka, nájdeme v nej oblasti, špecializujúce sa na porozumenie reči (Wernickeho oblasť), generovanie reči (Brockova oblasť), priestorová orientácia (hypokampus) a iné, ktoré sú v podstate akýmisi samostatnými subsystémami na vykonávanie danej činnosti, čo presne zodpovedá predchádzajúcej vete. Samozrejme, v skutočnom mozgu je to komplikovanejšie, ale princíp je rovnaký.

2.1 Elementy učenia posilňovaním

Aby autonómny agent s implementovaným RL mohol interagovať s prostredím, musí jeho model obsahovať nasledujúce elementy: *stratégiu*, *funkciu odmeny*, *funkciu hodnotenia* a poprípade aj *model* prostredia.

Stratégia [policy] určuje správanie agenta na základe vnímaných signálov, resp. mapuje stavy prostredia na akcie. Takouto stratégiou môže byť napríklad vyhľadávacia tabuľka, kde v jednom stĺpci sú stavy a v druhom akcie, ktoré sa majú vykonať, ak je prítomný stav v danom riadku ako je akcia. Samozrejme, sú možné aj iné, komplexnejšie, implementácie takejto stratégie založené na netriviálnych výpočtoch a iných metódach, ako sú napríklad neurónové siete. Stratégia je jadrom RL v takom zmysle, že na určenie správania agenta stačí len ona sama. Vo všeobecnosti by mala byť stratégia stochastická (Sutton a Barto, 1998).

Funkcia odmeny [reward function] mapuje stav prostredia na číslo, ktoré vyjadruje želateľnosť stavu. Čím je číslo vyššie, tým väčší bol zámer dosiahnuť daný stav. V podstate hovorí agentovi, ktorá akcia je zlá a ktorá dobrá, resp. lepšia, niečo ako bolesť a slasť u ľudí. Nesmie byť ovplyvniteľná agentom, ale mala by ovplyvňovať správanie agenta, čiže jeho stratégiu tak, aby odmena za vykonanú akciu bola čo najvyššia. Teda v prípade malej odmeny agent zmení svoje správanie tak, aby v budúcnosti, keď bude čeliť rovnakému či podobnému stavu, konal inak, s cieľom zvýšiť odmenu. Podobne ako stratégia, aj funkcia odmeny by mala byť stochastická.

Funkcia hodnoty [value function] sa od *funkcie odmeny* líši tým, že funkcia odmeny zodpovedá okamžitej odpovedi prostredia, zatiaľ čo funkcia hodnoty definuje čo je dobré v dlhodobom meradle. Zameriava sa na nasledujúce stavy (aktuálny stav vynímajúc), ktoré by mohli vzniknúť a *odmeny*, ktoré za tieto stavy dostane. Zoberme si príklad, kedy stav v čase *t+1* dosiahne veľmi vysokú odmenu iba v prípade ak stav v čase *t* dosiahne veľmi malú. Pokiaľ by sa agent riadil iba funkciou odmeny, snažil by sa maximalizovať odmenu v čase *t* aj za cenu nižšieho celkového súčtu odmien za stavy *t* a *t+1*. Naopak, pokiaľ by využil popísanú znalosť, dokázal by napriek dočasnému „poklesu“ dosiahnuť vyšší súčet za odmeny.

Aj keď sú odmeny primárne a hodnoty sekundárne, pretože bez odmien by neboli hodnoty, akcie vykonávané agentom sú založené práve na posúdení hodnôt, čím sa docieli maximalizácia súčtu odmien za dlhšiu dobu. Zatiaľ čo odmena je priamou odpoveďou prostredia, hodnoty musia byť odhadované a prehodnocované na základe pozorovania agenta po celú dobu jeho činnosti. A preto sú metódy na odhadovanie hodnôt najdôležitejším komponentom algoritmov RL.

Posledným elementom niektorých systémov je *model prostredia*, ktorý imituje správanie prostredia, vďaka čomu môžeme predpovedať na základe aktuálneho stavu a akcie ktorá sa vykoná nasledujúcu odmenu a stav, ktorý dostaneme. Modely slúžia na *plánovanie*, čo znamená posúdenie možných budúcich situácií na základe vykonaných akcií, bez toho aby sa skutočné prostredie zmenilo.

2.2 Hodnotenie pomocou spätnej väzby

Ako som už spomenul, najdôležitejšou črtou, ktorá rozlišuje RL od iných typov učenia je to, že hodnotí, pomocou odmeny, dané akcie namiesto toho, aby označil tú správnu, čo vytvára potrebu pre aktívne skúmanie, resp. explicitné hľadanie dobrého konania typu pokus-omyl. Tiež som spomenul, že hodnotenie pomocou spätnej väzby, tj. odmeny, popisuje nakoľko dobrá je daná akcia, ale nie či je najlepšia alebo najhoršia možná. Neskôr popíšem, ako sa táto odmena vypočítava.

2.3 Metódy pre odhad hodnoty akcie

V nasledujúcich kapitolách sa budeme bližšie venovať metódam, ako odhadnúť hodnotu akcií a ako na základe týchto odhadov vybrať akciu. Na to si musíme definovať význam jednotlivých zápisov:

$Q^*(a)$ = označenie skutočnej hodnoty akcie *a*

$Q_t(a)$ = odhadovaná hodnota akcie *a* vykonanej v čase *t*

Ak skutočnou hodnotou akcie je stredná hodnota dosiahnutých odmien, potom je prirodzené odhadovať túto hodnotu ako priemer dosiahnutých odmien. Inak povedané, ak v hre s poradovým číslom t je vybraná akcia a k_a–krát a dosiahnuté odmeny sú $r_1, r_2, r_3, \dots, r_{k_a}$, potom odhadovaná hodnota akcie bude

$$Q_t(a) = \frac{r_1 + r_2 + r_3 + \cdots + r_{k_a}}{k_a}$$

Takýto výpočet odhadu sa nazýva *metóda priemeru vzorky* [sample-average method], ale nejde o jedinú, ani najlepšiu možnú metódu odhadu. V prípade ak $k_a = 0$ potom $Q_t(a)$ nahradíme predvolenou hodnotou, napríklad $Q_0(a) = 0$. Pre $k_a \to \infty$, $Q_t(a)$ konverguje ku $Q^*(a)$.

Po odhade hodnoty akcie nasleduje výber akcie, ktorú agent vykoná. Najjednoduchšie bude vybrať takú akciu, ktorá má maximálnu odhadovanú hodnotu, resp. v hre t vyberieme greedy akciu a^*, *pre ktorú platí* $Q_t(a^*) = max_a Q_t(a)$. Pri tejto metóde sa však vôbec nezisťuje, či náhodou nie je iná akcia lepšia. Preto je lepšie z času na čas, s pravdepodobnosťou ϵ, urobiť náhodný výber, nezávislý na odhade hodnôt akcií. Metódy, ktoré sa takto správajú, nazývame ϵ-greedy metódy.

2.4 Nestacionárne problémy

Ak prostredie vracia pre rovnaké akcie stále tie isté odmeny, patrí do skupiny stacionárnych [stationary] RL problémov. Ak sa tieto odmeny menia s časom, ide o nestacionárne problémy a v takýchto prípadoch je lepšie ováhovať najnovšie odmeny viac ako tie staršie. Jedným z najpopulárnejších spôsobov ako to urobiť, je

použiť konštantnú rýchlosť učenia. Pravidlo inkrementálnej aktualizácie Q_t pre k minulých odmien by tak bolo:

$$Q_{k+1} = Q_k + \alpha[r_{k+1} - Q_k]$$

kde $0 < \alpha \leq 1$ a je konštantné. Toto vedie k tomu, že Q_k je váženým priemerom minulých odmien a počiatočného odhadu Q_0:

$$Q_k = Q_{k-1} + \alpha\,[r_{k+1} - Q_{k-1}]$$

$$= \alpha r_k + (1-\alpha)\,Q_{k-1}$$

$$= \alpha r_k + (1-\alpha)\alpha r_{k-1} + (1-\alpha)^2\,Q_{k-2}$$

$$= \alpha r_k + (1-\alpha)\alpha r_{k-1} + (1-\alpha)^2\,\alpha r_{k-2} + \cdots + (1-\alpha)^{k-1}\,\alpha r_1 + (1-\alpha)^k Q_0$$

$$= (1-\alpha)^k Q_0 + \sum_{i=1}^{k} \alpha(1-\alpha)^{k-i} r_i$$

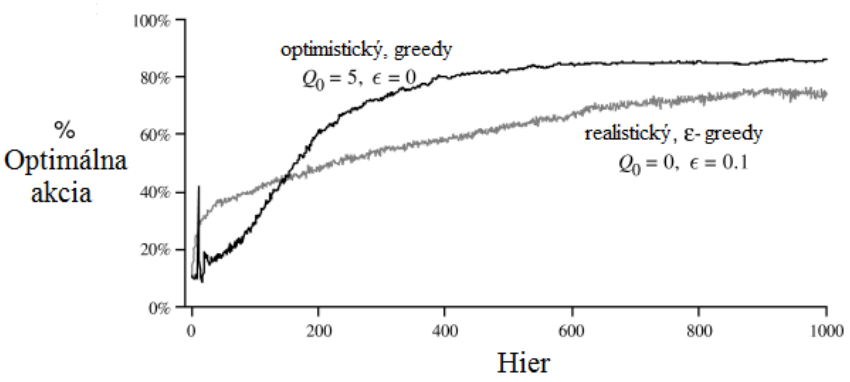
Je zrejmé, že váha $\alpha(1-\alpha)^{k-i}$ daná odmene r_i závisí od počtu predošlých odmien, *k-i*. Hodnota $(1-\alpha)$ je menšia ako 1, takže sa exponenciálne znižuje sila akou sa podieľa táto odmena r_i na odhade nasledujúcej odmeny. Preto sa táto metóda nazýva *priemer s exponenciálnym zabúdaním* [exponential, recency-weighted average].

2.5 Optimistické počiatočné hodnoty

Všetky metódy, ktoré som doteraz opisoval sú do určitej miery závislé na počiatočných odhadoch hodnoty akcie, $Q_0(a)$. Pri metóde priemeru vzorky zanikne táto počiatočná hodnota hneď, ako sú všetky akcie aspoň raz vybrané. Pre metódy s konštantnou a je však táto hodnota permanentná, aj keď sa jej sila oslabuje s časom. V praxi to nie je problém, dokonca to môže byť užitočné. Nevýhodou je, že ich buď vynulujeme alebo budeme od používateľa požadovať ich nastavenie. Svetlou stránkou veci je, že takto môžeme vložiť základné znalosti o očakávaných odmenách.

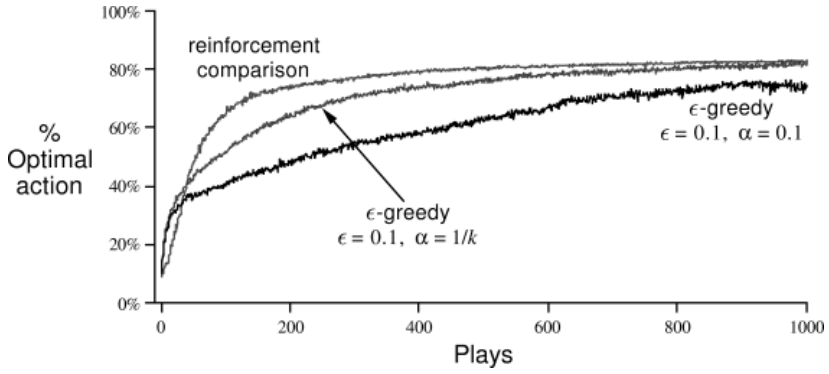
Inou možnosťou je využiť počiatočné hodnoty na posilnenie procesu *skúmania*. Predstavme si, že priemerné hodnoty odmeny dosahujú rozpätie $\langle 0{,}1 \rangle$, potom nastavenie počiatočnej hodnoty na $Q_0(a) = 5$ docieli vybranie akcie a, pretože ak

hodnoty zvyšných akcií sú menšie, je táto akcia greedy akciou a bude vybraná. Keďže ešte nebola predtým vybraná, nemožno jej odhad považovať za znalosť, resp. *využitie* znalosti, tým pádom ide o *skúmanie*. Táto technika na posilnenie skúmania sa nazýva *optimistické počiatočné hodnoty*.



Graf 1: *Konvergencia odhadov ku* $Q(a^*)$ *s rozdielnymi počiatočnými hodnotami. Je vidieť, že metóda s počiatočným nastavením všetkých odhadov na* $Q_0 = 5$ *konverguje rýchlejšie ku* $Q(a^*)$ *(Sutton a Barto, 1998).*

2.6 Porovnávacie metódy učenia posilňovaním



Graf 2: *Porovnanie efektívnosti porovnávacích metód učenia posilňovaním s predošlými metódami (Sutton a Barto, 1998).*

Intuitívne by sa mohlo zdať, že akcie, ktorých odmena je vysoká by sa mali častejšie opakovať ako tie, ktorých odmena je nízka. Na základe čoho však možno povedať, že odmena je vysoká alebo nízka? Očividne potrebujeme nejakú kľúčovú hodnotu, na základe ktorej by bolo spomínané porovnávanie možné, tzv. *referenčnú odmenu*. Je prirodzené dosadiť za ňu priemer dosiahnutých predošlých odmien a stanoviť, že ak

odmena za akciu a je vyššia ako referenčná odmena, potom je odmena vysoká, v opačnom prípade nízka. Metódy učenia založené na tejto myšlienke sa nazývajú *porovnávacie metódy učenia posilňovaním* [reinforcement comparison methods] a sú niekedy efektívnejšie ako metódy typu akcia-hodnota.

Porovnávacie metódy učenia posilňovaním prevažne neudržujú odhady hodnôt akcií, ale len celkovú úroveň odmeny, udržiavajú si len hodnoty preferencií jednotlivých akcií. Takúto preferenciu pre akciu a v čase t označujeme $p_t(a)$. Tieto preferencie môžu byť použité na zistenie pravdepodobnosti výberu akcie použitím softmaxového vťahu:

$$\pi_t(a) = Pr\{a_t = a\} = \frac{e^{p_t(a)}}{\sum_{b=1}^{n} e^{p_t(b)}}$$

kde $\pi_t(a)$ vyjadruje pravdepodobnosť vybrania akcie a v hre s poradovým číslom t. Myšlienka porovnávania je použitá pri aktualizácií preferencie jednotlivých akcií. Po každej hre t je k preferencii $p_t(a_t)$ vybranej akcie a_t prirátaný rozdiel odmeny r_t a referenčnej odmeny $\overline{r}_t$:

$$p_{t+1}(a_t) = p_t(a_t) + \beta[r_t - \overline{r}_t]$$

kde β je pozitívna rýchlosť učenia. Táto rovnica zabezpečí, že vysoké odmeny zvýšia svoju pravdepodobnosť opätovného vybrania, zatiaľ čo nízke odmeny znížia svoju pravdepodobnosť.

Referenčná odmena sa mení s časom nasledovne:

$$\overline{r}_{t+1} = \overline{r}_t + \alpha[r_t - \overline{r}_t]$$

kde α, $0 < \alpha \leq 1$, je tiež rýchlosť učenia. Počiatočná hodnota $\overline{r}_0$ môže byť nastavená ľubovoľne. Počiatočné hodnoty preferencií je lepšie nastaviť na nulu. Použitie rýchlosti učenia je znakom nestacionárnych sa problémov, no tento problém je vo svojej podstate stacionárny. Je to však nutné z dôvodu zmien distribúcie odmien, za ktoré môže zlepšovanie výberu akcií.

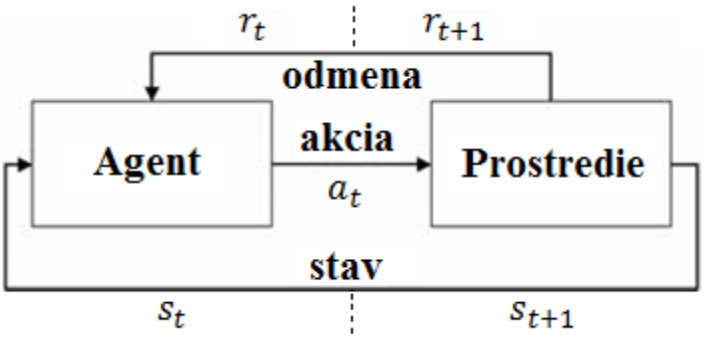
3 Problém učenia posilňovaním

Až do tejto chvíle som používal slovo problém v spojení s RL bez toho aby som ho jasnejšie vysvetlil. V tejto kapitole sa teda budem venovať práve jemu.

3.1 Rozhranie agent-prostredie

Ten kto sa učí a rozhoduje o nasledujúcich akciách sa nazýva *agent*. To s čím interaguje, zahŕňajúc všetko okolo agenta, sa nazýva *prostredie*. Navzájom interagujú kontinuálne tým, že agent vyberá akcie a prostredie vracia agentovi novú situáciu, *stav*. Prostredie taktiež dáva *odmeny*, špeciálne číselné hodnoty, ktoré sa agent snaží maximalizovať počas života. Kompletná špecifikácia prostredia definuje *úlohu*, jednu inštanciu problému učenia posilňovaním.

Interakcia medzi agentom a prostredím prebieha v diskrétnych krokoch, *t=0, 1, 2,...*, pričom v každom kroku, agent dostáva reprezentáciu *stavu* prostredia, $s_t \in \mathcal{S}$, kde $\mathcal{S}$ je množina možných stavov, a na základe tohto stavu vyberá *akciu*, $a_t \in \mathcal{A}(s_t)$, kde $\mathcal{A}(s_t)$ je množina možných akcií v stave s_t. V ďalšom kroku dostane agent číselnú *odmenu*, $r_{t+1} \in \mathcal{R}$, a prejde do nového stavu s_{t+1}.



Obrázok 1: *Interakcia agenta a prostredia v RL*

V každom kroku agent priraďuje stavy prostredia do pravdepodobností vybrania jednotlivých možných akcií. Toto priraďovanie sa volá *stratégia agenta* a označuje sa ako π_t, kde $\pi_t(s, a)$ je pravdepodobnosť že $a_t = a$ ak $s_t = s$. RL metódy špecifikujú, ako agent mení svoju stratégiu vzhľadom na získavané skúsenosti. Úlohou agenta je teda dosiahnuť maximálnu sumu odmien počas behu.

Na zamyslenie zostáva otázka, či je agent s úplnou znalosťou o prostredí schopný vyriešiť akýkoľvek problém v tomto prostredí. Sutton a Barto (1998) tvrdia, že nie a demonštrujú to na Rubikovej kocke, kde ako tvrdia máme znalosť o tom, ako to funguje ale nevieme ju správne poskladať. Dovolím si tvrdiť, že úplná znalosť prostredia nám zaručuje vyriešenie problému v tomto prostredí, pretože ak mám znalosť, ako funguje napríklad Rubikova kocka, tak musím aj poznať taktiku, akou túto kocku správne poskladať (ak táto taktika existuje). Inak nemožno hovoriť o úplnej znalosti prostredia.

3.2 Ciele a odmeny

Vieme už, že RL je cieľovo-orientované učenie sa. Tiež som spomenul odmeny z prostredia, na základe ktorých agent vie, nakoľko dobrá (správna) bola jeho posledná akcia, resp. v pri funkcii hodnoty, nakoľko dobre si doteraz počínal s riešením problému. Ako však vypočítať hodnotu odmeny?

Aby agent dosiahol svoj cieľ, musí odmena zodpovedať cieľu, resp. dosiahnutej úrovni cieľa. Ak by sme si zobrali príklad zo šoférovania po ceste, odmenou by mohlo byť číslo od 0 do 100, kde 0 by znamenala, že agent je mimo cesty a 100, že agent je v strede cesty. Keďže motiváciou agenta je maximalizovať odmenu, snažil by sa udržiavať auto v strede cesty, čo je v tomto prípade optimálne riešenie.

Akousi nadstavbou šoférovania by mohla byť snaha o dosiahnutie časového rekordu na uzavretej trati. Agent by dostával priebežné hodnoty 1 alebo 0 za to, či sa nachádza na ceste alebo nie a na konci okruhu by dostal bodové ohodnotenie svojho času, kde 1000 bodov by bolo za čas 1 sekunda (čo je nedosiahnuteľný čas) a počet bodov by klesal s pribúdaním sekúnd vo výslednom čase. Keby sme agentovi nedávali priebežnú odmenu, skrátil by si cestu po trati prechodom cez trávnik. Taktiež nemôžeme hodnotiť pozíciu auta vzhľadom na stred cesty, pretože najrýchlejšia trajektória nemusí byť vždy tá najkratšia.

V reálnom svete je v prípade biologického agenta výpočet odmeny lokalizovaný v jeho mozgu (resp. v riadiacom orgáne). Časť mozgu vyhodnocuje aktuálny stav a predáva odmenu, napríklad vo forme dopamínu, ďalej do iných častí mozgu.

V našej simulácií bude odmena vyrátavaná prostredím a dodávaná spolu s snovým stavovým signálom.

3.3 Zisk a zrážky

Snahou agenta je maximalizovať sumu odmien, tzv. zisk [return]. Keďže však agent nevie počas svojho života, aký zisk presne dosiahne, hovoríme o *očakávanom zisku*. Ak rozdelíme čas na menšie *časové kroky* [time steps], potom očakávaný zisk zapisujeme ako:

$$\mathcal{R}_t = r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_T$$

a je to suma očakávaných odmien v nasledujúcich časových krokoch po časovom kroku *t*. T je posledný krok v čase. Táto rovnica by bola správna, resp. vypočítateľná, ak by $T \neq \infty$.

Ak možno interakciu agenta s prostredím rozdeliť na menšie časti, resp. sekvencie, potom hovoríme o *epizódach*. Každá epizóda ma konečný počet časových krokov, s jediným koncovým stavom (posledný stav k čase T), teda predchádzajúca rovnica tu má zmysel. Zároveň každý časový krok má svoj stav, kde množina všetkých stavov okrem koncového sa označuje ako $\mathcal{S}$ a s koncovým stavom $\mathcal{S}^+$. Keď skončí epizóda, vynulujú sa všetky dosiahnuté hodnoty, ako napríklad funkcia hodnoty a iné. Úlohy v jednotlivých epizódach nazývame *epizodické úlohy*.

Ak však nemožno rozdeliť interakciu na časti, napríklad agent bude vykonávať svoju činnosť až do konca svojho „života“, potom hovoríme o *pokračujúcich úlohách* [continuing tasks], kde $T = \infty$. Hodnota $\mathcal{R}_t$ by v tomto prípade mohla dosiahnuť nekonečno, čo nie je práve vhodná hodnota. V takomto prípade je nutné upraviť výpočet $\mathcal{R}_t$ tak, že pridáme *parameter kvantifikujúci mieru zohľadnenia budúcich odmien* [discount rate] γ, kde $0 \leq \gamma \leq 1$:

$$\mathcal{R}_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Ak $\gamma < 1$, potom táto rovnica vracia konečnú hodnotu. Z tejto rovnice je tiež zrejmé, že ak $\gamma = 0$, potom agent maximalizuje funkciu odmeny, čiže okamžitú odmenu, a to, ako sme už spomenuli, nemusí viesť k najvyššiemu zisku.

3.4 Markovovská vlastnosť a stavový signál

Predstavme si agenta, ktorého úlohou je balansovať s paličkou tak, aby nepadla na zem. Čo taký agent potrebuje, aby splnil svoj cieľ? Potrebuje aktuálny stav prostredia a na základe neho vykonať správne rozhodnutie. Až doteraz sme sa nezaoberali tým, ako má vyzerať *stavový signál*, ktorý popisuje prostredie, resp. jeho stav. Prirodzene nám napadne, že stavový signál by mal obsahovať všetky *potrebné* a agentovi *dostupné* informácie na opísanie aktuálneho stavu. Slovo potrebné sme zvýraznili z toho dôvodu, že na rozdiel od svojho biologického modelu, štandardný RL nedokáže filtrovať vstupné (senzorické) údaje. Tým pádom treba ručne navrhnúť, ktoré údaje sú a ktoré nie sú potrebné. Taktiež slovo dostupné údaje nie je zbytočné. Pri hraní kariet má agent k dispozícii údaje o kartách ktoré má na ruke, ale nemá informáciu o tom, ktorá karta bude nasledujúca, aj keď táto informácie je prítomná v prostredí, ale je nedostupná.

Stavový signál nemusí vždy popisovať len aktuálne prostredie, môže mať formu predspracovanej (nejakým iným systémom) informácie, napríklad 360° pohľad sa dá urobiť jedine zozbieraním senzorických údajov behom časového úseku, za ktorý sa agent otočí o 360°. Napriek tomu, že senzorická informácia by bola vo forme obrázku, popisujúci časť prostredia, stavový signál bude obsahovať kompletný obraz okolia.

Ideálny stavový signál je taký, ktorý vhodne sumarizuje predošlé senzorické informácie tak, aby podstatné informácie zostali zachované, čo však už z definície vyžaduje viac ako len okamžitý senzorický vstup, ale v najhoršom prípade kompletnú históriu senzorických informácií. Pri balansovaní s paličkou nám teda stačí smer, ktorým palička padá a uhol voči zvislej polohe, pod ktorým je aktuálne natočená.

Takýto signál sa nazýva Markovovský (Sutton a Barto, 1989), resp. majúci Markovovskú vlastnosť, ktorá sa definuje pomocou úplnej distribúcie pravdepodobnosti nasledovne:

$$Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\}$$

pre všetky s', r, s_t, a_t. Za normálnych okolností všetky predchádzajúce kroky ovplyvňujú nasledujúce a preto možno napísať:

$$Pr\{s_{t+1} = s', r_{t+1} = r | s_t, r_t, a_t, s_{t-1}, r_{t-1}, a_{t-1}, \dots r_1, s_0, a_0\}$$

pre všetky $s', r, s_t, r_t, a_t, s_{t-1}, r_{t-1}, a_{t-1}, \dots r_1, s_0, a_0$. Ak je teda stavový signál Markovovský, potom sa predchádzajúce dve distribúcie musia rovnať. Predchádzajúce dve rovnice sú platné len pre prípad konečného počtu stavov a odmien, v opačnom prípade by sme museli používať integrály, ale pre túto prácu to nie je nutné. Takže takýto signál plne postačuje na výber nasledujúcich akcií. Aj napriek tomu, že stavy nie sú Markovovské, je výhodné rátať s nimi ako keby boli, resp. povedať, že sa približujú k ideálnym stavom.

3.5 Markovov rozhodovací proces

Markovov rozhodovací proces (MDP) je označenie takých úloh v RL, ktoré spĺňajú Markovovskú vlastnosť, čiže nasledujúca akcia (a_{t+1}) a stav (s_{t+1}) v čase *t* nezávisia od predchádzajúcich akcií, stavov a odmien. Ak je množina stavov a akcií konečná, potom ide o konečný Markovovský rozhodovací proces (Sutton a Barto, 1989).

Pravdepodobnosť výskytu nasledujúceho stavu s' za súčasného stavu s a následnej vykonanej akcii a, označovaná ako *prechodové pravdepodobnosti* [transition probabilities], je:

$$\mathcal{P}_{ss'}^{a} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$$

Podobne vyzerá aj rovnica pre očakávanú hodnotu nasledujúcej odmeny ak poznáme s, a a tiež nasledujúci stav s':

$$\mathcal{R}_{ss'}^{a} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\}$$

Tieto dve rovnice kompletne popisujú dôležité aspekty dynamiky končeného MDP.

3.6 Funkcie hodnoty

Popísal som funkciu hodnoty, ako jeden z elementov RL, no doteraz nebolo spomenuté ako sa vypočíta. Funkcia hodnoty vyjadruje nakoľko dobrý je aktuálny stav, resp. nakoľko dobrá je akcia v aktuálnom stave, čo sa týka budúcich dosahovaných odmien. Keďže funkcia hodnoty vypočítava hodnoty nasledujúcich stavov zo stavu s, je logické, že bude záležať na stratégii, π. Stratégia priraďuje stavy s na akcie a, čo zapisujeme ako $\pi(s, a)$. Potom sa funkcia hodnoty aktuálneho stavu, V, pre stratégiu π, počnúc stavom s, zapisuje ako $V^{\pi}(s)$ a vypočítava sa nasledovne:

$$V^{\pi}(s) = E_{\pi}\{\mathcal{R}_t | s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \, | s_t = s\right\}$$

V^{π} sa tiež nazýva aj *funkcia hodnoty stavu pre stratégiu* π [state-value function for policy π]. E_{π} označuje očakávanú hodnotu, ak agent nasleduje stratégiu π, kde t je ľubovoľný časový krok.

Podobne sa vypočíta aj funkcia hodnoty v stave s pre akciu a pri stratégii π:

$$Q^{\pi}(s, a) = E_{\pi}\{\mathcal{R}_t | s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \, | s_t = s \; a_t = a\right\}$$

Q^{π} sa nazýva *funkcia hodnoty akcie pre stratégiu* π [action-value function for policy π].

Podstatnou vlastnosťou funkcií hodnôt v RL a dynamickom programovaní je, že spĺňajú čiastočné rekurzívne vzťahy, čo znamená že pre ľubovoľnú stratégiu π a stav s možno funkciu hodnoty pre stratégiu π v stave s zapísať nasledovnom tvare:

$$V^{\pi}(s) = E_{\pi}\{\mathcal{R}_t | s_t = s\}$$
$$= E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \, | s_t = s\right\}$$

$$= E_\pi \left\{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \,|s_t = s \right\}$$

$$= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \,|s_{t+1} = s' \right\} \right]$$

$$= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^\pi(s') \right]$$

Táto rovnica je *Bellmanovou rovnicou funkcie hodnoty stavu pre stratégiu* π a vyjadruje vzťah aktuálneho stavu s a jeho nasledovníkov.

3.7 Optimálne funkcie hodnoty

Vieme už, že podstatou RL je maximalizovať funkciu hodnoty, ktorá je ovplyvnená vykonaním vybraných akcií v nasledujúcich stavoch. Výber akcie má za úlohu stratégia, no nie každá stratégia vedie k maximálnemu zisku. Potom musí existovať taká stratégia π, ktorej funkcia hodnoty je väčšia alebo rovná ako ľubovoľná iná stratégia π' zo zvyšných politík, čo zapisujeme ako $V^\pi(s) \geq V^{\pi'}(s)$, pre všetky $s \in \mathcal{S}$. Stratégia π je lepšia alebo rovnaká ako stratégia π', práve vtedy ak platí $V^\pi(s) \geq V^{\pi'}(s)$. Vo všeobecnosti platí, že existuje najmenej jedna stratégia lepšia alebo rovná ako zvyšné stratégie a tú nazývame *optimálna stratégia*, ktorú, resp. ktoré v prípade ak je ich viac, označujeme ako π^*. Všetky optimálne stratégie majú rovnakú funkciu hodnoty, tzv. *optimálnu funkciu hodnoty stavu*, označovanú ako V^*:

$$V^*(s) = \max_\pi V^\pi(s), \qquad s \in \mathcal{S}$$

Podobne je to aj s optimálnou funkciou hodnoty akcie:

$$Q^*(s,a) = \max_\pi Q^\pi(s,a), \qquad s \in \mathcal{S}, a \in \mathcal{A}$$

Keďže výber akcií a v stave s sa riadi optimálnou stratégiou a funkcia (s,a) nám vráti očakávaný zisk, potom možno zapísať:

$$Q^*(s,a) = E\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\}$$

Napriek tomu, že sme si definovali pojmy optimálnosti funkcie hodnoty a optimálnej stratégie, sú tieto v skutočnosti veľmi ťažko dosiahnuteľné, ak vôbec. Veľkým obmedzením je výpočtová a pamäťová náročnosť, čo sa najviac zvýrazňuje pri použití interného modelu prostredia. V praxi sa však stačí priblížiť čo najviac k týmto optimálnym hodnotám, resp. funkciám.

Podobne ako existuje Bellmanova rovnica funkcie hodnoty pre stratégiu π, existuje aj *Bellmanova optimálna rovnica pre* V^*, ktorá sa však môže zapísať vo forme neobsahujúcej stratégiu, π:

$$
\begin{aligned}
V^*(s) &= \max_{a \in \mathcal{A}(s)} Q^{\pi^*}(s,a) \\
&= \max_a E_{\pi^*}\{\mathcal{R}_t | s_t = s, a_t = a\} \\
&= \max_a E_{\pi^*}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \, | s_t = s, a_t = a\right\} \\
&= \max_a E_{\pi^*}\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \, | s_t = s, a_t = a\right\} \\
&= \max_a E\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\} \\
&= \max_{a \in \mathcal{A}(s)} \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^*(s')\right]
\end{aligned}
$$

Bellmanova optimálna rovnica pre Q^*:

$$
\begin{aligned}
Q^*(s,a) &= E\left\{r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \, | s_t = s, a_t = a\right\} \\
&= \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q^*(s', a')\right]
\end{aligned}
$$

Tieto vzťahy využijeme pri dynamickom programovaní, ktoré si popíšeme neskôr.

4 Základné metódy riešenia

Až doteraz som sa zaoberal základnými elementmi RL, ukázal som, ako sa vypočítajú a definoval som si nové pojmy. Najviac sme sa však venoval funkciám hodnôt, pretože sa v RL používajú na hľadanie dobrých politík rozhodovania najčastejšie. V nasledujúcej časti popíšem v rýchlosti tri základné triedy riešenia problému učenia posilňovaním:

- Dynamické programovanie
- Monte Carlo metódy
- Učenie na základe časových rozdieľov [Temporal Difference learning] (TD učenie)

Každá z týchto tried má svoje pre a proti. Napríklad dynamické programovanie zodpovedá skupine algoritmov schopných vypočítať optimálnu stratégiu dokonalého modelu prostredia ako Markovovský rozhodovací proces (MRP) a je matematicky dobre rozpracované. Vyžaduje ale kompletný a presný model prostredia. Klasické algoritmy dynamického programovanie majú svoje obmedzené využitie v RL jednak pre svoju výpočtovú náročnosť a jednak preto, že vyžadujú dokonalý model prostredia. Napriek tomu sú dôležité teoreticky.

Na rozdiel od dynamického programovania, Monte Carlo metódy nevyžadujú kompletnú znalosť o prostredí, pretože sú založené na skúsenostiach, čo v tomto prípade znamená použitie priemeru z viacerých predošlých hodnôt.

Podobne ako Monte Carlo metódy, aj TD učenie využíva skúsenosti na predikciu. Rozdiel medzi nimi je v tom, že zatiaľ čo TD učenie upravuje hodnoty priebežne, resp. vždy v nasledujúcom časovom kroku, Monte Carlo metódy robia tieto úpravy až po skončení epizódy.

Ak rovnica pre výpočet funkcie hodnoty stavu pre Monte Carlo metódy vyzerá nasledovne:

$$V(s_t) = V(s_t) + \alpha[\mathcal{R}_t - \mathrm{V}(s_t)]$$

a zároveň vieme z predchádzajúcich kapitol, že:

$$V^{\pi}(s) = E_{\pi}\{\mathcal{R}_t | s_t = s\}$$
$$= E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \, | s_t = s\right\}$$
$$= E_{\pi}\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \, | s_t = s\right\}$$
$$= E_{\pi}\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s\}$$

potom na základe toho možno R_t , súčet všetky odmien od časového kroku t, nahradiť $r_{t+1} + \gamma V_k(s_{t+1})$, čiže dostaneme rovnicu:

$$V(s_t) = V(s_t) + \alpha[r_{t+1} + \gamma \mathrm{V}(s_{t+1}) - \mathrm{V}(s_t)]$$

čo je rovnaké ako pri dynamickom programovaní. Toto je najjednoduchšia metóda a nazýva sa *TD(0)*. Vyjadrenie $r_{t+1} + \gamma \mathrm{V}(s_{t+1}) - \mathrm{V}(s_t)$ nazývame *TD chybovosťou* [TD error] alebo *TD singálom* [TD signal] a označuje sa ako δ_t.

Keďže TD učenie nevyžaduje model prostredia je teda výhodnejšie ako dynamické programovanie. Otázkou zostáva či je výhodnejšie ako Monte Carlo metódy, čo je ale otvorená otázka. Pri dlhších epizódach je logickejšie použiť TD učenie, hlavne ak ide o kritické problémy, napríklad agent si nemôže dovoliť robiť priveľa chýb, preto je lepšie učenie za behu. Otvorenou otázkou však zostáva, ktorá z týchto dvoch metód konverguje k optimálnemu riešeniu rýchlejšie. (Sutton a Barto, 1998).

4.1 On-policy a Off-policy metódy

Jedinou možnosťou ako zabezpečiť, že sa všetky akcie vyberú je pokračovať v epizódach, resp. vo výbere akcií. Poznáme dva druhy metód, ktoré nám toto zabezpečia a to sú *on-policy* a *off-policy* metódy. On-policy aktívne vyhodnocuje a vylepšuje stratégiu π, zatiaľ čo off-policy používa dve stratégie, kde jednu vylepšuje a druhou sa riadi.

Do on-policy patria, okrem iného, aj ε-soft stratégie, založené na metóde ε-greedy, ktorú sme si už predstavili dávnejšie v časti 2.3. Slovo „soft“ v názve označuje tie stratégie, ktorých $\pi(s, a) > 0$. Pre naše účely si definujeme, že pravdepodobnosť ne-

greedy akcií bude rovná $\frac{\varepsilon}{|\mathcal{A}(s)|}$, čiže podiel hodnoty ε a počtu možných akcií v stave s, a že pravdepodobnosť výberu greedy akcie bude $1-\varepsilon+\frac{\varepsilon}{|\mathcal{A}(s)|}$, čo je v podstate doplnok do čísla 1.

ε-soft stratégia vychádza z princípu metódy Monte Carlo ES. Rozdiel je len v tom, že vybraná akcia a v stave s, $\pi(s)$, nebude vždy greedy akcia, ale môže ňou byť aj ne-greedy, s pravdepodobnosťou $\frac{\varepsilon}{|\mathcal{A}(s)|}$. Takto sme eliminovali potrebu začatia epizódy vybranou dvojicou stav-akcia.

Druhým druhom metód sú off-policy metódy. Ako som už spomenul na začiatku, využívajú dve stratégie, $\pi \neq \pi'$. Zatiaľ čo sa epizóda vyvíja pri stratégii π', tzv. *stratégia správania* [behavior policy], na základe odmien upravujeme druhú stratégiu, π, tzv. *stratégiu odhadu* [estimation policy]. Výhodou takéhoto rozdelenia je fakt, že stratégia odhadu môže byť deterministická, zatiaľ čo stratégia správania môže byť nastavená tak, aby skúsila všetky akcie.

Podmienkou pre tieto dve stratégie je, aby akcia v stratégii odhadu π bola vyberateľná s nejakou pravdepodobnosťou väčšou než nula aj v stratégii správania π', formálne $\pi(s,a) > 0 \Rightarrow \pi'(s,a) > 0$. Ako už bolo spomenuté, epizódy sa generujú podľa stratégie π'. Uvažujme úplnú (tj. až do konca) sekvenciu stavov, ktorá nasleduje po i-tom prvom výskyte stavu s. Pravdepodobnosť výskytu tejto sekvencie pri stratégii π označme ako $p_i(s)$ a pri stratégii π' ako $p_i'(s)$. Odmenu, ktorú by sme získali pri tejto sekvencii označme $R_i(s)$ a celkovo je takýchto odmien pre tento stav, ale rozdielne i, n_s. Potom odhadovaná hodnota stavu s je:

$$V(s) = \frac{\sum_{i=1}^{n_s} \frac{p_i(s)}{p_i'(s)} R_i(s)}{\sum_{i=1}^{n_s} \frac{p_i(s)}{p_i'(s)}}$$

Žiaľ nepoznáme hodnoty $p_i(s)$ ani $p_i'(s)$. Poznáme však ich podiel, pretože ak:

$$p_i(s_t) = \prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k)\, \mathcal{P}_{s_k s_{k+1}}^{a_k}$$

potom

$$\frac{p_i(s)}{p_i^{'}(s)} = \frac{\prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k)\, \mathcal{P}_{s_k s_{k+1}}^{a_k}}{\prod_{k=t}^{T_i(s)-1} \pi^{'}(s_k, a_k)\, \mathcal{P}_{s_k s_{k+1}}^{a_k}} = \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi^{'}(s_k, a_k)}$$

čo už poznáme.

4.1.2 Metóda SARSA

Prvou z metód TD učenia je SARSA a patrí do kategórie on-policy metód, čiže tých, ktoré vyhodnocujú a vylepšujú tú istú stratégiu. Princípom metódy SARSA je vyhodnocovanie funkcie hodnoty akcie pre stratégiu π, čo je rozdiel oproti predchádzajúcej kapitole, kde sme brali do úvahy funkcie hodnoty stavu.

SARSA vyhodnocuje funkcie hodnoty akcie pre všetky stavy a akcie. V implementačnom ponímaní by sme mali dvojrozmerné pole, kde prvý rozmer je stav, resp. množina všetkých stavov, a druhý rozmer je akcia, resp. množina všetkých akcií pre daný stav. Počiatočná hodnota $Q(s, a)$ sa môže zvoliť náhodne. Potom rovnica pre zmenu hodnoty $Q(s, a)$ by vyzerala nasledovne:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Túto zmenu vykonávame až pokým nedorazíme do konečného stavu. Ak by bol stav s_{t+1} konečný, potom je hodnota $Q(s_{t+1}, a_{t+1}) = 0$. Názov SARSA vznikol z tohto, že táto rovnica využíva 5 premenných (s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}). Stratégiou by mohla byť napríklad greedy metóda, resp. ε-greedy.

4.1.3 Metóda Q-učenia

V roku 1989 vytvoril C.J. Watkins vo svojej doktorandskej práci novú metódu s menom *Q-učenie* [Q-learning]. Na rozdiel do metódy SARSA, patrí Q-učenie do kategórie off-policy metód. Jeho najjednoduchšou formou je jednokrokové Q- učenie [one-step Q-learning]:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Výber akcie a v stave s, čiže stratégia, je nezávislý od Q a jediné čo musí dodržať pre konvergenciu k Q^* je, že sa všetky páry (s, a) neustále upravujú, resp. že nezanedbáva ani jeden z nich, na čo možno využiť metódu ε-greedy.

4.1.4 R-učenie pre pokračujúce úlohy

Až doteraz sme spomínali také TD metódy, ktoré predpokladali, že úlohy sú epizodické. Teraz si predstavíme metódu pre neepizodické, pokračujúce úlohy, ktorá sa nazýva R-učenie a patrí do kategórie off-policy metód. Kedže sa R-učenie používa pre neepizodické úlohy, bude úlohou tejto metódy maximalizovať odmenu za časový krok. Zadefinujme si teda priemernú očakávanú odmenu za časový krok pri stratégii π ako:

$$\rho^{\pi} = \lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} E_{\pi}\{r_t\}$$

čo je v podstate súčet očakávaných odmien za n časových krokov delené počtom týchto krokov. V podstate z ktoréhokoľvek stavu dosiahneme rovnakú priemernú odmenu, no záleží na tom, ako rýchlo túto hodnotu dosiahneme, resp. určité poradie stavov dosiahne lepšie odmeny za kratší čas a naopak. A keďže je tu táto relativita voči priemerným odmenám, upravíme funkcie hodnôt nasledovne:

$$\widetilde{\mathrm{V}}^{\pi}(s) = \sum_{k=1}^{\infty} E_{\pi}\{r_{t+\mathrm{k}} - \rho^{\pi} | s_t = s\}$$

$$\widetilde{\mathrm{Q}}^{\pi}(s, a) = \sum_{k=1}^{\infty} E_{\pi}\{r_{t+\mathrm{k}} - \rho^{\pi} | s_t = s, a_t = s\}$$

Tieto rovnice nazývame *relatívne hodnoty*. Keďže ide o off-policy metódu, je nutné udržiavať si 2 stratégie, stratégiu správania a stratégiu odhadu. Stratégia správania by mohla byť ε-greedy zameraná na funkcie hodnoty. Zatiaľ je však táto metóda viac experimentálna ako prebádaná a tak k nej treba aj pristupovať.

4.2 Záver

Okrem spomínaných metód existujú ešte aj ďalšie, z ktorých niektoré sú len vylepšené tie metódy, ktoré som v tejto práci opísal, ako napríklad TD(λ) a SARSA(λ). Na implementáciu riadenia automobilu my však plne postačuje metóda SARSA(0), ktorú som už opísal.

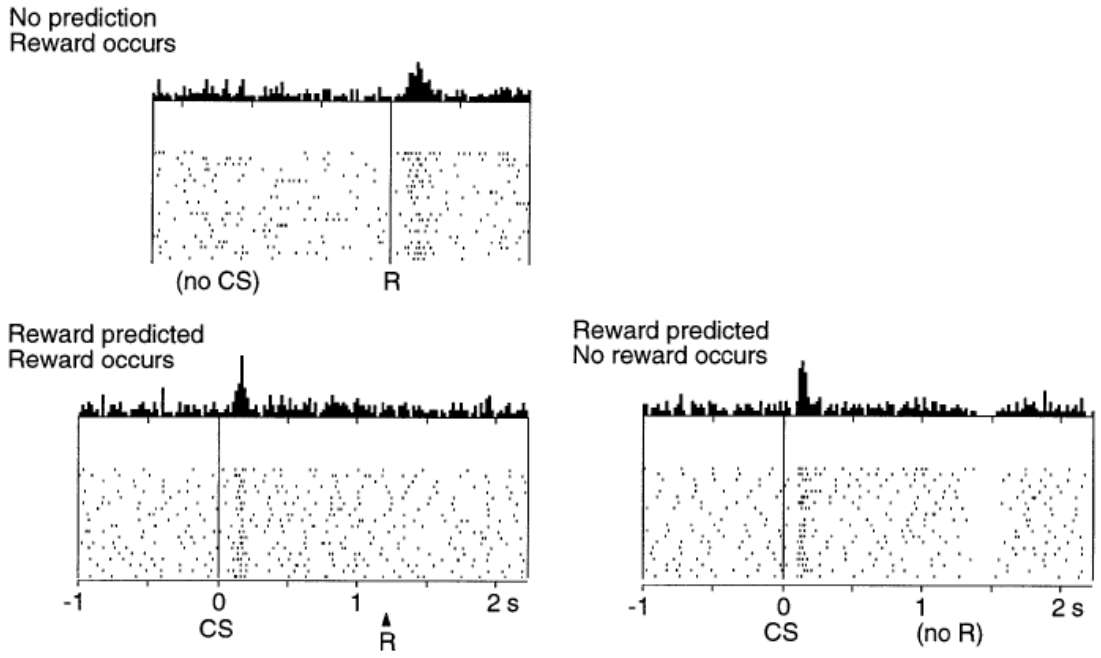
5 Učenie posilňovaním vo vyšších organizmoch

Na začiatku tejto práce som spomínal, že RL vychádza zo svojho biologického modelu. Doteraz som sa venoval len RL v strojom učení. Preto v tejto kapitole vysvetlím, ako je to v skutočnosti v ľudskom mozgu, resp. v mozgu cicavcov.

Asi najviac sa v súvislosti s R pri živočíchoch hovorí o *dopamíne*. Dopamín je neuromodulátor, ktorý vzniká v hypotalame a hrá dôležitú úlohu v motorických funkciách mozgu, pri spracovaní informácií v čelnom laloku a pri pozornosti. Jeho nedostatok spôsobuje choroby ako Parkinsonová choroba, schizofrénia a iné (Cummings, 1999).

Na základe pozorovaní sa predpokladá, že dopamín úzko súvisí s odmenami. Túto teóriu podporujú experimenty s užívaním drog, najmä kokaínu a amfetamínov, kde tieto buď podporili tvorbu a následné uvoľnenie dopamínu v mozgu a tým vyvolali pocit šťastia, nabudenia, fascinovania z okolitých vecí u testovaného subjektu, alebo samé sa ako dopamín správali, čo viedlo k rovnakej reakcii. Ďalšie experimenty sa zamerali na *stimuláciu odmien v mozgu* [brain stimulation reward], kde boli u zvierat elektricky stimulované určité časti mozgu patriace do *dopamínového okruhu* [dopamine-related circuitry], čo malo za následok uprednostňovanie tejto formy odmeny pred prirodzenou odmenou, napríklad z konzumácie potravy (Murray a Shizgal, 1996).

Alternatívna hypotéza pripisuje dopamínu funkciu indikácie *chyba predikcie odmeny* [reward prediction error] (Schultz, 1998). Na základe tejto hypotézy bola v dopamínových neurónoch nameraná aktivita pri výskyte odmeny v čase t (obrázok 2). Ak bola táto odmena podmienená inou udalosťou, stimulom, v čase $t - x$, napríklad zapnutím svetla v miestnosti, presunula sa aktivita dopamínových neurónov z času výskytu odmeny t do času výskytu podmienenej akcie $t - x$. V prípade, ak sa nedostavila odmena po výskyte podmienenej akcie v čase $t - x$, došlo k potlačeniu aktivity dopamínových neurónov v predikovanom čase výskytu odmeny, teda v čase t (Schultz, 1997).



Obrázok 2: *Aktivita dopamínových neurónov (Schultz, 1997)*

Ak si spomenieme na rovnicu TD signálu $r_{t+1} + \gamma \mathrm{V}(s_{t+1}) - \mathrm{V}(s_t)$, tak práve táto rovnica sa správa veľmi podobne ako sa správajú dopamínové neuróny. Ak totiž nie je prítomný žiaden podmieňovací stimul, ani nie je očakávaná žiadna odmena, potom je hodnota TD signálu rovná r_{t+1}. Ak je však očakávaná nejaká odmena, potom sa hodnota TD signálu zmení v rovnakom smere, v akom bude odmena (kladná odmena spôsobí kladnú hodnotu TD signálu, a naopak), aj keď táto odmena zatiaľ neprišla, čo nám zaručuje argument $\mathrm{V}(s_{t+1})$. TD signál sa správa rovnako aj v prípade, že odmena bola očakávaná, ale neprišla, čo nám zas zaručuje rozdiel $\gamma \mathrm{V}(s_{t+1}) - \mathrm{V}(s_t)$, pretože dochádza k potlačeniu, resp. k zníženiu hodnoty TD signálu (Doya, 2007).

Všetky experimenty na túto tému naznačujú, že dopamínové neuróny nesúvisia so samotnou odmenou, ako skôr s jej predikciou. Presnejšie, že tieto neuróny nesú chybový signál na predikciu budúcich odmien (Montague, 1996).

Sú tu však aj iné hypotézy, založené na prvotných experimentoch, ktoré asi najviac odporujú doteraz spomínanému modelu. Tieto, na rozdiel od predošlých hypotéz, pripisujú dopamínu inú funkciu ako predikciu odmien. Podľa nich dopamín reaguje na nové stimuly či stimuly, ktoré síce zdieľajú kvality so stimulom predpovede odmeny, ale samé o sebe nepredpovedajú túto odmenu (Horvitz, 2000). Horvitz rozšíril

myšlienku, že dopamínové odozvy v konzumačných situáciách sú len špeciálnym prípadom širšej a viac motivačne neutrálnej funkcie, v ktorej sa dopamín aktivuje pre všetky zvláštne či prekvapujúce udalosti nehladiac na ich motivačnú hodnotu.

Jedným zo spomínaných prvotných experimentov, bolo pozorovanie excitácie dopamínových neurónov pri novom neutrálnom stimule v Schultzovom laboratóriu (Ljungberg, 1992). Neskôr bol tento jav preskúmaný podrobnejšie experimentmi na mačkách (Horvitz, 1997). Zistilo sa, že úroveň odozvy úzko súvisí s výnimočnosťou stimulu, kde takýto stimul stupňoval silu odozvy. A naopak, pokiaľ sa jednalo o opakovaný stimul, sila odozvy klesala. Tento jav bol pre Horvitza základom jeho pozornostnej hypotézy.

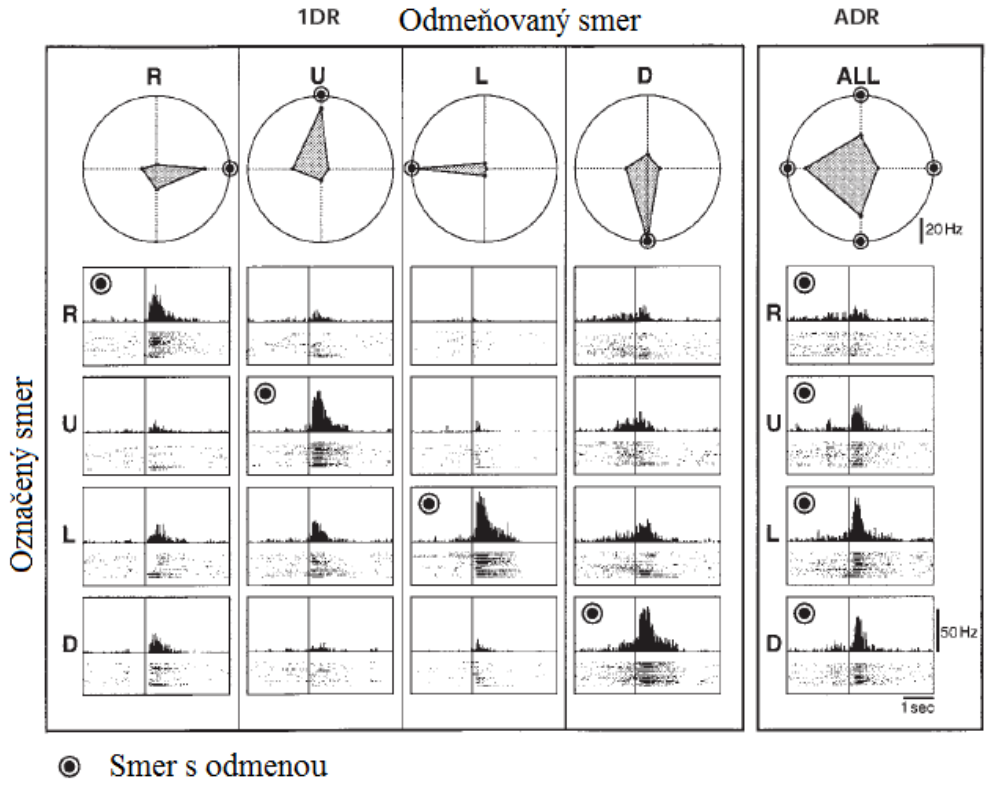
V protiklade s tým, Kakade a Dayan (2002) spájajú dopamínovú odozvu na novosť [novelty] s tým, ako pracuje RL, pri ktorom sa dochádza k rozhodovaniu medzi skúmaním nových stavov a využívaním naučených stavov. Štandardným riešením je pridať fiktívnu odmenu (tzv. bonus) do rovnice, čím sa podporí skúmanie, čo má však za následok skreslenie optimálnej politiky. Kakade a Dayan prišli s viac premyslenou verziou, v ktorej je tento bonus vypočítaný ako rozdiel funkcie ϕ, resp. ϕ_s, ktorá vyjadruje nakoľko je stav s nový. Chybový signál sa potom vypočíta nasledovne:

$$\delta_t = r_t + V_{s_{t+1}} - V_{s_t} + \phi_{s_{t+1}} - \phi_{s_t}$$

Tento spôsob by tak nemal byť skresľujúci, pretože kladný bonus, pri novom stave, je neskôr eliminovaný negatívnym bonusom, keďže stav už nebude nový.

5.1 Kódovanie akcia-hodnota v corpus striatum

Ďalšia z hypotéz pripisuje metódu TD učenia bazálnym gangliám, a preto boli vykonané rôzne experimenty, behom ktorých sa merala aktivita neurónov v corpus striatum (Kawagoe, 1998). Táto aktivita by mala zodpovedať zmenám v predpovedaní odmeny. Corpus striatum je časť okruhu v bazálnych gangliách skladajúceho sa z paralelných okruhov vedúcich z kôry, cez striatum, globus pallidus a talamus späť do kôry.



Obrázok 3: *Aktivita neurónov v corpus striatum; 1DR stĺpce (Right, Up, Left, Down) naľavo zodpovedajú štyrom smerom, ktoré mal subjekt označiť, aby dostal odmenu pri jednosmernom odmeňovaní. Riadky označujú smer, ktorý subjekt označil za správny. Je vidno, že aktivita neurónov je najsilnejšia pri tej akcii, resp. výbere smeru, ktorý bol odmeňovaný. Tento smer je označený aj terčíkom. Naopak, pokiaľ sa jednalo o všesmerné odmeňovanie ADR, čiže subjekt bol odmeňovaný vždy, keď vybral správny smer, je vidieť väčšiu aktivitu pri ľavom smere ako pri zvyšných. V každom stĺpci sa nachádza aj polárny diagram (kruh), ktorý vyjadruje intenzitu aktivity neurónov v každom smere (Kawagoe, 1998).*

Kawagoe (1998) vo svojej práci opisuje jeden z týchto experimentov na dvoch opiciach. Úlohou testovaných subjektov bolo nasmerovať svoj pohľad na obrazovke tým smerom, kde zablikal štvorec a svoj výber smeru potvrdiť žmurknutím, takže šlo zároveň o testovanie vizuálnej pamäte. Odmenou za správnu odpoveď bolo pitie. Keďže schopností opíc sú obmedzené, boli na výber len 4 smery, a to hore, dole, vľavo, vpravo. Testované subjekty boli trénované na zmenu orientácie ich pohľadu za dvoch podmienok odmeňovania: *všesmerné odmeňovanie* (ADR) a *jednosmerné*

odmeňovanie (1DR). Pri všesmernom odmeňovaní, bol testovaný subjekt odmeňovaný pri každom správnom výbere orientácie pohľadu, to znamená pri pohľade smerom k svietiacemu objektu. Pri jednosmernom odmeňovaní bol subjekt odmenený len pri jednom smere, čo znamená, že aj keď jeho pohľad smeroval správnym smerom, tak pokiaľ sa nejednalo o dopredu, vedcami, zvolený smer, odmena sa nedostavila vôbec alebo bola minimálna. V testovaní sa pokračovalo, až keď subjekt určil správny smer, v opačnom prípade sa čakalo na správnu „odpoveď“. Podstatou teda bolo jednak skúmať pamäť a jednak zistiť schopnosti predikcie odmeny.

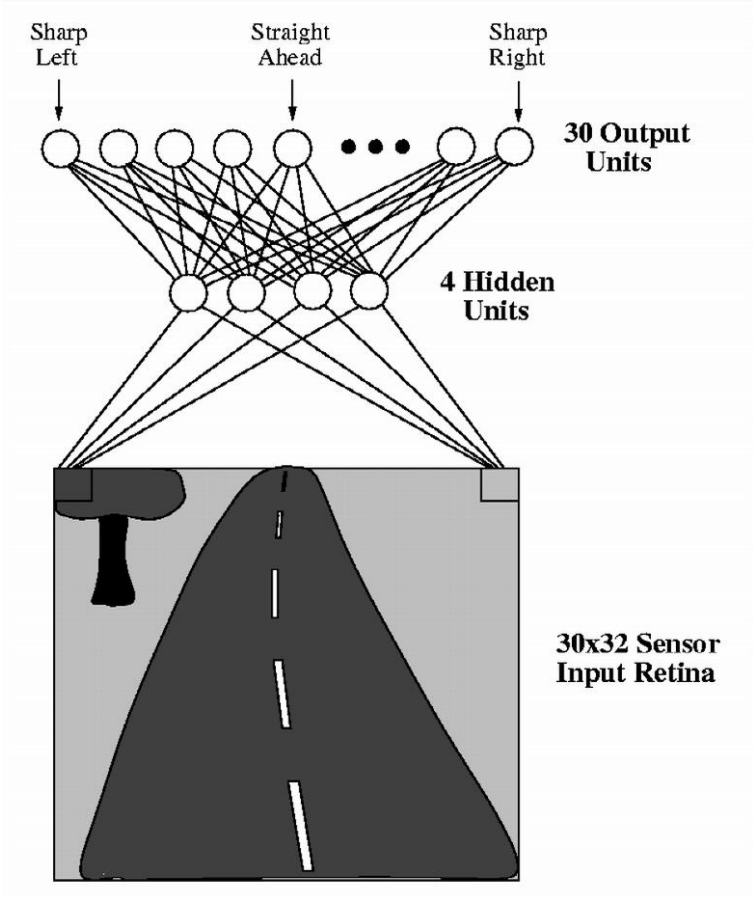
Pri 1DR testované subjekty reagovali rýchlejšie na ten smer, ktorý vracal odmenu. Na obrázku 3 je vidieť, že pri 1DR bola aktivita neurónov v corpus striatum silnejšia pri predpovedaní odmeny po určení správneho smeru než je tomu pri ADR. Pri ADR mali neuróny svoju orientačnú preferenciu, čiže skupina neurónov reagovala len na určitý smer, iná skupina na ďalší smer atď. ..., pretože každý smer vracal odmenu. Naopak, pri 1DR reagovali všetky neuróny prevažne na jeden smer a to práve na ten, ktorý vracal odmenu.

Toto pozorovanie naznačuje, že tieto neuróny nielenže reagujú na samotnú akciu, ale taktiež ich excitácia úzko súvisí s odmenou, čo súvisí s významom funkcie hodnoty akcie.

Na základe týchto zistení sa zmenil pohľad na význam bazálnych ganglií a to je modulácia signálov z motorickej kôry na ich ceste k alfa motorickým neurónom. Teraz je jasné, že sa tiež významne podieľajú na riadení správania založeného na odmenách, a taktiež dokážu tieto odmeny predpovedať, čo je nutný predpoklad pre úspešné riadenie. Za všetky tieto pozorovania vďačíme čoraz dokonalejším a presnejším zobrazovacím technológiám, ako sú FMRI a PET.

6 Implementácia učenia posilňovaním

Až doteraz som vysvetľoval čo je to RL, aké sú jeho hlavné metódy a ako súvisí s biologickým učením. V tejto kapitole začnem s prvou implementáciou jednej z metód učenia na riadenie automobilu a postupne ju budem vylepšovať.



Obrázok 4: *Neurónová sieť použitá v projekte ALVINN.*

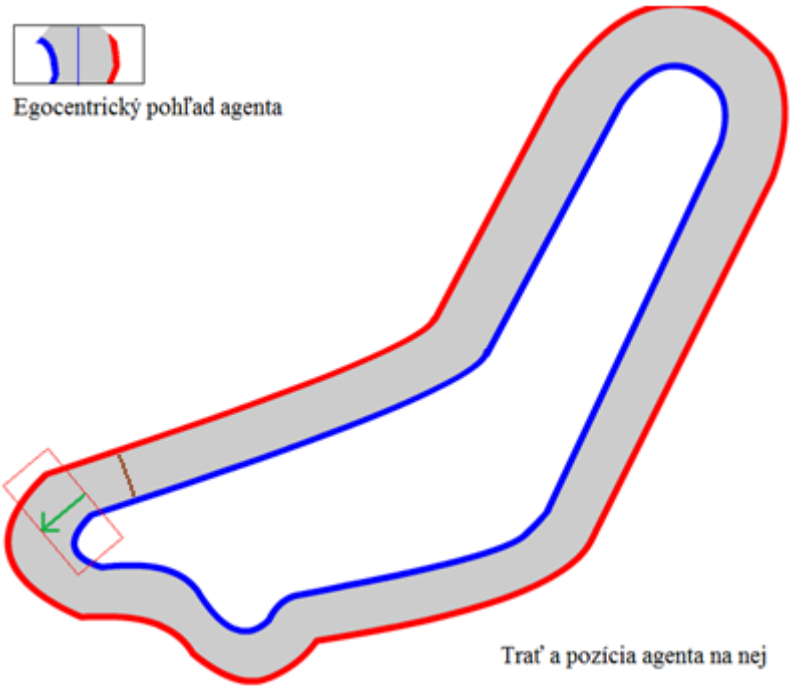
Táto práca nie je jediná, ktorá sa snaží naučiť stroj ovládať automobil. Jedným z takýchto projektov je aj ALVINN z roku 1989 (Autonomous Land Vehicle In a Neural Network), ktorý však pristupuje k úlohe inak ako v tejto práci. ALVINN využíva neurónovú sieť, podobne ako tomu bude v tejto práci, no na vstup dostáva vizuálny podnet o veľkosti 30x32 pixelov (obrázok 4), čo znamená že vstupná vrstva má rovnaký počet vstupných neurónov. Neurónová sieť používa učenie s učiteľom, čo je asi najväčší rozdiel oproti tejto práci. Projekt ALVINN však už nie je v tejto dobe ďalej vyvíjaný.

6.1 Prostredie

Skôr ako začnem so samotnou implementáciou je nutné si definovať prostredie v ktorom sa agent bude pohybovať. Zámerom tejto práce je čo najviac sa priblížiť ku skutočnej realite, kde ľudskí vodiči ovládajú svoje automobily, viď obrázok 5. Preto bude agent vnímať prostredie egocentricky a bude vidieť len časť prostredia, presnejšie časť cesty pred sebou a po bokoch, viď obrázok 6.

Obrázok 5: *Egocentrický pohľad na prostredie u vodiča automobilu.*

Keďže úlohou agenta je naučiť sa ovládať automobil na ceste, budeme potrebovať nejakou formou reprezentovať a generovať práve túto cestu. Na výber sa ponúkajú dve možnosti a to mať uzavretý okruh, čo poznáme zo závodných áut, alebo neobmedzenú neustále sa generujúcu dráhu. Pre učenie je lepšie používať uzavretý okruh, ktorý bude obsahovať zákruty rôznej náročnosti. Okraje cesty budú označené 2 farbami, červenou a modrou, aby agent mal prehľad či jeho aktuálne smerovanie nie je v protismere jazdy. Okruh bude reprezentovaný vo forme obrázku, tzv. bitmapy, takže nepôjde o okruh definovaný vektormi ale pixlami, čo má síce svoje nevýhody čo do presnosti, ale pre účel tejto práce je to plne postačujúce, už len z toho dôvodu, že ani človek nevníma prostredie ako vektory, ale skôr ako obrázok a vďaka duálnemu videniu, ľavé a pravé oko, je tento pohľad dokonca trojrozmerný.



***Obrázok 6**: Okruh a pohybujúci sa agent. Zelená šípka označuje smerovanie agenta. Červený rámček označuje egocentrický pohľad agenta. Hnedá čiara označuje štartovaciu pozíciu agenta.*

Už vieme, že existujú dva typy úloh a to sú epizodické a neepizodické, tzv. pokračujúce úlohy. Ak by som chcel túto úlohu implementovať ako neepizodickú, agent by mal mať možnosť pohybovať sa po celom prostredí neobmedzene, čo znie ako rozumný nápad, no v skutočnosti to bude viesť len k tomu, že ak agent opustí cestu, bude blúdiť v prostredí a ani fakt, že náhodou nájde ceste mu nezaručí že opätovne túto cestu neopustí. Takto by strávil mimo cesty viac času ako na nej, čo by viedlo len k degradácii učenia. Preto budem túto úlohu implementovať ako epizodickú, čo v tomto prípade znamená, že ak agent vybočí z cesty, jedna epizóda sa ukončí a začne nová, kde bude agent opätovne postavený na štart a nasmerovaný správnym smerom. Učenie sa tým podstatne urýchli.

Úlohou agenta teda bude prejsť okruh bez opustenia cesty. V optimálnom prípade, po kompletnom naučení, by mal byť agent schopný krúžiť po okruhu neobmedzene dlho.

6.2 Stav

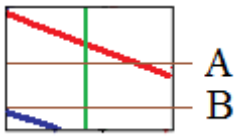
Na obrázku 6 je znázornené, čo vidí agent pri egocentrickom pohľade na okolie. Toto je jediná informácia, ktorú budem spracovávať do vhodnej formy, ktorú potom dodám agentovi, čo v podstate znamená, že všetky informácie musím vyčítať práve z tohto pohľadu, resp. obrázku. Tento prístup najviac konverguje ku skutočnosti, pretože aj človek dostáva len vizuálnu informáciu okolia a následne určité procesy v mozgu prevedú túto vizuálnu informáciu do inej formy, ktorá človeku povie kde na ceste sa nachádza a podobne. Ja však nahradím tieto mozgové procesy klasickým strojovým spracovaním obrazu z dôvodu komplexnosti samotnej úlohy spracovania obrazu.

Otázkou teraz je ako previesť vizuálnu informáciu do stavu. Dôležité je uvedomiť si, čo vlastne stav popisuje. V tomto prípade musí stav popisovať polohu agenta vzhľadom na cestu, resp. vzhľadom na stred cesty. Vyzerá to ako postačujúca informácia, no nie je. To čo doteraz nebolo v tejto práci spomenuté je dôležitosť správneho vyjadrenia stavu. Ak totiž existuje jeden stav, ktorý popisuje dve rôzne situácie, dochádza k skresleniu vizuálnej informácie. Do určitej miery je toto skreslenie prijateľné, dokonca výhodne, no ak prekročí určité hranice dochádza k nesprávnemu interpretovaniu skutočného stavu, viď obrázok 7.

***Obrázok 7**: Stav A a B je rovnaký pre rôzne situácie. Agent teda nevie či cesta smeruje doľava alebo doprava a dochádza k nesprávnemu rozhodnutiu.*

Z tohto dôvodu budem potrebovať ďalšiu informáciu. Mohla by ňou byť vzdialenosť k okraju cesty pri danom nasmerovaní, na obrázku 7 je to veľkosť zelenej čiary. Táto informácia je však tiež skresľujúca, ako vidieť na obrázku 7. Ďalšou, tentoraz správnou informáciou, by bola vzdialenosť od stredu cesty pri konci vizuálnej informácie, viď obrázok 8. Treba však podotknúť, že informácia o vzdialenosti od stredu musí byť bipolárna, čo znamená, že ak bude agent naľavo od stredu cesty, bude hodnota tejto informácie naberať negatívne hodnoty, zatiaľ čo ak bude napravo od

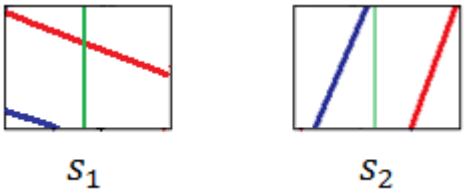
stredu, budú hodnoty kladné. Pokiaľ sa agent bude nachádzať v strede, hodnota bude rovná nule. Na základe tohto možno povedať, že ak bude stav popísaný dvoma hodnotami [0;0], potom je agent v strede cesty nasmerovaný tak, aby sa v strede cesty aj naďalej udržal. Teraz už len stačí definovať jednotku vzdialenosti. Keďže okruh je reprezentovaný ako bitmapa, bude jednotkou pixel, čiže vzdialenosť vyjadruje počet pixlov od stredu cesty.



Obrázok 8: *Egocentrický pohľad agenta na prostredie. Hnedou farbou sú vyznačené oblasti, kde sa meria vzdialenosť od stredu. Oblasť B je bližšie k agentovi ako oblasť A.*

6.3 Odmena

Ako už bolo spomenuté, veľmi dôležitým elementom RL je odmena. Odmena motivuje agenta v procese učenia. Podobne ako stav, musí byť aj odmena správne vyhodnotená. Napríklad je nezmysel, aby odmena za stav s_1 bola vyššia ako odmena za stav s_2 (obrázok 9), pretože agentovi ide o maximalizovanie odmien a takáto odmena by ho viedla k opúšťaniu cesty.



Obrázok 9: *Príklad dvoch rozdielnych stavov.*

Taktiež by nebolo vhodné, ak by dva podobné stavy, resp. symetrické, napríklad [1;0] a [-1;0] mali rozdielne odmeny. Preto treba rovnicu pre výpočet odmeny koncipovať tak, aby dávala najvyššie hodnoty pre stav [0;0] a pre všetky ostatné stavy odmeny nižšie a zároveň aby bola symetrická podľa stredu, čo sa veľmi ľahko dosiahne používaním absolútnej hodnoty pri argumentoch stavu. Keďže chcem, aby agent

zostal na ceste a najlepším riešením je byť v strede cesty, resp. byť nasmerovaný na stred cesty, treba agenta odmeňovať pozitívne aj v prípade, keď sa síce nenachádza v strede, ale je natočený smerom k nemu.

Aby som si uľahčili výpočet, odmenou bude vzdialenosť od stredu v oblasti A, $offset_A$ (obrázok 8). Keďže by však maximálna odmena mala byť dosahovaná len vtedy, ak je vzdialenosť od stredu rovná 0, treba rovnicu otočiť opačne, to znamená, že budeme od nejakého čísla, označme ho $r_{\max}$, odrátavať hodnotu $offset_A$. Zvolenie vhodnej hodnoty $r_{\max}$ je zložitejšie ako by sa mohlo zdať. Treba si uvedomiť aké extrémy dosahuje hodnota $r_{\max}$. Tie závisia od extrémov $offset_A$. Extrémy $offset_A$ však závisia od šírky pozorovaného prostredia, resp. egocentrického pohľadu, čo už je ale záležitosť samotnej implementácie, preto si toto rozhodnutie nechám na neskôr.

6.4 Metóda učenia, stratégia a počiatočné hodnoty

Ako metódu učenia si zvolíme SARSA metódu, spadajúcu do kategórie TD učenia, pretože je veľmi jednoduchá na implementáciu a má nízke pamäťové a výpočtové nároky:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Prevažnú časť práce bola odporúčaná stratégia ϵ-greedy. Takáto stratégia je však veľmi nebezpečná, pretože s pravdepodobnosťou ϵ vedie k náhodnému, prevažne chybnému, rozhodnutiu. Predstavme si situáciu, v ktorej sa agent už naučil ovládať automobil a následne ide po ceste, tesne pri jej ľavom kraji, a zrazu sa rozhodne otočiť volant o 9 stupňov doľava, čím sa dostane mimo cesty, a tým sa epizóda končí. Takýto agent nikdy nebude schopný stopercentne ovládať automobil. Preto by sa tento prvok náhody nemal používať, ak už je agent naučený.

Aby som podporil skúmanie, inicializujem počiatočné hodnoty $Q(s, a)$ na hodnotu rovnú $r_{\max}$ a to z jednoduchého dôvodu. Hodnota $Q(s, a)$ bude prevažne klesať, pretože optimálna akcia je takmer vždy len jedna, čím sa umožní skúšanie zvyšných akcií. Aby som toto klesanie ešte urýchlil, zavediem tzv. *penalizáciu*, čo nie je nič iné ako použitie negatívnej odmeny, čo znamená že horné hranica $r_{\max}$ bude menšia ako horná hranica $offset_A$. Otázkou je, prečo nestačilo zvýšiť hodnotu parametra α.

Problém je v tom, že α sa týka δ_t, čiže chybovosti TD učenia, v ktorom je zahrnutá aj $\gamma Q(s_{t+1}, a_{t+1})$. Penalizácia sa však týka len odmeny r_{t+1}, čo je výhodnejšie.

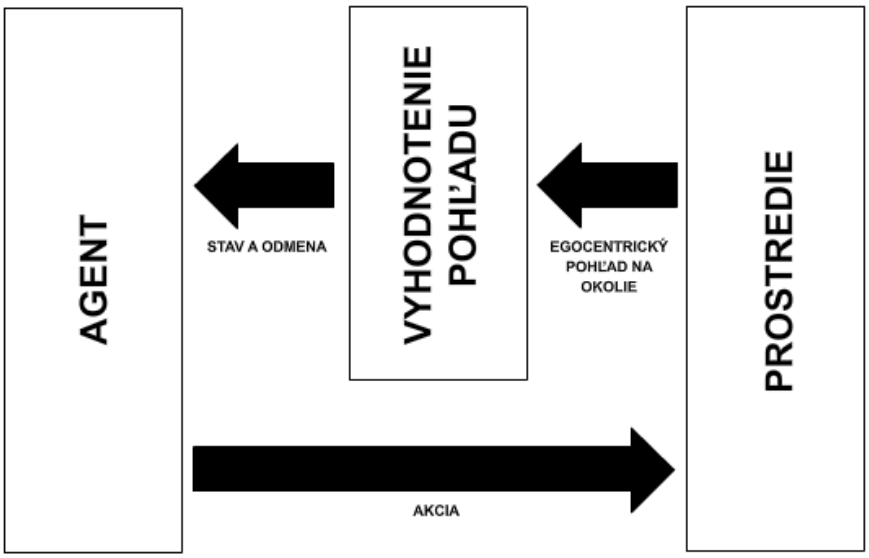
Nakoniec nastavím nasledujúce parametre na hodnoty:

$$\alpha = 0.6;\ \gamma = 0.9; \epsilon = 0.3$$

6.5 Implementácia

V tejto časti sa budem venovať modelu, metódam a rovniciam, ktoré sú potrebné na implementovanie agenta schopného naučiť sa ovládať automobil.

6.5.1 Model



Obrázok 10: *Model systému.*

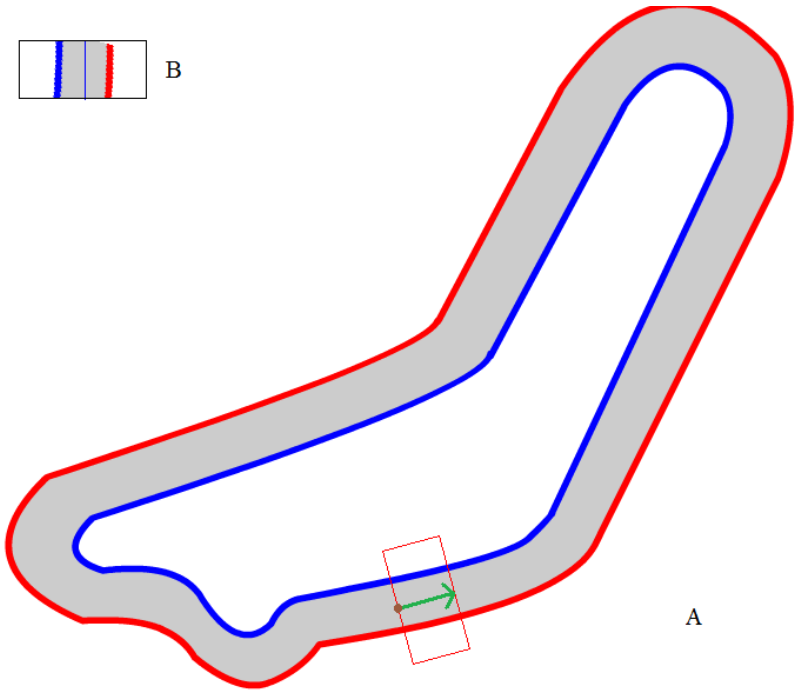
Na obrázku 10 vidno model implementovaného systému. Už vieme, že sa agent pohybuje v prostredí, s ktorým interaguje. Podobne, ako u človeka dochádza k vizuálnemu vnímaniu prostredia prostredníctvom zraku, *prostredie* v tomto modeli zasiela vizuálnu informáciu systému pre *vyhodnotenie pohľadu*. Jeho úlohou je vyhodnotiť tento pohľad, čiže zistiť vzdialenosti od stredu cesty v oblastiach *A* a *B*, a taktiež vypočítať aktuálnu odmenu. Túto informáciu, resp. stav a odmenu, zašle následne agentovi, ktorý použije stratégiu na výber vhodnej akcie, ktorú následne zašle prostrediu, ktoré ju aplikuje na seba a celý proces sa začína znova.

6.5.2 Prostredie

Prostredie pracuje nad obrázkom okruhu (obrázok 11/A), po ktorom sa má pohybovať automobil, resp. agent. Zároveň musí mať informácie, kde sa aktuálne agent nachádza (hnedá bodka), a ktorým smerom je nasmerovaný (zelená šípka). Tiež by mal vedieť, do akej diaľky a šírky agent vidí (červený rámček).

Ak prostredie obdržalo od agenta akciu, vykoná ju. K dispozícii je 19 možných akcií, z toho 9 je otočenie doľava o uhol 1 až 9 stupňov, jedna akcia je udržuj smer, a ďalších 9 akcií je otočenie doprava o uhol 1 až 9 stupňov.

Následne prostredie pošle systému pre vyhodnotenie pohľadu obrázok toho, čo vidí agent, v tomto prípade je to červený rámček (samozrejme z pohľadu agenta), viď obrázok 11/B.



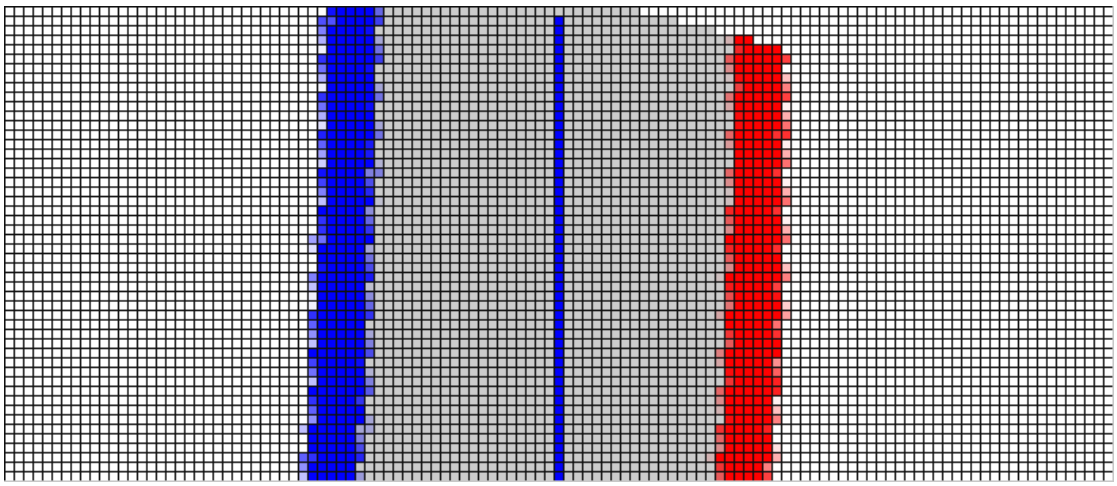
Obrázok 11: *Ukážka prostredia a egocentrický pohľad agenta naň.*

6.5.3 Vyhodnotenie pohľadu

To čo systém pre vyhodnotenie pohľadu dostane od prostredia je bitmapa určitých rozmerov *ŠxV*, v tomto prípade 100x50. Bitmapa sa skladá z dvojrozmerného poľa pixlov, kde každý pixel má svoju farbu, ako je vidieť na obrázku 14. Úlohou systému

je nájsť ľavý (modrý) a pravý (červený) okraj cesty a na základe toho určiť, ako ďaleko je agent od stredu cesty.

Princíp výpočtu je triviálny, pretože ak sa agent nachádza v strede cesty, potom vzdialenosť ľavého a pravého okraja cesty musí byť rovnaká. V prípade ak je agent viac na pravej strane cesty, čiže je napravo od stredu, bude vzdialenosť pravého okraja väčšia ako ľavého a naopak. Vzdialenosť meriame od okraja bitmapy tak, že pri hľadaní ľavého okraju (červená farba) začíname hľadať od pravej strany bitmapy a naopak pri prvom okraji. Dôvod pre takého rozhodnutie tkvie v tom, že ak by sa na obrázku nevyskytoval žiaden červený pixel v oblasti, resp. riadku, kde ho hľadáme, potom je tento mimo pohľadu, a preto sa vzdialenosť ľavého okraja nastaví na 0. Pre pravý okraj sa v prípade absencie modrých pixlov nastaví vzdialenosť na šírku pohľadu, v tomto prípade hodnota *Š*. Pretože pohľad je schopný zabrať celú cestu vrátane bielych miest naokolo nej, situácia kedy by ľavý okraj bol rovný nule a pravý rovný *Š* nastane len v prípade, ak je agent v danom riadku mimo cesty.



Obrázok 12: *Priblížený pohľad na prostredie.*

Po určení vzdialeností okrajov nasleduje výpočet posunutia voči stredu cesty:

$$offset = (pravý\_okraj - \text{šírka}) - (\text{šírka} - \text{ľ}avý\_okraj)$$

kde $pravý\_okraj$ je vzdialenosť od pravého okraja, ľ$avý_okraj$ je vzdialenosť od ľavého okraja a š$írka$ je polovičná hodnota zo šírky pohľadu *Š*, čiže horizontálny

stred pohľadu (na obrázkoch je znázornený tenkou modrou zvislou čiarou). Obor hodnôt premennej $offset$ je $\langle -100, 100 \rangle$.

Ako už bolo spomenuté, posunutie voči stredu cesty sa meria v dvoch oblastiach (obrázok 8). Oblasť *A* je vo vzdialenosti 20 pixlov od spodného okraja a oblasť *B* vo vzdialenosti 5 pixlov od spodného okraja. Keďže ani človek nevidí periférne 180°, ani agent nebude vidieť, preto je oblasť *B* posunutá na spomínanú hodnotu dopredu, smerom od agenta. Oblasť *A* je v danej vzdialenosti preto, lebo by sa medzi tieto dve oblasti mohla vtesnať ostrejšia zákruta a agent by si ju tak nevšimol. Stav je teda definovaný ako $[offset_B; offset_A]$.

Nakoniec už zostáva len vypočítať odmenu. Keďže už poznáme extrémy $offset_A$, môžeme určiť hodnotu $r_{\max}$. Nezabúdajme, že chceme zachovať penalizáciu, preto väčšina stavov musí mať negatívnu odmenu. Chceme totiž agenta silnejšie motivovať k nájdeniu optimálnej trasy, čiže stredu cesty. V časti o odmene som spomenul, že budem používať absolútnu hodnotu vzdialenosti od stredu v oblasti A, čiže $offset_A$, a tiež chcem aby len pár stavov bolo ohodnotených kladne. Dosadím teda za $r_{\max}$ nízke číslo, napríklad 10[2], čo znamená že len 21 vzdialeností, $\langle -10, 10 \rangle$, bude mať kladnú hodnotu odmeny a z toho len hodnota 0 bude mať maximálnu odmenu:

$$r = r_{\max} - |offset_A|$$

6.5.4 Agent

Ako už vieme, agent obdrží od prostredia popis stavu a odmenu za aktuálny stav. Úlohou agenta je sa na základe týchto informácií rozhodnúť pre nasledujúcu akciu. Ak je stav popísaný dvoma hodnotami, $[offset_B; offset_A]$, potom ide o dvojrozmerné pole. Ak ku každému stavu pridáme ešte 19 možných akcií, ide o trojrozmerné pole. ϵ-greedy stratégia znamená nájdenie akcie s najvyššou hodnotou

[2] Výber čísla 10 je čisto náhodný výber nízkeho čísla a je otázkou, či by výber iného čísla podporil alebo potlačil rýchlosť procesu učenia.

v treťom rozmere tohto poľa, pretože prvé dva rozmery máme už dané, sú popísané aktuálnym stavom. Funkcia pre vybranie akcie by mohla vyzerať nasledovne:

```
function get_next_action(s: state): action;
var
  a, max: Integer;
begin
  max := Low(Q[s.offset_b, s.offset_a]);
  for a := max + 1 to High(Q[s.offset_ b, s.offset_a]) do
    if (Q[s.offset_b, s.offset_a, a] > Q[s.offset_ b, s.offset_a, max]) then max := a;

  if (random(100) <= EPSILON) then Result := Random(High(Q[s.offset_ b, s.offset_a]))
  else Result := max;
end;
```

Teraz keď poznáme aktuálny stav s_t, odmenu za neho r_t, akciu a_t, ktorá sa má vykonať, a výpočet SARSA:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

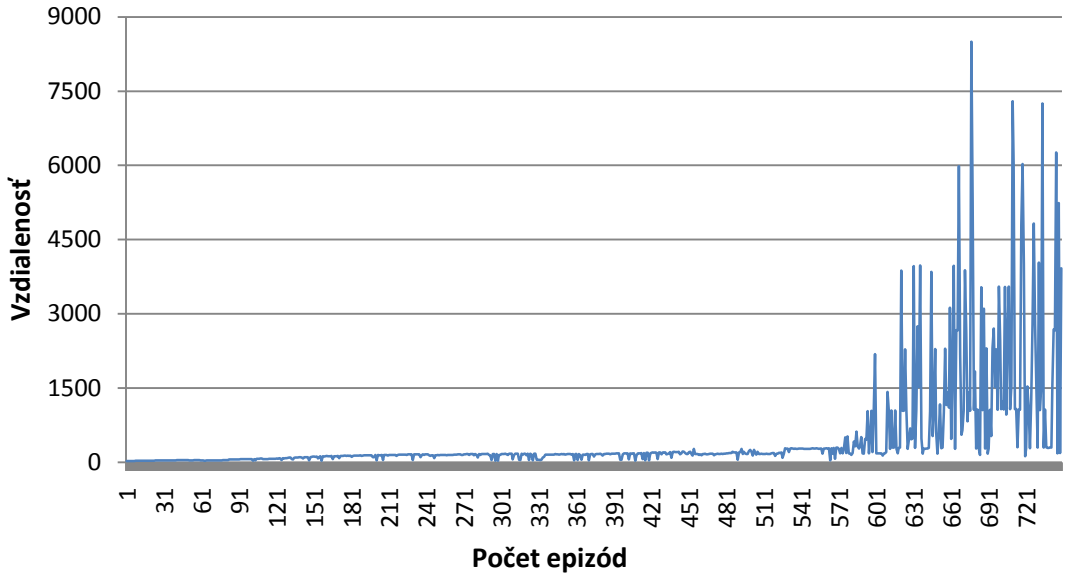
Zostáva nám už len zistiť hodnotu nasledujúceho stavu s_{t+1}, odmenu za tento stav r_{t+1} a akciu a_{t+1} ktorú vykonáme následne, na základe čoho sa potom opýtame na predpokladanú hodnotu akcie $Q(s_{t+1}, a_{t+1})$. Z tohto dôvodu, musí implementácia umožňovať aplikovať akciu na stav bez vplyvu okolité prostredie.

Ak už máme všetky parametre známe, môžeme upraviť tabuľku predpokladaných hodnôt $Q(s, a)$. Teraz agent pošle prostrediu informáciu, ktorú akciu si vybral. Prostredie sa následne zmení a celý proces začína odznova.

6.6 Vyhodnotenie

Na grafe 3 je vidno, ako dlho (resp. koľko epizód) trvalo agentovi, kým bol schopný prejsť kompletnú trať aspoň raz. Jedno kolo na trati má dĺžku približne 1500 pixlov. Mohlo by sa zdať, že agent napriek tomu, že odjazdil celé kolá, nebol schopný pokračovať ďalej, pretože zišiel z cesty. Netreba však zabúdať, že sme hodnotu ϵ nastavili na 0.3, čo je veľmi vysoká hodnota, no cieľom bolo podporiť skúmanie nových stavov, pretože agent konverguje ku optimálnej trajektórii, čiže k strede cesty,

a tak sa s hraničnými stavmi po väčšine ani raz nestretne, čo však znamená, že agent nie je *úplne naučený*. Pri tejto implementácií teda agent nebol očividne schopný vysporiadať sa s každou novou situáciou, resp. stavom.



***Graf 3**: Dosiahnutá vzdialenosť v pixloch.*

6.7 Vylepšenie stavu

Z implementácie vidíme, že veľkosť stavového priestoru je 201x201=40401. Aby sme mohli prehlásiť, že agent sa úplne naučil šoférovať, musí tento prejsť všetkých 40401 stavov minimálne toľkokrát, koľko je akcií pre každý z týchto stavov. To je enormne veľké číslo. Urobíme preto vylepšenie. Na začiatku tejto kapitoly som spomínal skreslenie. A práve toto skreslenie teraz využijem. Aký je rozdiel medzi stavmi:

[-5;0], [-4;0], [-3;0], [-2;0], [-1;0]

[5;0], [4;0], [3;0], [2;0], [1;0]

[-10;0], [-9;0], [-8;0], [-7;0], [-6;0]

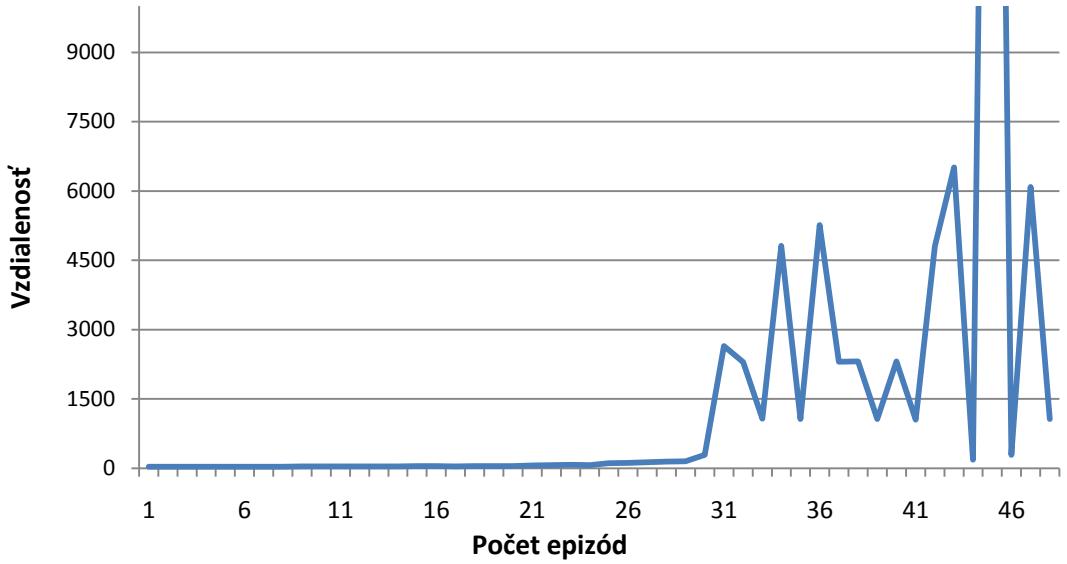
[10;0], [9;0], [8;0], [7;0], [6;0]

.....?

Optimálne akcie v týchto stavoch sa líšia minimálne. A preto, argumenty x, y stavu $[x; y]$ upravíme nasledovne: $x = \lceil x/5 \rceil; y = \lceil y/5 \rceil$, kde 5 je hodnota *faktoru skreslenia*. Týmto sa veľkosť stavového priestoru zníži na 41x41=1681 a doba

potrebná na naučenie agenta sa tiež výrazne zmenší, ako je vidieť na grafe 4. Toto skreslenie je veľká výhoda, pretože agent veľmi rýchlo preskúma väčšinu stavov a dokáže sa tak lepšie vysporiadať s náhodným výberom akcie. Keďže argument vstupujúci do výpočtu odmeny sa tiež zmenšil 5-násobne, je nutné upraviť tento výpočet aby sme zachovali pôvodnú penalizáciu nasledovne:

$$r = r_{\max} - |5 * \ offset_A| = 10 - |5 * \ offset_A|$$



Graf 4: *Dosiahnutá vzdialenosť v pixloch použitím faktoru skreslenia.*

6.8 Rýchlosť

Až doteraz sa agent pohyboval konštantnou rýchlosťou a jediné rozhodnutie, pre ktoré sa rozhodoval bolo natočenie seba samého, resp. automobilu, o určitý uhol do vybraného smeru. Skutočný vodič však musí okrem toho ovládať aj rýchlosť automobilu, v ktorom sa nachádza, preto pridám do rozhodovacej logiky agenta parameter rýchlosti.

Presnejšie, popis stavu rozšírime o ďalší parameter $speed$, teda o rýchlosť. Dôvod je zrejmý: je rozdiel mať vychýlenie oproti optimálnej trajektórii o 15° a ísť rýchlosťou 1km za hodinu a mať rovnaké vychýlenie a ísť rýchlosťou 100 km za hodinu. Aj keď obe situácie budú riešené potočením volantu do rovnakého smeru, uhol natočenia bude iný. To znamená, že ak by som nepridal parameter rýchlosti do popisu stavu,

potom by jeden stav popisoval 2 rozdielne situácie. Zároveň si však treba uvedomiť, že stavový priestor sa takto opätovne zväčšil, no agent by mal byť stále schopný naučiť sa ovládať automobil a zároveň maximálne využiť potenciál rýchlosti.

Počet možných rýchlostí nastavím na 3. Pri implementácii bude rýchlosť reprezentovať násobok čísla 3, to znamená, že pri rýchlosti jedna prejde agent za jednotku času 3 pixle, pri rýchlosti dva prejde 6 pixlov, pri rýchlosti tri 9 pixlov, čo sú hodnoty aj opticky rozpoznateľné. Zároveň je však nutné zvýšiť maximálny uhol otočenia agenta z 9° na 11°, čo znamená 23 (11+1+11) možných akcií v každom stave, pretože okruh, po ktorom sa agent pohybuje je natoľko náročný, že už pri rýchlosti jedna a maximálnom možnom natočení 9 by agent po väčšine nebol schopný zvládnuť sériu zákrut na spodku okruhu.

Okrem pridania parametra rýchlosti, musím pozmeniť aj stratégiu agenta. Agent totiž musí vedieť pridať rýchlosť, udržať aktuálnu alebo spomaliť. Pridaním rýchlosti budeme rozumieť zvýšenie aktuálnej rýchlosti o 1, udržaním rýchlosti znamená nemeniť ju a spomalením je zníženie rýchlosti o 1 s tým, že hodnota rýchlosti je z rozsahu $\langle 1, 3 \rangle$. Preto agent musí prehľadávať pri stratégii ϵ-greedy stavový priestor od $[offset_B; offset_A; speed - 1]$ po $[offset_B; offset_A; speed + 1]$.

Nakoniec ešte musím upraviť výpočet odmeny. Ten musí motivovať agenta k prejdeniu úseku vo väčšej rýchlosti aj za cenu horšieho stavu, čo znamená aj za cenu väčšej vzdialenosti od stredu v oblasti *A*. Po sérií experimentov som došiel k tomuto výpočtu:

$$r = 2 * speed + 5 - |5 * \ offset_A|$$

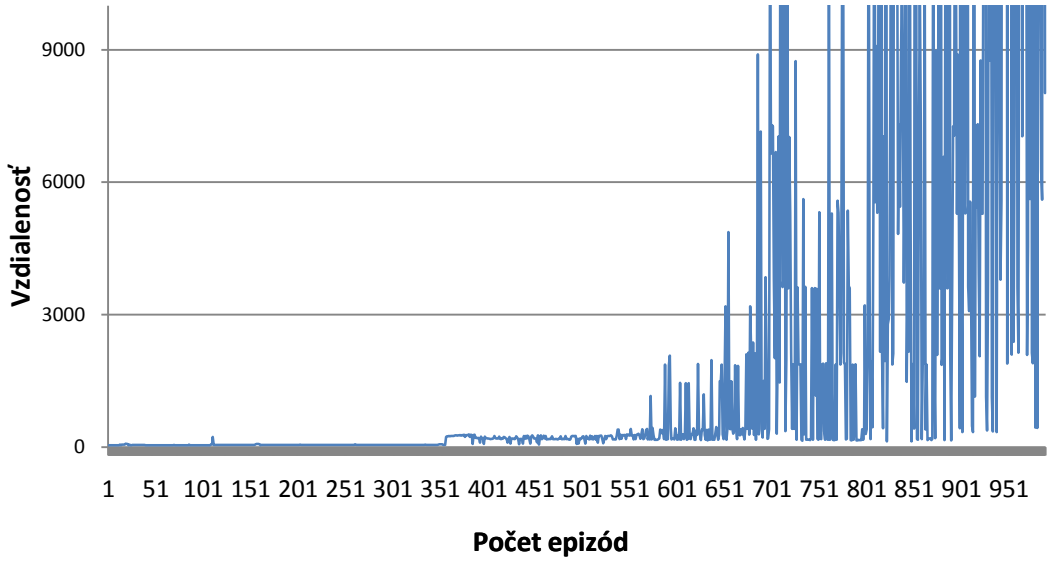
Keďže sa zmenila aj maximálna možná odmena, treba upraviť aj počiatočné hodnoty $Q(s, a)$, kde s popisuje vzdialenosti v oblasti A a B a rýchlosť $speed$, na:

$$Q(s, a) = speed * 2 + 5$$

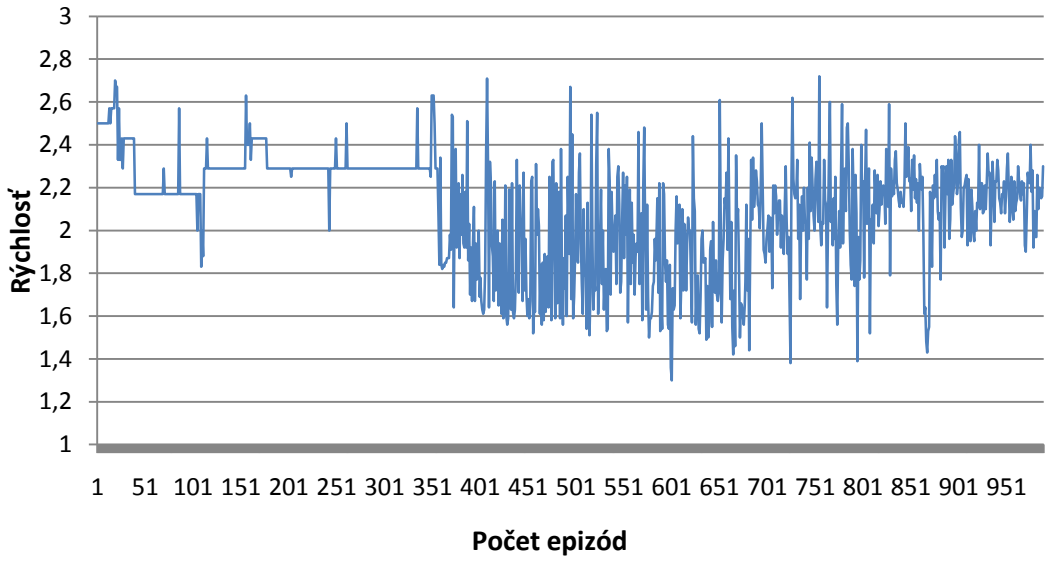
6.8.1 Vyhodnotenie

Na grafe 5 vidíme, že schopnosť agenta urobiť aspoň jedno kolo (približne 1500 pixlov) prišla až po 550 epizódach. Podobne ako v predchádzajúcich modeloch

agenta, aj tento nebol ani po 963 epizódach a veľkom počte odjazdených kôl schopný dokonale ovládať automobil a v prípade, ak ho náhodný výber dostal do nového stavu opustil trať.



Graf 5: *Dosiahnutá vzdialenosť v pixloch pri modeli s rýchlosťou.*



Graf 6: *Priemerná rýchlosť počas epizódy pri modeli s rýchlosťou.*

V tomto modeli agenta máme k dispozícii aj graf jeho priemernej rýchlosti za dobu, behom ktorej zotrval na ceste (graf 6). Je vidieť, že v prvotných epizódach bola rýchlosť agenta vyššia, aj keď jeho schopnosť prejsť dlhší úsek bola menšia, čo ale

nie je nič neočakávané, pretože štartovacia pozícia agenta bola tesne pred sériou náročných zákrut (obrázok 6).

6.9 Neurónová sieť ako náhrada faktoru skreslenia

Až doteraz sme používali faktor skreslenia na zrýchlenie učenia agenta. Prílišné skreslenie je však nebezpečné, pretože zanedbáva detaily, čo môže viesť v skutočnom svete k havárii. Preto sa namiesto skreslenia využívajú neurónové siete (NS), ktoré majú schopnosť interpolácie. Nevýhodou NS je ich doba učenia. Pri zložitých systémoch, resp. systémoch s veľkým stavovým priestorom, je schopnosť NS naučiť sa do takej miery, aby chybovosť klesla pod 1% skôr v teoretickej rovine ako v praxi, pretože výpočtová zložitosť procesu by presiahla bežne dostupné kapacity. Ďalšou nevýhodou je výskyt lokálnych miním pri učení. NS je tak nutná dočasne degradovať svoje učenie aby sa z lokálneho minima dostala. Taktiež je NS náchylná na zabúdanie, pretože nové dáta na vstupe interferujú so starými.

Využitie neurónovej siete s RL na ovládanie automobilu, resp. niečoho, čo by automobil pripomínalo, nie je v praxi až tak bežné, práve kvôli veľkej časovej náročnosti na učenie. V mojom prípade je táto nevýhoda tiež prítomná, pretože mám stavový priestor v mojom agentovi bez využitia faktoru skreslenia o veľkosti 40401 pre jednu akciu a máme 23 akcií pre každú rýchlosť, pričom počet rýchlostí sme si zvolili 3, teda dokopy by musela neurónová sieť vedieť interpolovať 2787669 hodnôt odmien pre ľubovoľnú akciu v ľubovoľnom stave, čo je enormné číslo.

Na zjednodušenie učenia (a elimináciu interferencie rôznych akcií) môžeme využiť rozdelenie neurónovej siete na menšie časti tak, aby každá akcia mala svoju vlastnú neurónovú sieť, nezávislú od ostatných (Wiering a Hasselt, 2007). Každej neurónovej sieti sa na vstup načíta aktuálny stav a následne prebehne výpočet výstupnej hodnoty, teda predpokladanej nasledujúcej odmeny, na každej sieti. Tieto výstupné hodnoty sa následne porovnajú a vyberie sa tá akcia, ktorej výstupná hodnota je najvyššia, čo je identické so stratégiou greedy, resp. ϵ-greedy, ak budem využívať aj náhodný výber akcie. Po obdržaní skutočnej odmeny sa upravujú váhy na tej neurónovej sieti, ktorá prislúcha vybranej, teda vykonanej, akcii.

Aby sme ukázali, že aj neurónová sieť je schopná ovládať automobil využitím RL, obetujem presnosť za rýchlosť učenia, pretože zmením spôsob, akým sa popisuje stav.

Až doteraz som na popis stavu používal tri parametre, $offset_B$;$offset_A$;$speed$. Keďže chcem zmenšiť stavový priestor, je nutné znížiť alebo úplne eliminovať jeden z parametrov, v tomto prípade jeden z parametrov vzdialenosti. Logicky vychádza potlačenie parametra vzdialenosti v oblasti B, pretože ho potrebujem najmenej. Treba si však byť vedomý toho, že presnosť akcie týmto výrazne utrpí, pretože by napríklad mohla nastať situácia (obrázok 13), kde je hodnota $offset_A$ rovnaká a dokonca smer otočenia bude rovnaký, ale uhol, o ktorý by sa mal agent natočiť je rozdielny. Preto možno tento model považovať aj za experiment, ktorý dokáže, či agent je schopný zostať na ceste aj napriek výraznej nepresnosti.

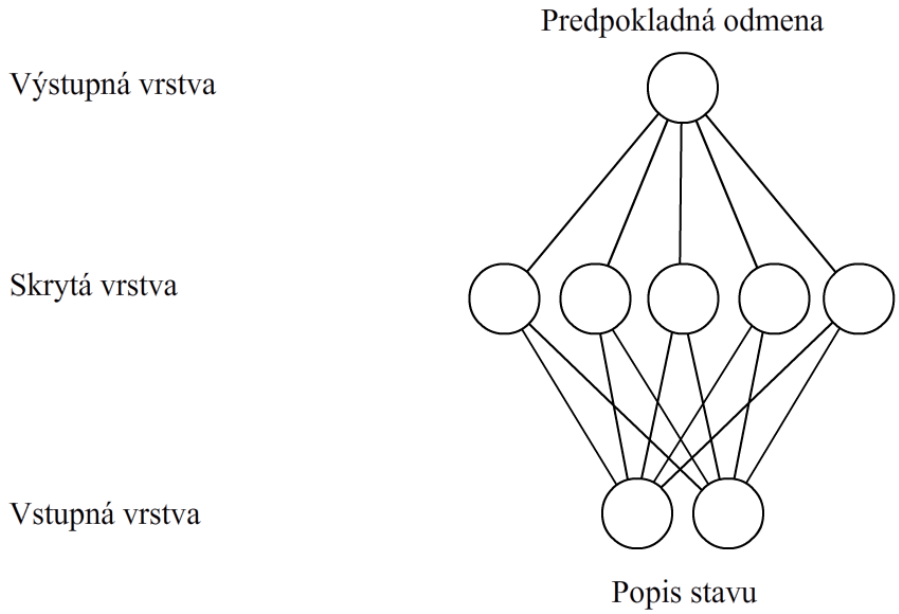
Obrázok 13: *Dve rozdielne situácie s rovnakou hodnotou parametra* $offset_A$.

Stavový priestor sa teraz zmenšil na veľkosť 101 pre každú rýchlosť. Zatiaľ čo pôvodný spôsob popisovania stavu bol jedinečný pre každú situáciu, nový popis stavu môže predstavovať viacero veľmi podobných situácií, čo je však daň za požadovanú rýchlosť učenia. Nakoniec, keďže sa pracuje s NS, je nutné vstupné hodnoty preškálovať do rozsahu $\langle -1; 1\rangle$ pre rýchlosť a $\langle -5; 5\rangle$ pre $offset_A$, nastaviť rýchlosť učenia $\alpha = 0.0001; \gamma = 0.9; \epsilon = 0.3$. Tieto hodnoty sú intuitívne vybrané, vzhľadom na priebeh rôznych simulácií.

Na porovnanie rýchlosti učenia a iných faktorov urobím dve simulácie, kde prvá bude NS len s jednou, konštantnou, rýchlosťou agenta a druhá bude umožňovať dve rýchlosti. Predpokladom je, že NS s konštantnou rýchlosťou sa bude učiť rýchlejšie.

6.9.1 Štruktúra neurónovej siete a procesy

Testovaním rôznych štruktúr NS sa ukázala ako najvhodnejšia štruktúra 2:5:1 (obrázok 14).



Obrázok 14: *Štruktúra neurónovej siete.*

Celkovo má NS 15 váh a ich počiatočné hodnoty boli nastavené náhodne v rozsahu $\langle -0{,}5;\, 0{,}5 \rangle$ a prah neurónov na hodnotu -1. Keďže výpočet odmeny obsahuje funkciu absolútnej hodnoty a úlohou NS je konvergovať k tomuto výpočtu, bolo nutné použiť skrytú vrstvu. Menší počet neurónov než 5 na skrytej vrstve sa ukázal ako nepostačujúci na dobrú interpoláciu, a práve táto štruktúra sa dokázala extrémne rýchlo učiť.

Výpočet výstupnej hodnoty y^O NS prebieha nasledovne:

1. Vloženie preškálovaných hodnôt popisujúcich stav na vstupnú vrstvu I,
2. Výpočet výstupnej hodnoty y pre h-ty neurón v skrytej vrstve H podľa vzorca: $y_h^H = \tanh(b_h^H + \sum_{i=1}^{3} I_i\, W_{ih}^H)$, kde b_h^H je prah h-teho neurónu na skrytej vrstve a W_{ih}^H je váha medzi i-tym vstupným neurónom a h-tym neurónom na skrytej vrstve,
3. Výpočet výstupnej hodnoty y na výstupnej vrstve O NS podľa vzorca: $y^O = \frac{1}{5}(b^O + \sum_{h=1}^{10} y_h^H\, W_{ho}^O)$, kde menovateľ zlomku je hodnota rovná počtu neurónov na skrytej vrstve a to z toho dôvodu, že výstup hyperbolického tangensu, čiže aktivačnej funkcie neurónov na skrytej vrstve, je maximálne 1 a ak máme týchto neurónov v skrytej vrstve 5, potom maximálny súčet je rovný 5.

Lenže výstupná hodnota z lineárnej aktivačnej funkcie výstupného neurónu by mala byť hodnota maximálne 1 (maximálna odmena je 0.5 a predpovedáme krok dopredu, takže 2 * 0.5), čo by mali zabezpečiť váhy smerujúce do tohto neurónu. Problémom však je hodnota váh, ktorá musí byť v tomto prípade 5-krát menšia, ako keby sme delili priamo vstupnú hodnotu do aktivačnej funkcie číslom 5. A takáto nízka hodnota váh si vyžaduje veľmi nízku hodnotu rýchlosti učenia, α, čo by však spomaľovalo úpravu váh medzi vstupnými a skrytými neurónmi. V praxi sa teda ukázalo výhodnejšie používať v lineárnej aktivačnej funkcii spomínaný zlomok.

NS, ktorej výpočet výstupnej hodnoty prebieha takto sa nazýva aj dopredná neurónová sieť [feedforward neural network]. Následne sa vyberie tá neurónová sieť, ktorej výstupná hodnota je najvyššia a vykoná sa akcia, ktorá tejto NS prislúcha.

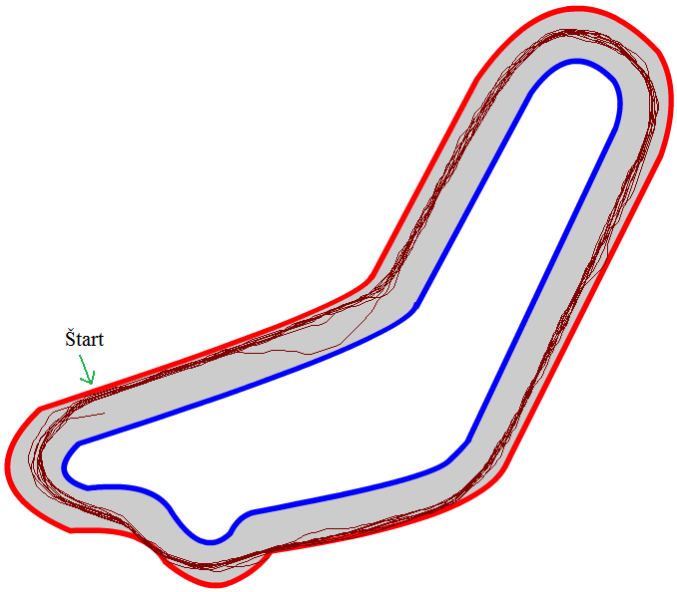
Učenie neurónovej siete prebieha pomocou metódy backpropagation nasledovne:

1. Upraví sa prah výstupného neurónu:
   $b^O = b^O + \alpha\delta_t$, kde $\delta_t = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$ a δ_t je z rozsahu $\langle -1; 1 \rangle$,
2. Upravia sa váhy medzi skrytou a výstupnou vrstvou: $W_{ho}^O = W_{ho}^O + \alpha\delta_t y_h^H$,
3. Upraví sa prah na neurónoch v skrytej vrstve:
   $b_h^H = b_h^H + \alpha e_h^H$, kde $e_h^H = (1 - (y_h^H)^2) W_{ho}^O(-\delta_t)$,
4. Upravia sa váhy medzi vstupnou a skrytou vrstvou: $W_{ih}^H = W_{ih}^H - \alpha e_h^H y_h^H$.

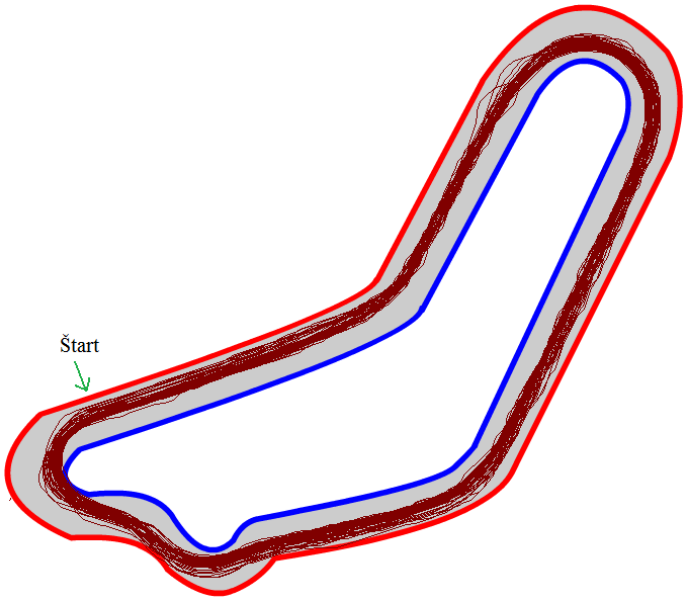
6.9.2 Vyhodnotenie simulácie s konštantnou rýchlosťou

Učenie behom simulácie prebiehalo nad očakávania, agentova schopnosť udržať sa na ceste bola prítomná veľmi skoro po inicializácii NS. Ako je z grafu 7 vidieť, NS sa učila veľmi rýchlo, aj preto je celkovo počet epizód veľmi malý, čo je potešujúce, ale hlavne prekvapivé.

Na obrázku 15 je znázornená trajektória dráhy agenta na trati počas šiestej (poslednej) epizódy, v tomto prípade mala epizóda dĺžku vyše 11 kôl. Štartovacia pozícia bola zámerne vybraná pred sériou zákrut, pretože sa ukázalo, že agent sa rýchlejšie učí, ak ho čakajú náročné podmienky hneď na začiatku.

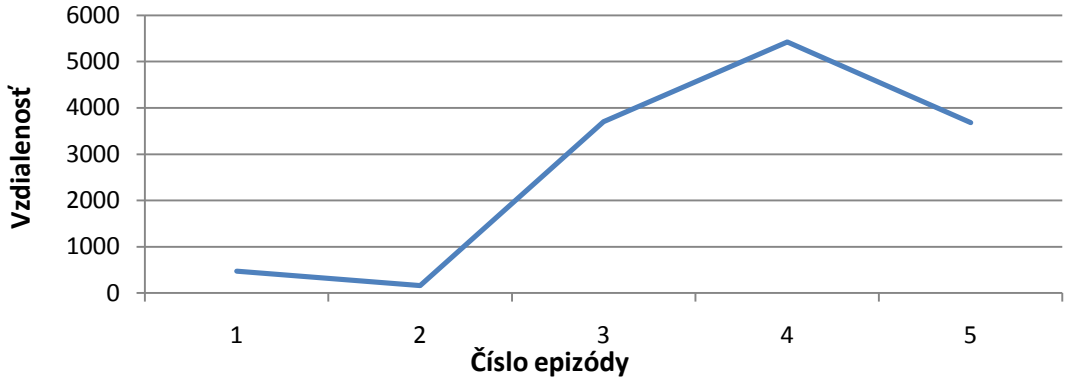


Obrázok 15: *Trajektória dráhy agenta po 11 kolách pre model jednej neurónovej siete pre každú akciu a s konštantnou rýchlosťou.*

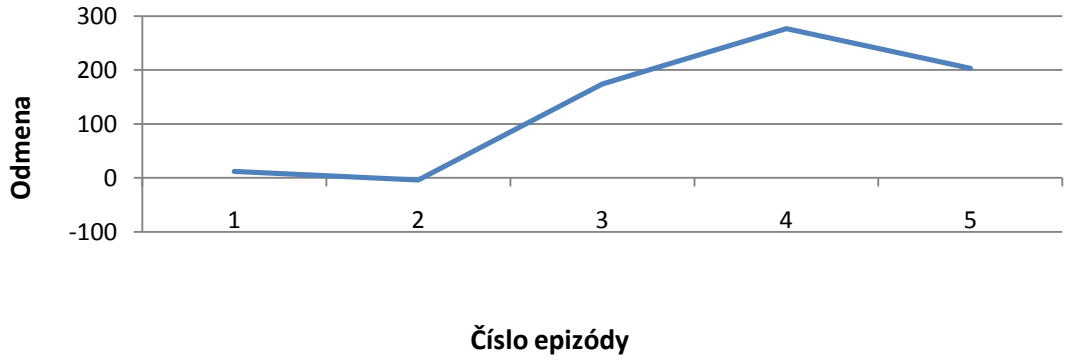


Obrázok 16: *Trajektória dráhy agenta počas celej doby simulácie pre model jednej neurónovej siete pre každú akciu a s konštantnou rýchlosťou.*

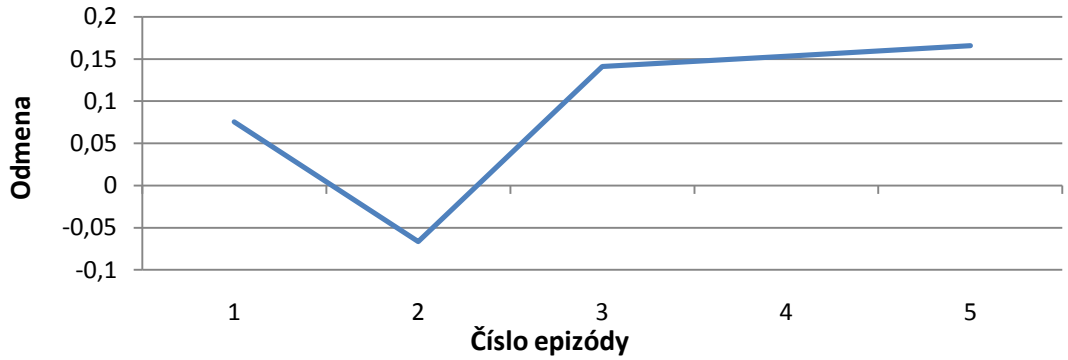
Simuláciu som zastavil po šiestej epizóde z dôvodu vysokého počtu agentom odjazdených kôl po trati, viac ako 11 kôl, čo znamená že agentove schopnosti dosiahli nami želateľnú úroveň.



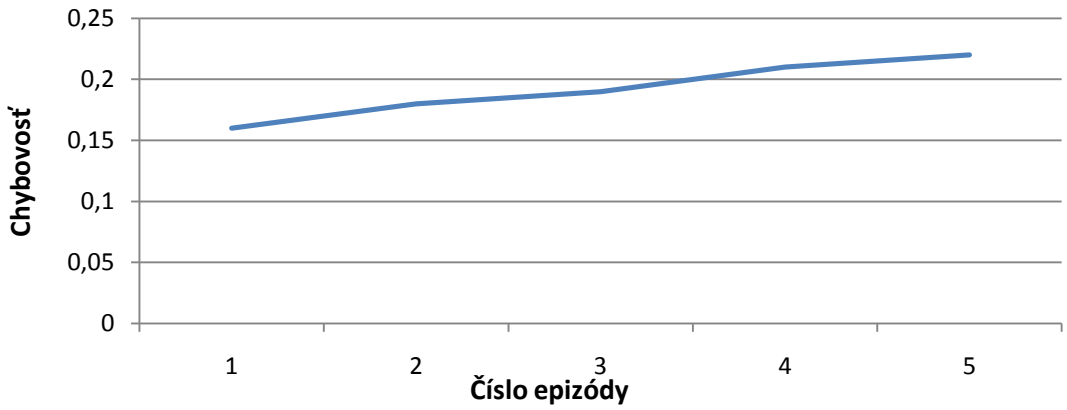
Graf 7: *Dosiahnutá vzdialenosť v pixloch pre model jednej neurónovej siete pre každú akciu a s konštantnou rýchlosťou.*



Graf 8: *Dosiahnutá odmena počas epizódy pre model jednej neurónovej siete pre každú akciu a s konštantnou rýchlosťou.*



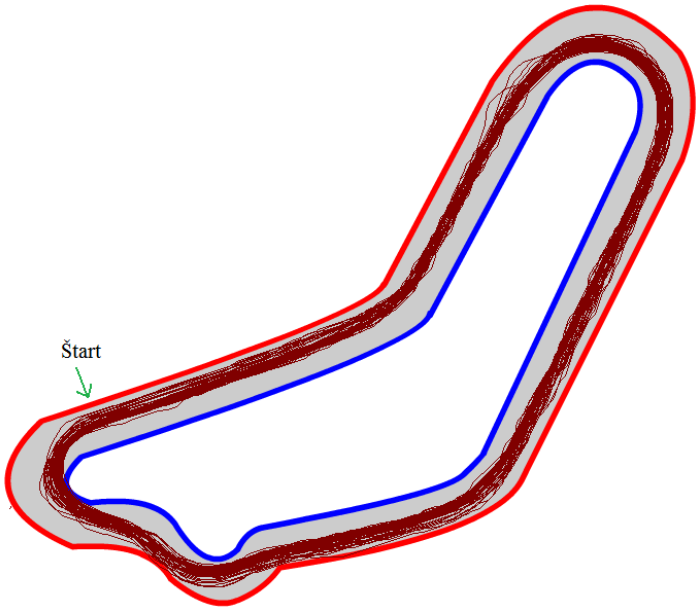
Graf 9: *Priemerná odmena za akciu pre model jednej neurónovej siete pre každú akciu a s konštantnou rýchlosťou.*



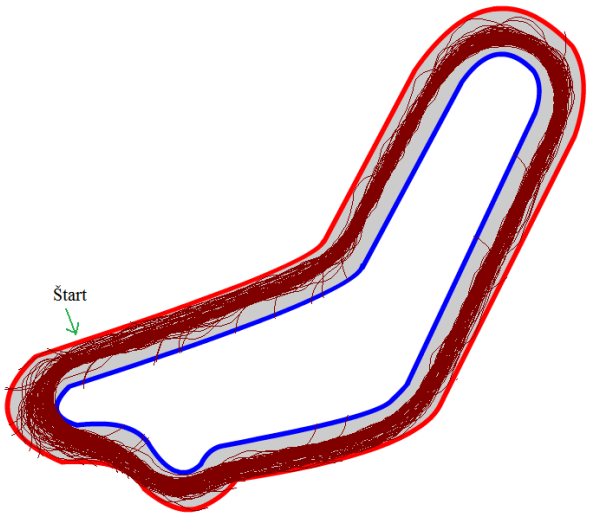
Graf 10: *Priemerná chybovosť počas epizódy pre model jednej neurónovej siete pre každú akciu a s konštantnou rýchlosťou.*

6.9.3 Vyhodnotenie simulácie s dvoma rýchlosťami

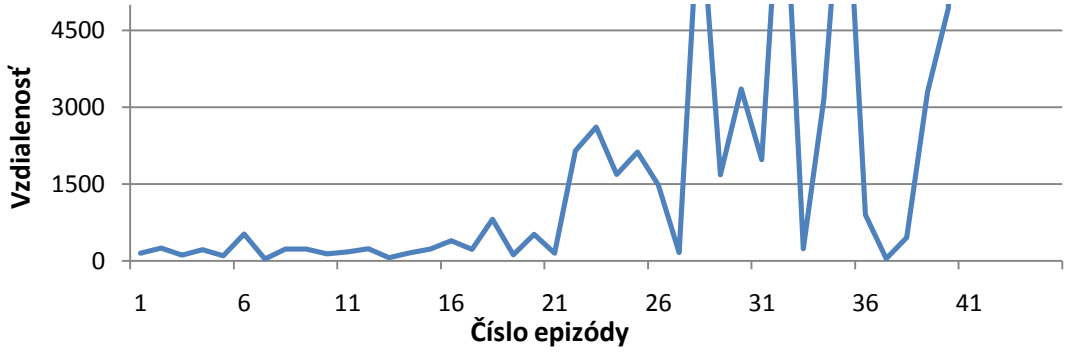
V tejto simulácii prebiehalo učenie agenta tiež rýchlo (aj keď nie až tak, ako v predošlej simulácii), ako vidieť na grafe 11. Nemožno však povedať, že by dosiahnutá rýchlosť bola oslnivá (graf 12). Pravdepodobne by stačilo len nechať simuláciu bežať dlhšie. Paradoxom zostáva fakt, že napriek schopnosti agenta zostať na ceste dlho, chybovosť stúpa. Len pripomeniem, že hodnoty v grafoch sú ohraničené z hora z dôvodu, aby bolo lepšie vidieť nízke hodnoty.



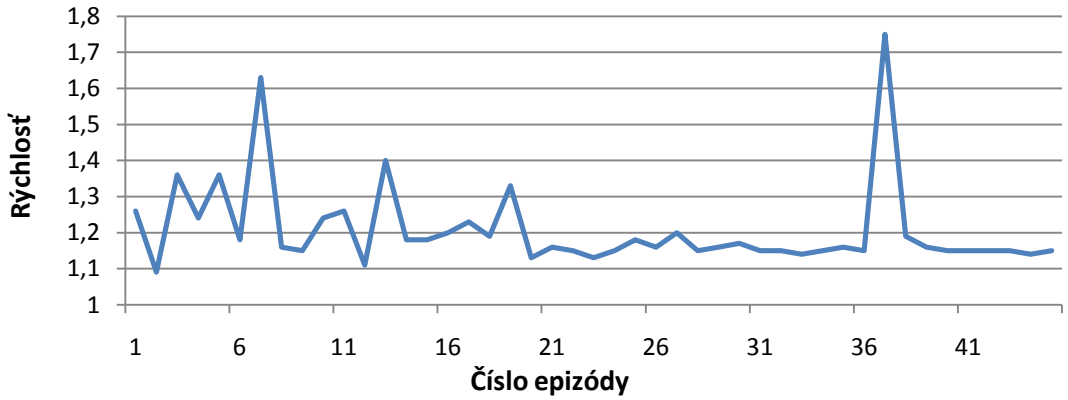
Obrázok 17: *Trajektória dráhy agenta počas poslednej epizódy pre model jednej neurónovej siete pre každú akciu s dvoma rýchlosťami.*



Obrázok 18: *Trajektória dráhy agenta počas celej doby simulácie pre model jednej neurónovej siete pre každú akciu s dvoma rýchlosťami.*

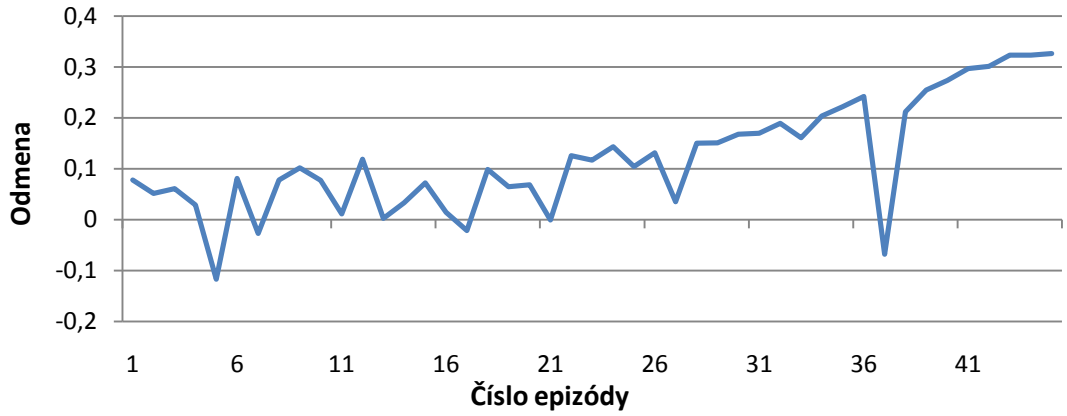


Graf 11: *Dosiahnutá vzdialenosť v pixloch pre model jednej neurónovej siete pre každú akciu s dvoma rýchlosťami.*

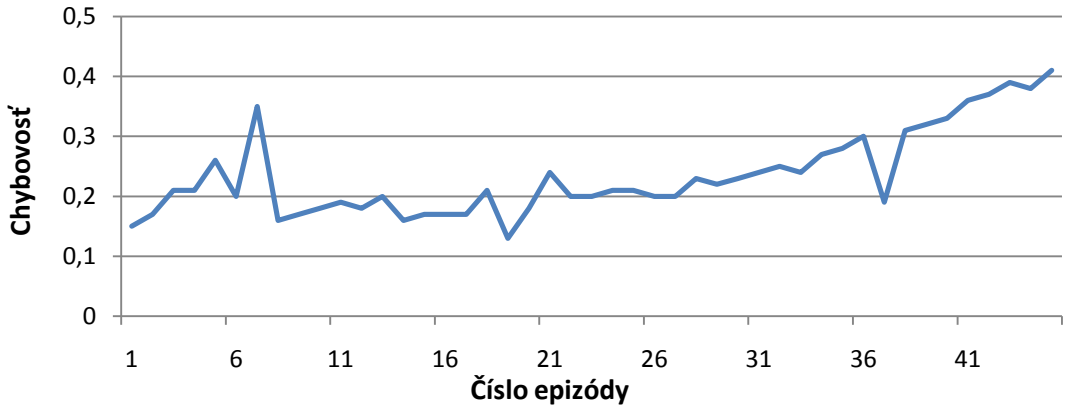


Graf 12: *Priemerná rýchlosť počas epizódy pre model jednej neurónovej siete pre každú akciu s dvoma rýchlosťami.*

Graf 13: *Dosiahnutá odmena počas epizódy pre model jednej neurónovej siete pre každú akciu s dvoma rýchlosťami.*



Graf 14: *Priemerná odmena za akciu pre model jednej neurónovej siete pre každú akciu s dvoma rýchlosťami.*

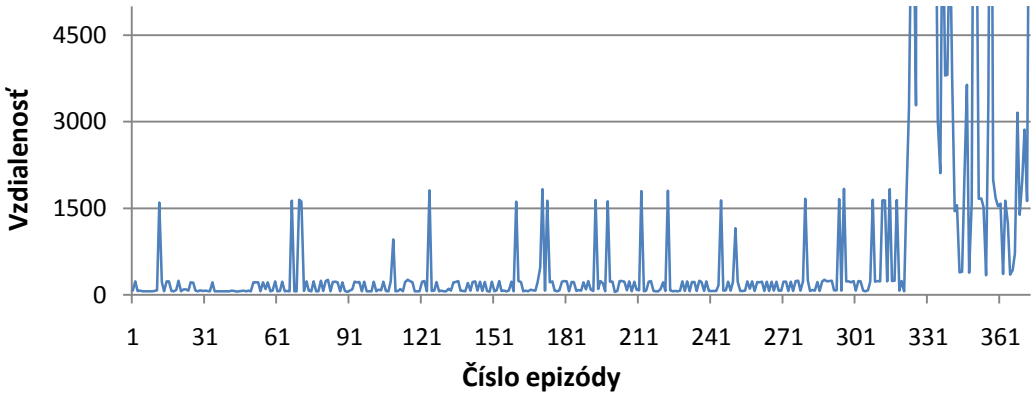


Graf 15: *Priemerná chybovosť počas epizódy pre model jednej neurónovej siete pre každú akciu s dvoma rýchlosťami.*

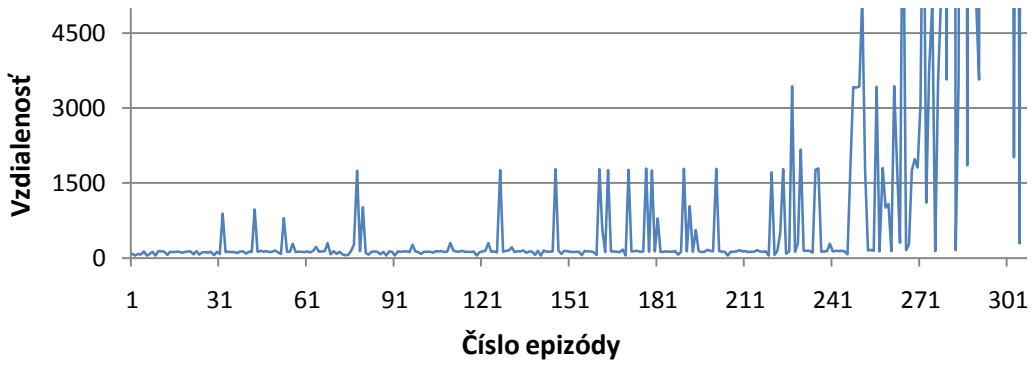
6.9.4 Agent s jednou neurónovou sieťou

Aj keď predchádzajúci model agenta využíval neurónové siete, nemožno povedať, že je to dokonalé riešenie pre interpoláciu. Dôvodom je využívanie nezávislých NS pre jednotlivé akcie, čo nie je efektívne. Preto v tejto časti skúsim spojiť všetky NS do jednej tak, aby časť neurónovej siete bola spoločná a časť bola samostatná pre každú akciu osobitne.

Na to stačí len upraviť pôvodný model siete pre jednu akciu. Namiesto jedného výstupného neurónu, bude mať táto NS výstupov toľko, koľko je možných akcií. Takto budú vstupné a skryté neuróny spoločné, zatiaľ čo výstupné neuróny budú samostatné. Pre jednu rýchlosť stačí použiť len 1 vstupný neurón, takže stav bude popísaný len parametrom $offset_A$.



Graf 16: *Dosiahnutá vzdialenosť v pixloch pre model spoločnej neurónovej siete pre všetky akcie s konštantnou rýchlosťou.*

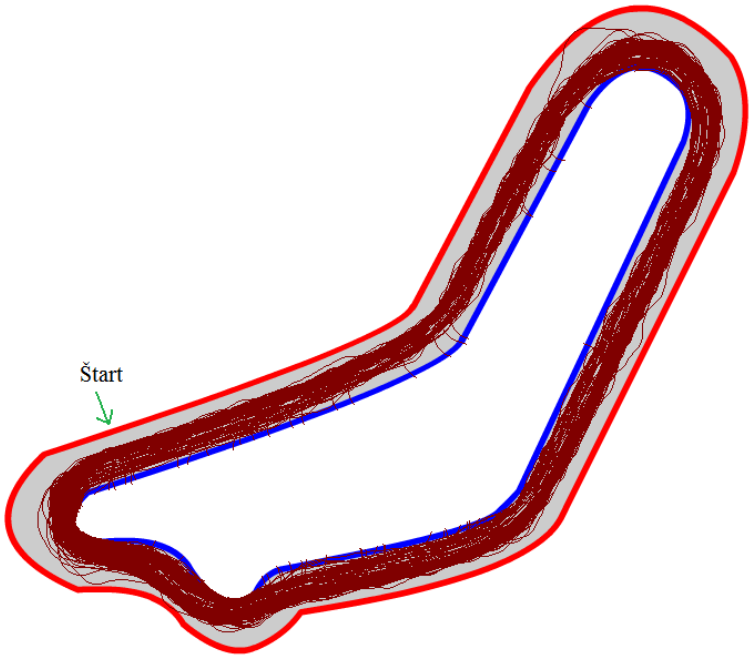


Graf 17: *Dosiahnutá vzdialenosť v pixloch pre model spoločnej neurónovej siete pre všetky akcie s dvoma rýchlosťami.*

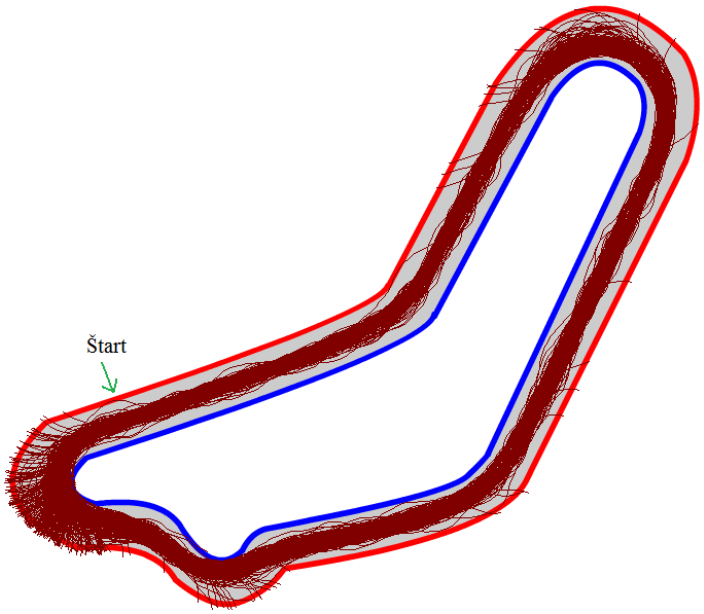
Testovaním som došli k záveru, že postačuje použiť len 2 neuróny na skrytej vrstve pre konštantnú rýchlosť, resp. 8 neurónov pre dve rýchlosti, a ponechať rýchlosť učenia na $\alpha = 0.0001$, aby bol agent schopný naučiť sa dostatočne ovládať automobil, resp. aby schopnosť NS interpolovať odhady odmien pre jednotlivé akcie bola dostatočná na zotrvanie na ceste. Ide teda o najvýraznejšiu a poslednú optimalizáciu v tejto práci.

Z grafu 16 vidno, že agent nadobudol dostatočné schopnosti na udržanie automobilu na ceste pri konštantnej rýchlosti pomerne rýchlo. Podobne je tomu aj pri dvoch rýchlostiach (graf 17), kde som simuláciu ukončil po 36 odjazdených kolách v poslednej epizóde.

Trajektórie dráh v týchto dvoch simuláciách sa dosť výrazne líšia v sérii zákrut hneď po štarte. Zatiaľ čo agent s konštantnou rýchlosťou (obrázok 19), bol schopný zvládnuť prvú zákrutu od začiatku, agent s dvoma rýchlosťami (obrázok 20) túto schopnosť nemal. Tiež si môžeme všimnúť, že trajektória agenta s konštantnou rýchlosťou je viac orientovaná k ľavej strane cesty, zatiaľ čo u druhého agenta je zrejmá orientácia na stred.

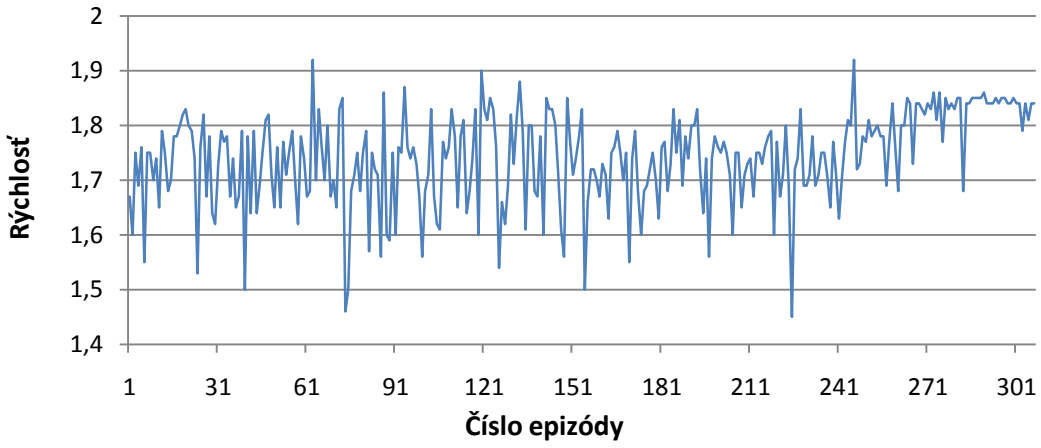


Obrázok 19: *Trajektória dráhy agenta počas celej doby simulácie pre model spoločnej neurónovej siete pre všetky akcie s konštantnou rýchlosťou*

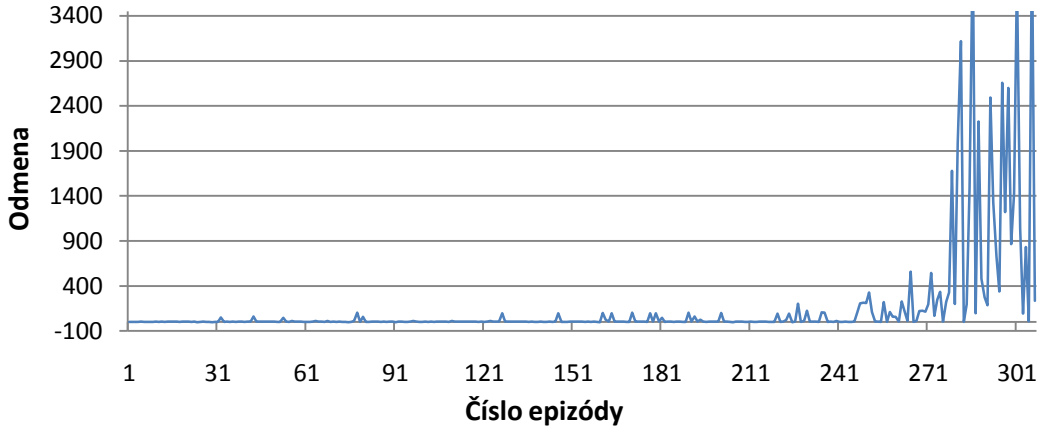


Obrázok 20: *Trajektória dráhy agenta počas celej doby simulácie pre model spoločnej neurónovej siete pre všetky akcie s dvoma rýchlosťami.*

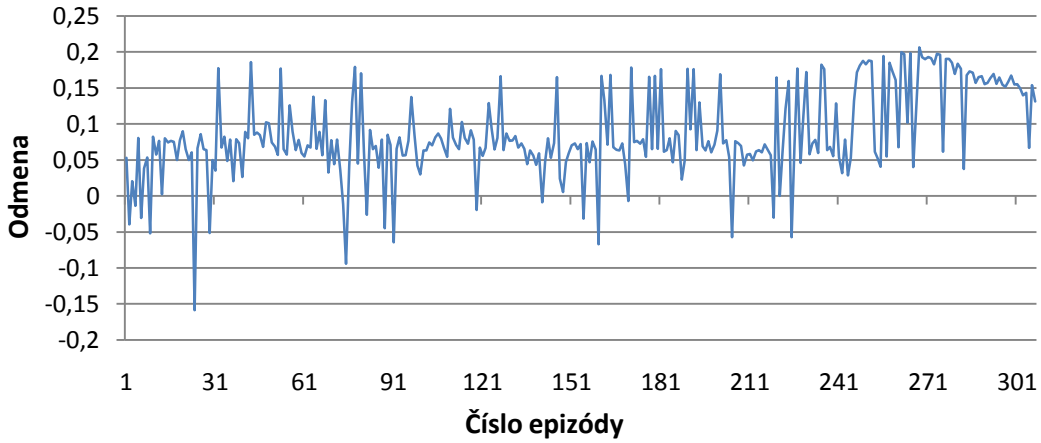
Keďže sú pre nás dôležitejšie dve rýchlosti ako jedna, ukážeme si už len grafy simulácie pre dve rýchlosti (grafy 18-21). Dôležitejšie ako odmena za celú epizódu (graf 19), kde vyššie hodnoty sú logické vzhľadom na dlhší priebeh trvania epizódy, je graf priemernej odmeny za akciu počas epizódy (graf 20) a graf chybovosti (graf 21).



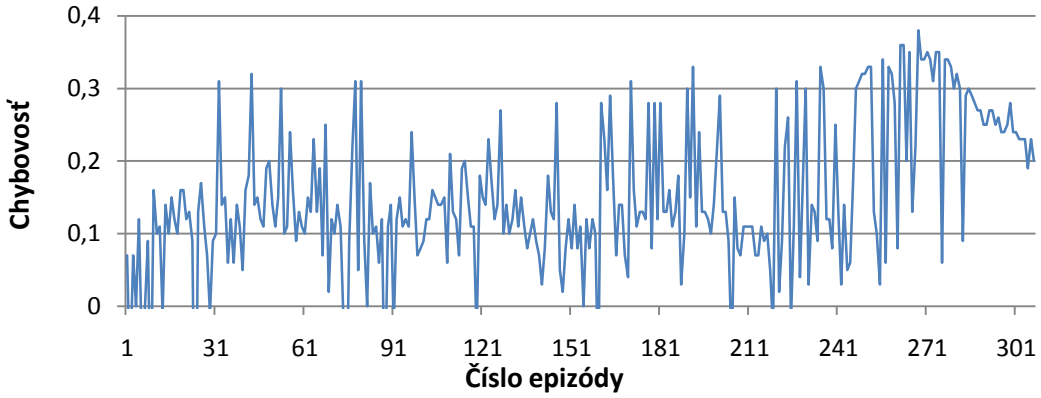
Graf 18: *Priemerná rýchlosť počas epizódy pre model spoločnej neurónovej siete pre všetky akcie s dvoma rýchlosťami.*



Graf 19: *Dosiahnutá odmena počas epizódy pre model spoločnej neurónovej siete pre všetky akcie s dvoma rýchlosťami.*



Graf 20: *Priemerná odmena za akciu pre model spoločnej neurónovej siete pre všetky akcie s dvoma rýchlosťami.*



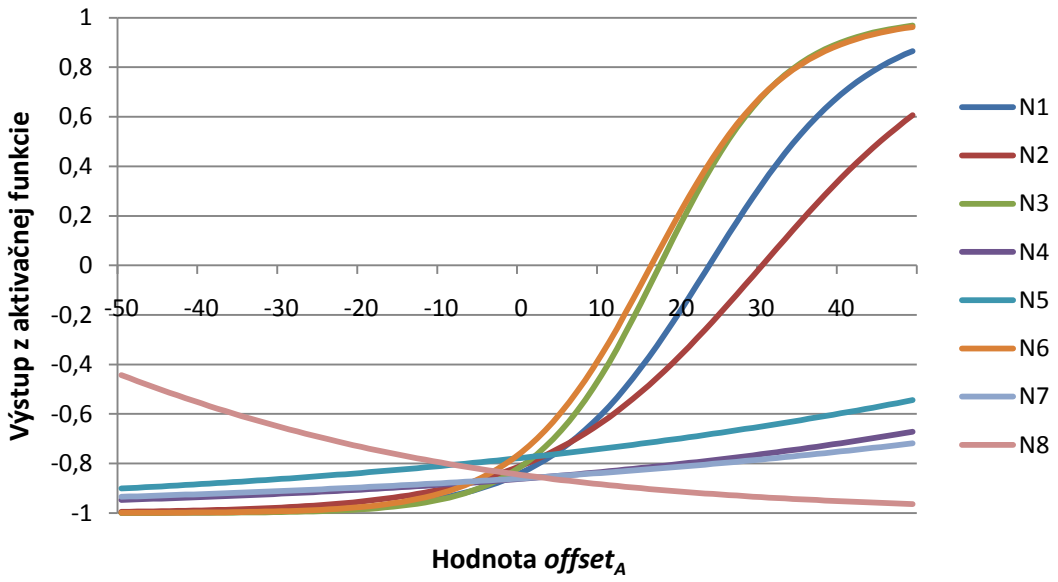
Graf 21: *Priemerná chybovosť počas epizódy pre model spoločnej neurónovej siete pre všetky akcie s dvoma rýchlosťami.*

Zdá sa, akoby tieto dva grafy spoločne súviseli, pretože keď na konci simulácie začala klesať odmena, klesala aj chybovosť. V skutočnosti však tieto dva grafy nemajú žiadnu priamu závislosť, a fakt, že trendová čiara je podobná, je len čisto náhodný. Chybovosť totiž vyjadruje rozdiel odhadovanej hodnoty vybranej akcie a jej skutočnej hodnoty, zatiaľ čo graf odmeny vyjadruje skutočné hodnoty obdržané za vybranú akciu. To, že klesá chybovosť je dôkazom, že sa NS učí správne, nehovorí to však o tom, že agent vyberá optimálne akcie. Ten vyberá len tie akcie, ktoré majú najvyššiu odhadovanú hodnotu. Ak teda optimálna akcia má nižšiu odhadovanú akciu ako akcia, ktorú agent vybral, na grafoch sa to nijako neodrazí.

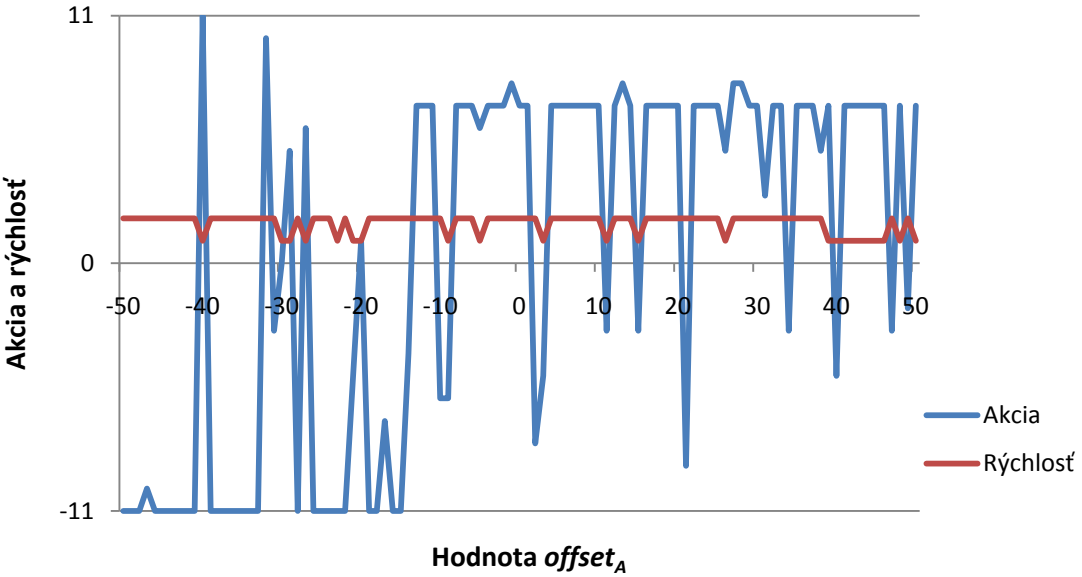
Aby sme však mohli povedať, či je agent dostatočne naučený, treba si pozrieť graf závislosti výberu akcie od stavového signálu (grafy 23 a 25). Na prehlásenie, že agent vyberá vhodné[3] (nie nutne optimálne) akcie, by museli byť všetky akcie pre stavový signál v rozsahu $\langle -50, 0)$ záporné a pre rozsah $(0, 50\rangle$ kladné, pretože záporné hodnoty akcie znamenajú potočenie doľava, keďže záporné hodnoty stavového signálu znamenajú vychýlenie doprava a naopak. Z grafu 23 však vidieť, že NS nevyberá zakaždým vhodné akcie, čo značí že aj po 307 epizódach a veľa odjazdených kolách hrozí, že agent vyletí z trate. Dôvod, prečo je agent schopný zostať na trati je v tom, že dokáže korigovať chybný výber akcie v nasledujúcom kroku, tj. väčšina akcií je vhodných.

V grafoch 22 a 24 vidieť ako sa menia hodnoty neurónov v skrytej vrstve so zmenou stavového signálu, teda so zmenou hodnoty $offset_A$ pri jednotlivých rýchlostiach. Všetky čiary sa pretínajú buď tesne pred oblasťou $offset_A$=0 alebo tesne za ňou. Predpokladom je, že v tomto bode by sa mala hodnota akcie rovnať 0, čiže žiadne otočenie agenta, čo však ani v jednom z prípadov nie je pravda a možno to tiež považovať za dôsledok nedokonalého naučenia. Taktiež z týchto grafov vidno, že aktivita na väčšine skrytých neurónov (okrem posledného neurónu) narastá priamo úmerne s nárastom hodnoty $offset_A$. To znamená, že tieto výstup z týchto neurónov je najvyšší, ak je nutné zatočiť doprava a naopak, je najnižší ak, agent zatáča doľava.

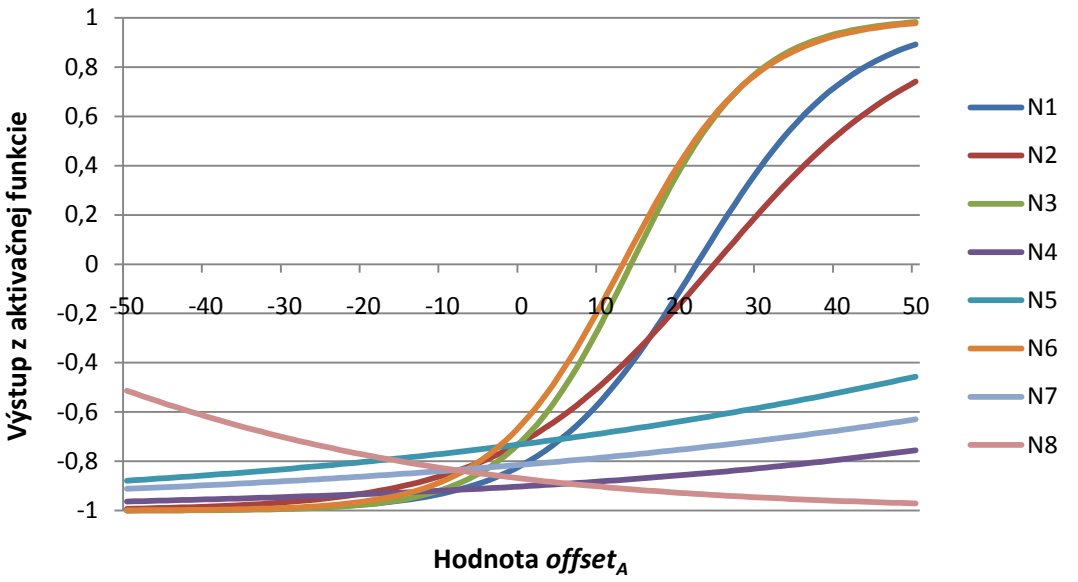
[3] Vhodná akcia je taká akcia, ktorá otočí agenta správnym smerom, ale nie nutne o správny uhol.



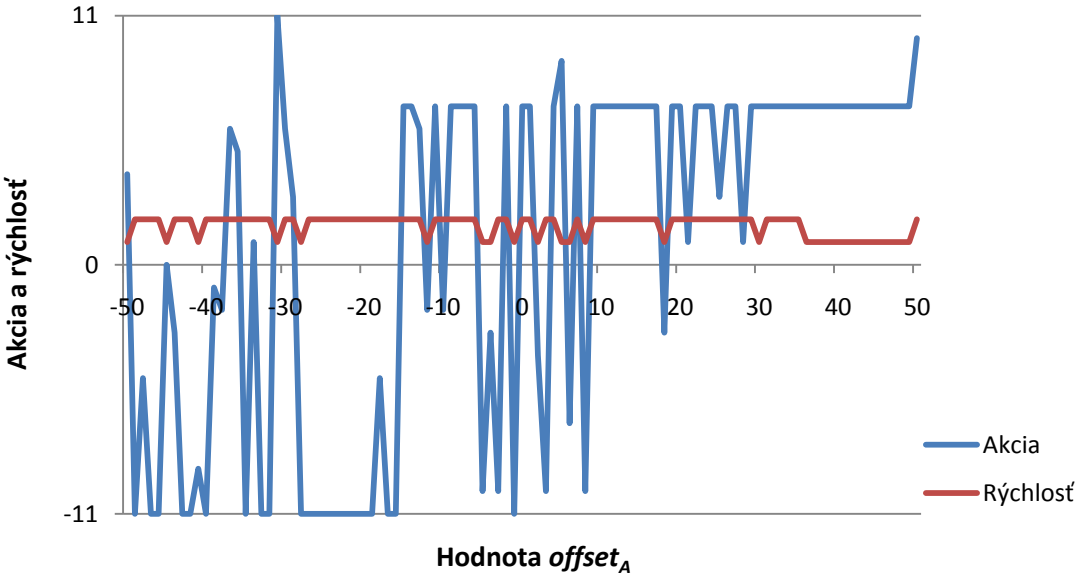
Graf 22: *Závislosť výstupnej hodnoty skrytých neurónov (N1-N8) od stavového signálu pre rýchlosť 1 v modeli spoločnej neurónovej siete pre všetky akcie s dvoma rýchlosťami.*



Graf 23*: Závislosť výberu akcie od stavového signálu pre rýchlosť 1 v modeli spoločnej neurónovej siete pre všetky akcie s dvoma rýchlosťami. Rýchlosť nadobúda len dve hodnoty a to 1 alebo 2.*



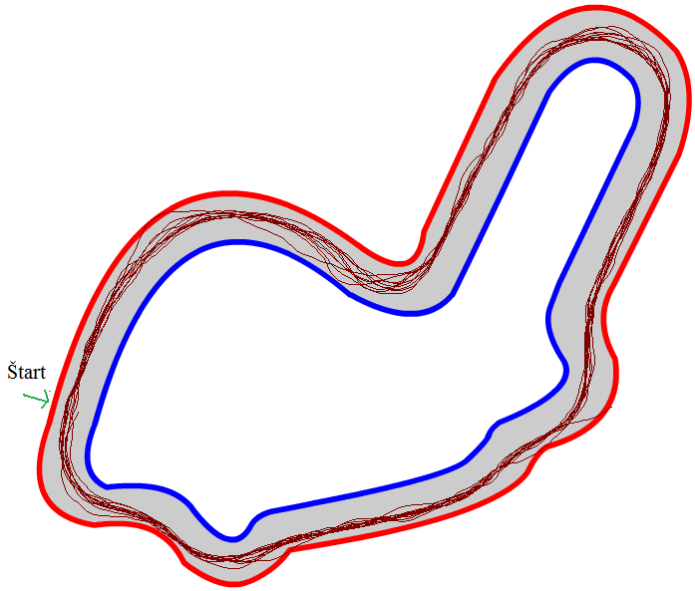
Graf 24: *Závislosť výstupnej hodnoty skrytých neurónov (N1-N8) od stavového signálu pre rýchlosť 2 v modeli spoločnej neurónovej siete pre všetky akcie s dvoma rýchlosťami.*



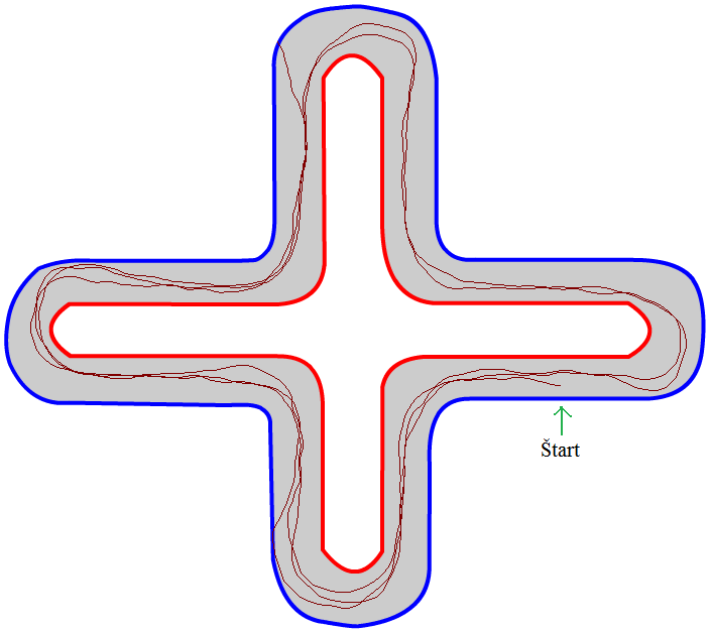
Graf 25: *Závislosť výberu akcie od stavového signálu pre rýchlosť 2 v modeli spoločnej neurónovej siete pre všetky akcie s dvoma rýchlosťami. Rýchlosť nadobúda len dve hodnoty a to 1 alebo 2.*

Keďže tvrdím, že tento posledný model je zatiaľ najlepší zo všetkých v tejto práci navrhnutých, budem ho testovať aj na iných okruhoch (obrázky 21 a 22) s tým, že

zmeriam, koľko pixlov agent prešiel, kým zišiel z trate. Agenta najskôr naučím na doteraz používanej trati a potom ju zmením. Zároveň zakážem agentovi učiť sa, ako aj zrušíme možnosť náhodného výberu akcie v stratégií ϵ-greedy.



Obrázok 21: *Trajektória dráhy na druhom type okruhu. Agent prešiel vyše 10 kôl, kým sa dostal mimo trate.*



Obrázok 22: *Trajektória dráhy na treťom type okruhu. Agent prešiel vyše 2.5 kola, kým sa dostal mimo trate.*

Napriek dlhému testovaniu, agent nakoniec zišiel z trate, čo znamená, že sieť nebola dostatočne naučená, resp. to môže aj znamenať, že jej schopnosť učiť sa a správne interpolovať nie je dostatočná. Príčinou môže byť nedostatočný počet neurónov v skrytej vrstve, výskyt lokálneho minima, alebo aj fakt, že trať na ktorej sa agent učil nie je ideálnym[4] okruhom na učenie. V tom prípade by sme mali vymeniť trénovací okruh. Keďže agent urobil na okruhu 3 (obrázok 22) len 2.5 kola, je jasné, že tento okruh je náročnejší ako ten prvý. Taktiež obsahuje ľavotočivé aj pravotočivé zákruty, vrátane roviniek, čím vyhovuje definícií ideálneho okruhu. Som však už na konci tejto práce, a preto nechávam tento problém otvorený pre ďalšie skúmanie.

[4] Ideálny v zmysle obsahujúci všetky typy zákrut a rovín, s ktorými by sa mohol agent stretnúť v budúcnosti.

7 Záver

Cieľ tejto práce, vytvoriť agenta schopného ovládať automobil vo virtuálnom svete, som prevažne splnil. Postupoval sme inkrementálne, čiže najskôr som vysvetlil problematiku, v tomto prípade princíp RL, následne vytvoril prvý model agenta, a potom som neustále vylepšoval jeho schopnosti, rýchlosť učenia, ako aj architektúru.

Určite však nemožno povedať, že posledná verzia je úplne dokonalá. Stále existuje priestor na vylepšovanie a na testovanie. Spomeňme si na projekt ALVINN, kde neurónová sieť vnímala prostredie a pomocou učenia s učiteľom sa učila ovládať skutočný automobil. Ja som tiež využil v posledných dvoch verziách neurónovú sieť, no vstupom nebol obraz ale tri parametre popisujúce stav. Možno predpokladať, že by moja neurónová sieť s RL bola schopná naučiť sa ovládať automobil aj pri obrazovom vstupe, čo je vlastne kombinácia môjho prístupu a prístupu v projekte ALVINN. Takáto architektúra by sa dala považovať ako najvhodnejšia pre tento typ úlohy.

Nielen architektúra ale aj hodnoty parametrov, napríklad rýchlosť učenia, parameter kvantifikujúci mieru zohľadnenia budúcich odmien a iné sú citlivé faktory ovplyvňujúce celkový výkon. Preto je treba ich podrobnejšie preskúmať a nájsť pre ne vhodné hodnoty, čo by mohlo byť cieľom práce iného diplomanta. V tejto práci som ich totiž volil intuitívne. Netreba tiež zabudnúť, že som využíval na učenie metódu SARSA(0), ktorá patrí medzi najjednoduchšie metódy, no napriek tomu model pracoval relatívne správne. S lepšími metódami by sa mali dostaviť aj lepšie výsledky, aj keď miera konvergencie ku skutočnému biologickému modelu bude naďalej otázna. Otázne je aj, či je nutné k nemu konvergovať.

Osobne zastávame názor, že rýchlejšie vyvinieme inteligentný systém, keď sa nebudeme snažiť kopírovať prírodu, ale budeme sa snažiť vytvoriť svoju vlastnú verziu, ktorá samozrejme, môže byť inšpirovaná biologickým modelom.

V celej práci som si nepoložil otázku, prečo využívať neurónovú sieť a RL, keď sa dá táto úloha vyriešiť aj priamym naprogramovaním. Samozrejme, ak máme malé množstvo vstupných dát, ktoré už môžu byť určitým spôsobom predspracované, nie je problém ručne vytvoriť algoritmus, riadiaci správanie agenta.

Problém nastane v prípade, ak nepoznáme všetky vzťahy medzi vstupnými dátami, alebo sú tieto vzťahy natoľko komplikované, že je takmer nemožné či neefektívne ich vyjadriť programovým kódom. Vtedy sa využíva práve metóda učenia. A ako som už napísal, v prípade ak potrebujeme aj interpoláciu hodnôt, vhodným riešením je využitie neurónovej siete.

Nakoniec nám už len zostáva vyjadriť názor, že náš posledný model agenta sa dá aplikovať aj na agentov v reálnom svete, teda by mal byť schopný ovládať automobil na ceste. Rozdiel bude len v tom, že úloha nebude epizodická a agent by sa už nemal učiť, aby nedošlo k degradácií jeho znalostí, resp. schopností, a z tohto dôvodu je nutné trénovať agenta najskôr virtuálne omnoho dlhší čas ako sme to vykonávali v tejto práci.

Zoznam bibliografických odkazov

Cummings, J. L. (1999). Understanding Parkinson Disease, *JAMA*, 281(4), p. 376–378.

Doya, K. (2007). Reinforcement learning: Computational theory and biological mechanisms. *HFSP Journal*, vol. 1, p. 30-40.

Horvitz, J. C. (2000). Mesolimbocortical and nigrostriatal dopamine responses to salient non-reward events. *Neuroscience*, vol. 96, p. 651–656.

Horvitz, J., Stewart, T., Jacobs, B. L. (1997). Burst activity of ventral tegmental dopamine neurons is elicited by sensory stimuli in the awake cat. *Brain Research*, vol. 759(2), p. 251–258.

Kakade, S., Dayan, P. (2002). *Dopamine generalization and bonuses*. *Neural Networks*, vol. 15, p. 549–559.

Kawagoe, R., Takikawa, Y., Hikosaka, O. (1998). Expectation of reward modulates cognitive signals in the basal ganglia, *Nat. Neurosci*. 1, p. 411–416.

Ljungberg, T., Apicella, P., Schultz, W. (1992). Responses of monkey dopamine neurons during learning of behavioral reactions. *Journal of Neurophysiology*, vol. 67, p. 145–163.

Montague, P. R., Dayan, P., Sejnowski, T. J. (1996). A framework for mesencephalic dopamine systems based on predictive Hebbian learning. *Journal of Neuroscience*, vol. 16, p. 1936–1947.

Murray, B., Shizgal, P. (1996). Behavioral measures of conduction velocity and refractory period for rewardrelevant axons in the anterior LH and VTA. *Physiology and Behavior*, vol. 59, p. 643–652.

Murray, B., Shizgal, P. (1996). Physiological measures of conduction velocity and refractory period for putative reward-relevant MFB axons arising in the rostral MFB. *Physiology and Behavior*, vol. 59, p. 427–437.

Pomerleau, D. A. (1989). ALVINN: an autonomous land vehicle in a neural network, *Advances in neural information processing systems 1*, Morgan Kaufmann Publishers Inc., San Francisco, CA

Schultz., W. (1998). Predictive reward signal of dopamine neurons. *Journal of Neurophysiology*, vol. 80, p. 1–27.

Schultz, W., Dayan, P., Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, vol. 275, p. 1593–1599.

Sutton, R.S., Barto, A.G. (1998). Reinforcement Learning, MIT Press, Cambridge, Mass.

Szita, I., Lörincz, A., (2007). Learning to Play Using Low-Complexity Rule-Based Policies: Illustrations through Ms. Pac-Man, *Journal of Articial Intelligence Research 30*, p. 659-684

Wiering, M., Hasselt, H. (2007). Two Novel On-policy Reinforcement Learning Algorithms based on TD(lambda)-methods. *Proceedings of IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL07)*, p. 280-287