Univerzita Komenského, Bratislava
Fakulta Matematiky, Fyziky a Informatiky
Katedra Informatiky

Trénovanie neurónových sietí s echo stavmi pomocou posilňovania

(bakalárska práca)

Viliam Dillinger

Študijný odbor: 9.2.1 Informatika
Vedúci práce: doc. Ing. Igor Farkaš, PhD.

Bratislava, 2009

Abstrakt

Cieľom mojej práce bolo ukázať účinnosť neurónových sietí s echo stavi na riešenie sekvenčných problémov v čiastočne pozorovateľnom prostredí pri použití učenia sa posilňovaním. V rámci dvoch experimentov som implementoval dve siete riešiace dva problémy: Tiger za dverami a Bludisko. Taktiež som popísal podrobný postup implementácie. Pri prvom experimente som skúmal priemernú úspešnosť siete v závislosti od veľkosti rezervoára a priemernú úspešnosť siete vzhľadom na pomer odmeny a trestu pri dvoch veľkostiach rezervoára. V druhom experimente som skúmal priemernú úspešnosť siete pri dvoch rôznych bludiskách.

Kľúčové slová: neurónová sieť s echo stavmi, učenie sa posilňovaním, sekvenčné problémy, čiastočne pozorovateľné prostredie

Abstract

Goal of my thesis was to show effectivity of solving sequential problems in partially observable enviroment with echo state networks and reinforcement learning. In two experiments I implemented two networks solving two problems: Tiger behind door and Maze. I have described exact process of implementation. In first experiment I describe connection bteween size of reservoair and average performance and connection bteween proportion of reward and penalty, size of reservoair and average success rate. In second experiment I describe average performance of network in two different mazes.

Key words: echo state network, reinforcement learning, sequential problems, partially observable enviroment

ČESTNÉ VYHLÁSENIE

Čestne vyhlasujem, že som bakalársku prácu vypracoval samostatne, a že som uviedol všetku použitú literatúru súvisiacu so zameraním bakalárskej práce.

Bratislava .....

.....

Viliam Dillinger

POĎAKOVANIE

Touto cestou vyslovujem poďakovanie doc. Ing. Igorovi Farkašovi, PhD. za pomoc, odborné vedenie, cenné rady a pripomienky pri vypracovaní mojej bakalárskej práce.

Bratislava ....

Viliam Dillinger

Obsah

1 Úvod

Mnoho problémov z nášho každodenného života má charakter sekvenčných rozhodovacích procesov. Do tejto kategórie problémov patrí väčšina spoločenských hier, nájdenie najkratšej cesty respektíve poradie krokov na dosiahnutie nášho cieľa. Na tieto problémy sa veľmi často používajú rôzne algoritmy, ako napríklad minimax s alfa-beta orezávaním.

Naozajstné problémy však začínajú, ak náš problém nie je plne pozorovatľný, ale len čiastočne pozorovateľný. Ako by sa, napríklad, zmenila Vaša hra šachu, keby ste nevideli celú šachovnicu, ale len políčka, na ktorých stoja Vaše figúrky a políčka, na ktoré môžu stúpiť? Ďalšie sťaženie úlohy by bolo, keby sme ju previedli do priestoru, ktorý sa v čase mení a nevieme predvídať ako. Bežné algoritmy, mnohokrát veľmi jednoduché, by, pri snahe toto všetko obsiahnuť, mohli byť časovo a programátorsky veľmi náročné.

Jednou z metód, ako riešiť tieto problémy, je použitie neurónových sietí, ktoré sú vynikajúce do takéhoto prostredia nakoľko sú prispôsobivé a pri ich vytváraní nehovoríme, ako majú problém riešiť, ale použítím učiacich algoritmov im implicitne dáme návod, ako sa k nemu môžu dopracovať.

Pri mnohých algoritmoch a technikách, aby správne fungovali, potrebujeme okrem vstupov aj množstvo informácií od prostredia, aby sme úlohu mohli riešiť. V prípade neurónových sietí s echo stavmi trénovanými pomocou posilňovania (reinforced learning) nám stačí jedno jediné skalárne číslo - odmenu/trest, ktorú nám dáva prostredie podľa toho, ako sa v ňom správame. Napríklad, ak by sme chceli robiť agenta hrajúceho pinball, na jeho trénovanie by sme mohli použiť počet bodov, ktoré nahral.

Kombináciou neurónových sietí s echo stavmi a technikou učenia pomocou posilňovania sa zaoberá moja práca. Mnoho ľudí odsudzuje neurónové siete a podceňuje ich účinnosť a pritom často bývajú vynikajúcim riešením mnohých problémov, hlavne tých, ktoré sa týkajú rozlišovania a kategorizácie. Táto vlastnosť sa dá vynikajúco využiť v čiastočne pozorovateľných prostrediach, kde sieť skúsi rozoznať podľa daného pozorovania a pamäte, v akom stave sa nachádza. Následne zostáva už len namapovať stavy na akcie.

Rád by som ešte spomenul dve vlastnosti neurónovych sietí, ktoré sú podľa mňa veľmi dôležité.

Ak máme robota v reálnom svete, mnohokrát dostáva zo svojich receptorov

poškodené vstupy, napríklad zašpinenú kameru, Neurónové siete, vo všeobecnosti, sú na riešenie takýchto úloh vynikajúce, nakoľko aj z poškodeného vstupu dokážu dostať, po vhodnom natrénovaní, veľmi dobré výsledky.

Druhá vlastnosť, ktorú som chcel spomenúť, má dvojsečný charakter. Ak natrénovaného agenta necháme pracovať, môžeme ho nechať ďaľej sa učiť. To môže mať za následok dve veci. Ak sa prostredie zmení, agent môže byť schopný znovu sa adaptovať aj na zmenené prostredie bez nutnosti zásahu zvonku. Problémom však môže byť kolísajúci charakter výkonnosti - agent v snahe nájsť lepšie riešenie môže prejsť k horšiemu. Samozrejme existuje mnoho spôsobov, napríklad využívanie checkpointov, aby sa vedel dostať späť k pôvodnému lepšiemu riešeniu.

Moja práca sa delí na dve časti - teoretickú, v ktorej interpretujem teoretické vedomosti s ohľadom na moje ciele a experimentálnu, zaoberajúcu sa mojou implementáciou, pokusmi a meraniami.

V teoretickej časti popisujem Markovove rozhodovacie procesy (MDP) ako matematický formalizmus na modelovanie rozhodnutí v situáciách, kde následky sú čiastočne náhodné a čiastočne pod kontrolou rozhodujúceho. Následne opisujem algoritmy Q-učenie (Q-learning) a SARSA (State-Action-Reward-State-Action alebo Stav-Akcia-Odmena-Stav-Akcia) ako vhodné bezmodelové riešenia Markovovych rozhodovacích procesov, ako aj ϵ-greedy skúmanie (ϵ-greedy exploration) ako vhodný prostriedok na podporu skúmania. Ďalej píšem o čiastočne pozorovateľných Markovovych rozhodovacích procesoch (POMDP - Partially observable MDP) a o ich riešení cez konverziu na MDP a spojitom priestore predpokladov stavov.

V závere tejto časti popisujem neurónové siete s echo stavmi ako aproximátory funkcie odhadu stavu pri riešení POMDP.

V praktickej časti podrobne popisujem implementáciu neurónovej siete s echo stavmi, ktorú som použil pri experimentoch, ako aj oba experimenty.

V rámci prvého experimentu som učil sieť hrať hru Tigrer za dverami a opisujem merania, ktorými som sledoval priemernú úspešnosť siete v závislosti na veľkosti jej rezervoára a úspešnosť sieťe v závislosti od pomeru veľkosti odmeny a trestu a veľkosti jej rezervoára, ako aj presný spôsob trénovania, ktoré som použil.

V druhom experimente sa sieť učila orientovať v bludisku. Opisujem spôsob trénovania siete pri tomto probléme, ako aj výsledky úspešnosti siete počas neho. Taktiež som opísal pokus, v ktorom som zmenil bludisko.

2 Teória

V tejto časti popisujem Markovove rozhodovacie procesy (MDP) ako matematický formalizmus na modelovanie rozhodnutí v situáciách, kde následky sú čiastočne náhodné a čiastočne pod kontrolou rozhodujúceho. Následne opisujem algoritmy Q-učenie (Q-learning) a SARSA (State-Action-Reward-State-Action alebo Stav-Akcia-Odmena-Stav-Akcia) ako vhodné bezmodelové riešenia Markovovych rozhodovacích procesov, ako aj ϵ-greedy skúmanie (ϵ-greedy exploration) ako vhodný prostriedok na podporu skúmania. Ďalej píšem o čiastočne pozorovateľných Markovovych rozhodovacích procesoch (POMDP - Partially observable MDP) a o ich riešení cez konverziu na MDP a spojitom priestore predpokladov stavov.

V závere tejto časti popisujem neurónové siete s echo stavmi ako aproximátory funkcie odhadu stavu pri riešení POMDP.

2.1 Sekvenčné rozhodovacie procesy

2.1.1 Markovove rozhodovacie procesy (MDP)

Tento proces, pomenovaný podľa Andreya Markova, ponúka matematický formalizmus na modelovanie rozhodnutí v situáciách, kde následky sú čiastočne náhodné a čiastočne pod kontrolou rozhodujúceho. [6]

Predstavte si hrací automat s n pákami. Po potiahnutí každej páky môžete, ale aj nemusíte získať nejaké body, poprípade o niekoľko z nich prísť, pričom dôvod, prečo vám ich automat dá, alebo zoberie, nemusí závisieť od terajšieho potiahnutia páky, ale aj od potiahnutí v minulosti. Vaším cieľom je nahrať na takomto automate čo najviac bodov.[1]

Formálne povedané, nech:

- S je konečná množina stavov, $s \in S$;
- A je konečná množina akcií, $a \in A$;
- $P_a(s, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ je pravdepodobnosť, že ak v stave s vykonáme akciu a dostaneme sa v čase $t + 1$ do stavu s';
- $R_a(s, s')$ je okamžitá odmena obdržaná po prejdení zo stavu s do stavu s' s pravdepodobnosťou $P_a(s, s')$.

Naším cieľom je maximalizovať odmenu v budúcnosti, teda maximalizovať:

$$\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1})$$

kde γ je parameter kvantifikujúci mieru zohľadnenia budúcich odmien (discount rate) a $0 < \gamma \leq 1$. Pri $\gamma = 1$ berie do úvahy celú epizódu.

Riešenie je vcelku jednoduché. Musíme vytvoriť stratégiu $\pi(s) = a$, v ktorej namapujeme stavy na akcie, ktoré je v nich najlepšie vykonať. $V(s)$ je ohodnotenie discontnej odmeny v stave s.

$$\pi(s) := \operatorname*{argmax}_a R_a(s, s') + \sum_{s'} P_a(s, s') V(s')$$

$$V(s) := R(s) + \gamma \sum_{s'} P_{\pi(s)}(s, s') V(s')$$

Ak sa teda budeme riadiť našou stratégiou, dosiahneme optimálne riešenie problému.

2.1.2 Učenie sa posilňovaním

Ak pri riešení MDP nepoznáme pravdepodobnosti, problém patrí do kategórie učenia sa posilňovaním (reinforcement learning). [4]

Na riešenie tohto problému sa dá použiť viacero algoritmov, z ktorých tie najpoužívanejšie a pre moju úlohu zaujímavé, predstavím v tejto časti.

Q-učenie(Q-learning)

Základnou myšlienkou je pomocou prírastkov odhadnúť hodnoty párov stav-akcia, Q-hodnoty, založené na odskúšaných odmenách v prostredí a agentovými vlastnými odhadmi Q-hodnôt. Adaptačné pravidlo Q-učenia v svojej najjednoduchšej forme vyzerá takto:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha[r_{t+1} + \gamma \operatorname*{argmax}_a Q_t(s_{t+1}, a_t) - Q_t(s_t, a_t)]$$

kde α je rýchosť učenia a r je obdržaná odmena. [5]

Pred začiatkom učenia Q-funkcia vracia pevnú hodnotu, určenú dizajnérom. Potom vždy, keď agent dostane odmenu (stav sa zmení), nové hodnoty sú vyrátané pre každú kombináciu stavu $s \in S$ a akcie $a \in A$. Jadro algoritmu je jednoduché iteratívne aktualizovanie hodnôt. Predpokladá starú hodnotu a urobí korekciu založenú na nových informáciách. [8]

Q-funkcie sa môžu zdať iba ako ďalší spôsob skladovania informácií o účinnosti, ale majú veľmi dôležitú vlastnosť: agent učiaci sa Q-funkciu nepotrebuje mať model prostredia ani pre učenie, ani pre výber akcií. Z tohoto dôvodu sa Q-učenie niekedy nazýva aj metódou bez modelu (model-free method). [1]

SARSA

SARSA (State-Action-Reward-State-Action alebo Stav-Akcia-Odmena-Stav-Akcia) je ďalším algoritmom využívaným pri učení sa posilňovaním. Na rozdiel od Q-učenia, kde stratégiu, ktorú agent sleduje nie je zhodná s tou, ktorú učí (off-policy learning), pri algoritme SARSA agent sleduje rovnakú stratégiu, ktorú sa učí (on-policy learning) a

vyzerá nasledovne:

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

Ako je vyjadrené hore, Q-hodnota pre stav a akciu je aktualizovaná podľa chyby, regulovaná rýchlosťou učenia. Q-hodnoty reprezentujú možnú odmenu, ktorú agent dostane v ďalšom kroku ak vykoná v stave s akciu a plus discontované budúce odmeny obdržané po nasledujúcom pozorovaní stavu a akcie. [9]

Toto pravidlo bude stále konvergovať k optimálnym hodnotám a optimálnej stratégii, pokiaľ stratégia konverguje v medznej hodnote ku greedy stratégii, pri ktorej už nepotrebuje ďalej objavovať.

SARSA má výhodu oproti Q-učeniu, keď agent celý svoj život musí skúmať svoje prostredie, napríklad ak sa nemá prestať učiť. V tomto prípade chceme, aby sa naučil stratégiu, ktorá bude ďalej skúmať prostredie a nie hypotetickú greedy stratégiu, ktorú aj tak nikdy nebude používať. Ďalšia výhoda je spätá s použitím aproximátorov funkcií namiesto tabuliek reprezentujúcich Q-učenie v niektorých prípadoch môže divergovať kvôli svojej off-policy povahe, avšak SARSA nie. Predsa len obvykle dáva väčší zmysel učiť sa hodnotu stavu založenú na odhade najlepšej akcie, ktorú môže agent urobiť v ďalšom stave (Q-učenie), ako akciu, ktorá môže byť skúmajúca a veľmi ďaleko od najlepšej. [5]

ϵ-greedy skúmanie (ϵ-greedy exploration)

Ak máme bezmodelovú metódu, pravdepodobne pri výbere akcie používame greedy techniku a vyberáme vždy najlepšie ohodnotenú akciu. Ak však naše riešenie skonverguje k istej hodnote, môžeme si byť istí, že sme natrafili na globálne a nie len lokálne maximum? Agent by teda mal svoje okolie neustále skúmať a niekedy vybrať akciu, ktorá síce nevedie k zatiaľ najlepšiemu riešeniu, ktoré poznáme, ale skúša nájsť lepšie riešenie.

Jednou z takýchto techník je aj ϵ-lačné skúmanie, kedy agent s istou malou pravdepodobnosťou ϵ vyberie náhodnú akciu. Toto sa dá zrealizovať napríklad tým, že mierne pozmeníme výber akcie. Nech je agent v stave s, potom je pravdepodobnosť $p(s, a)$, že si vyberie akciu a, kde

$$p(s, a) = \frac{e^{Q(s,a)/\tau}}{\sum_{n=1}^{N} e^{Q(s,a_n)/\tau}}$$

kde N je počet možných akcií a τ je teplotný parameter. Čím je vyšší, tym je väčšia pravdepodobnosť, že agent bude voliť inú ako (zatiaľ) ideálnu akciu a teda viacej bude skúmať prostredie. Ak sa τ blíži k 0, z výberu sa stáva greedy výber. [5]

2.1.3 Čiastočne pozorovateľné Markovove rozhodovacie procesy (POMDP)

Čiastočne pozorovateľné MDP (POMDP - Partially observable MDP), sú omnoho zložitejšie ako MPD. Môžu byť vyriešené konverziou na MDP v spojitom priestore predpokladov stavov. Optimálne správanie v POMDP obsahuje zhŕňanie informácií na redukovanie pochybností. [1]

Formálne, nech:

- S je konečná množina stavov, $s \in S$;
- A je konečná množina akcií, $a \in A$;
- O je konečná množina pozorovaní, $o \in O$;
- $P_a(s, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ je pravdepodobnosť, že ak v stave s vykonáme akciu a dostaneme sa v čase $t + 1$ do stavu s';
- $R_a(s, s')$ je okamžitá odmena obdržaná po prejdení zo stavu s do stavu s' s pravdepodobnosťou $P_a(s, s')$.

V čase t je prostredie v stave s. Agent vykoná akciu a a to s pravdepodobnosťou $P_a(s, s')$ zmení stav prostredia na s'. Následne agent dostane odmenu $R_a(s, s')$, ktorú sa v čase snažíme maximalizovať.

Agent však nevie, v akom stave sa nachádza prostredie, môže to len odhadovať podľa pozorovaní ktoré dostane. Nech teda $b(s)$ je pravdepodobnosť, že svet je v stave s a nech agent v stave s' dostane pozorovanie o s pravdepodobnosťou $P_a(o, s')$. Ak sme s pravdepodobnosťou $b(s)$ boli v stave s, vykonali akciu a a dostali pozorovanie o potom:

$$b'(s') = \eta P_a(o, s') \sum_{s \in S} P_a(s, s') b(s)$$

kde

$$\eta = P_a(o, b)$$

$$P_a(o, b) = \sum_{s' \in S} P_a(o, s) \sum_{s \in S} P_a(s, s') b(s)$$

Na predpoklad stavu, do ktorého sme sa dostali z predpokladu b akciou a a po pozorovaní o sa dá zapísať ako $b' = \tau(b, a, o)$.

Doteraz sme sa pohybovali v diskrétnom stavovom priestore. Náš problém sa však dá transformovať do spojitého priestoru na takzvaný predpokladový MDP (belif MDP). Nech:

- B je množina predpokladaných stavov nad POMDP stavmi, $b \in B$
- A je konečná množina akcií, rovnaká ako v originálnom POMDP
- τ je prechodová funkcia predpokladaných stavov
- $r(b, a)$ je predpokladaná odmena pri vykonaní akcie a v predpoklade stavu b, $r(b, a) = \sum_{s \in S} b(s)R(s, a)$

Pri riešení POMDP sa snažíme nájsť stratégiu $\pi^*(b) = a$ - ktorú akciu je najlepšie vykonať ak sme v predpokladanom stave b.

Predpokladaná odmena stratégie π a počiatočného predpokladaného stavu b bude:

$$J^{\pi}(b) = E\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s'_t)|b, \pi\Big]$$

a optimálna stratégia bude

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, J^{\pi}(b_0)$$

kde b_0 je počiatočný predpokladaný stav. Na výpočet odmeny, ktorú dostaneme, keď budeme postupovať podľa optimálnej stratégie bude [7]

$$V^*(b) = \underset{a}{\operatorname{argmax}} \Big[r(b, a) + \gamma \sum_{o \in O} P_a(b, a)V^*(\tau(b, a, o))\Big]$$

Pri riešení POMDP sa snažíme aproximovať naše riešenie k ideálnemu. Na tento účel sa dajú použiť aj neurónové siete s echo stavmi, ktoré popíšem v nasledujúcej kapitole.

2.2 Neurónové siete s echo stavmi

Mnoho problémov z reálneho sveta je veľmi zložitých a ťažko v nich môžeme hovoriť o celej množine stavov, v ktorom sa môže nachádzať.

Napríklad šach a backgammon sú len malá podmnožina reálneho sveta a to ich stavový priestor obsahuje od 10^{50} do 10^{120}. Bolo by absurdné predpokladať, že ich musíme všetky navštíviť, aby sme sa naučili hrať danú hru. Jediná cesta, ako vyriešiť takéto problémy, je použiť aproximácie funkcie, čo znamená použiť hociakú reprezentáciu pre funkciu okrem tabuľky. [1]

Neurónové siete s echo stavmi môžu byť vnímané ako aproximátory funkcie, konajúce na základe vnútorného stavu vyvinutého zo sérií predchádzajúcich vstupov, ktorými minulosť zakomponujú do reprezentácie stavu. Pozorovania v čase t samostatne nie sú dostačujúce na výber optimálnej akcie, ale spolu s vnútornou reprezentáciou budú mať Markovovu vlastnosť.

Vnútorný stav $\mathbf{u}_t$ v čase t sa dá vyrátať nasledovne:

$$\mathbf{u}_t = \sigma(\mathbf{F}\mathbf{u}_{t-1} + \mathbf{G}\mathbf{x}_t)$$

kde σ je monotónna nelineárna funkcia (napríklad hyperbolický tangens), $\mathbf{F}$ je rekurentná matica siete, ktorá musí byť náhodná, riedka a musí mať spektrálny rádius menší ako 1, $\mathbf{G}$ je vstupná váhová matica a $\mathbf{x}_t$ je vstupný vektor v čase t.

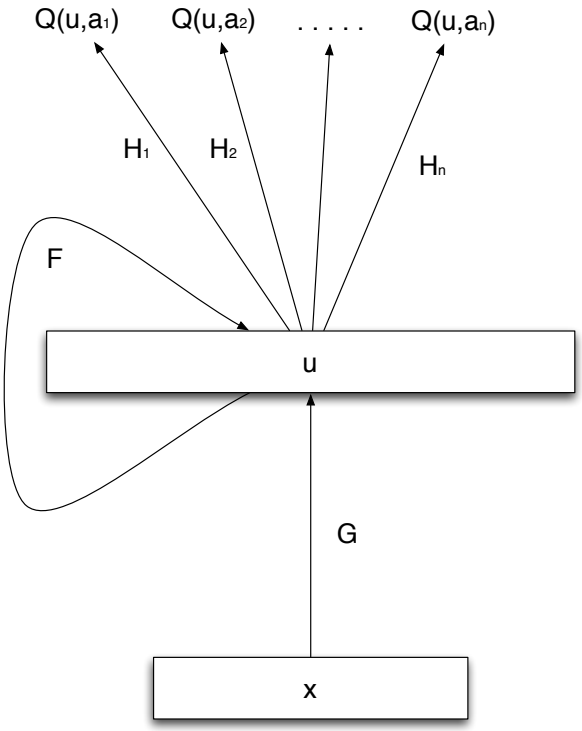
Predpokladajme, že máme n akcií. Pre každú z nich bude mať sieť jeden výstup. Cieľom je natrénovať maticu $\mathbf{H}$, pretože vďaka nej môžeme vyrátať Q-hodnoty akcií v stave u:

$$Q(\mathbf{u}, a_i) = \mathbf{H}_i\mathbf{u}$$

kde $\mathbf{H}_i$ reprezentuje i-ty riadok matice $\mathbf{H}$. Na trénovanie použijeme nasledovné aktualizačné pravidlo:

$$\mathbf{H}_i^{t+1} = \mathbf{H}_t^i + \alpha_t(d^t - o_i^t))\mathbf{u}_t$$

kde α_t je rýchlosť učenia v čase t, d^t je trénovacia hodnota v čase t a o_i^t je i-ty výstup v čase t. [3]



Obr. 1: Grafické zobrazenie fungovania neurónovej siete s echo stavmi

3 Experimenty

Táto práca je inšpirovaná prácou Szita et. al. [3], kde opísali fungovanie týchto neurónových sietí a ukázali ich účinnosť na troch problémoch. V ich práci však neboli spomenuté parametre ich siete (počet neurónov v rezervoári, rýchlosť učenia, atď.) a ani presný spôsob trénovania, preto som sa rozhodol v rámci experimentálnej časti tejto práce dva z problémov vyriešiť. Konkrétne sa jednalo o Tiger za dverami (Tiger problem) a Bludisko (3x4 maze problem).

3.1 Postup pri implementácii

Implementáciu som realizoval v jazyku Java v prostredí Eclipse. Použil som balíček Jama - Java Matrix class, ktorý mi pomohol pri práci s maticami, pretože operácie ako násobenie matíc, sčítanie matíc a vyrátanie vlastného čísla v ňom boli implementované a teda som sa mohol sústrediť na trénovanie.

Vrstvy (reprezentované vektormi) a prepojenia (reprezentované maticami) v sieti sa inicializovali nasledovne:

- vstupný ($\mathbf{v}_{in}$), výstupný ($\mathbf{v}_{out}$), skrytý ($\mathbf{v}_{hid}$) a rekurentný ($\mathbf{v}_{rec}; |\mathbf{v}_{rec}| = |\mathbf{v}_{hid}|$) vektor na 0;
- maticu prepojení vstupnej a skrytej vrstvy ($\mathbf{P}_{ih}$) a maticu prepojení skrytej a výstupnej vrstvy ($\mathbf{P}_{ho}$) na náhodné hodnoty v intervale $\langle -1, +1 \rangle$;
- maticu prepojení vstupnej a skrytej vrstvy ($\mathbf{P}_{rh}$) na náhodné hodnoty v intervale $\langle -1, +1 \rangle$. Následne som z nej spravil riedku maticu tým, že som vynuloval 80% jej prvkov. Aby som zabezpečil, že sa signál bude v čase tlmiť, vyrátal som najväčšie vlastné číslo, ním predelil každý prvok a následne ho vynásobil 0.9.

Rozhodovací proces siete začal nastavením vstupov (podľa danej úlohy). Pretože sieť bola veľmi citlivá na veľkosť čísel na vstupe a príliš veľké hodnoty robili problémy v skrytej vrstve (pri použití hyperbolického tangensu stavy splývali), všetky čísla, ktoré som dával na vstup, boli vynásobené číslom f, ktoré záviselo od veľkosti rezervoára a v oboch úlohách sa líšilo.

Tento problém sa dal tiež vyriešiť inicializáciou prepojovacej matice $\mathbf{P}_{ih}$ na menšie hodnoty, napríklad hodnoty z $\langle -0.2, +0.2 \rangle$.

Následne sa skopíroval $\mathbf{v}_{hid}$ do $\mathbf{v}_{rec}$, ktorý takto simuloval rezervoár v čase $t - 1$. Aktivácia skrytej vrstvy vyrátala ako

$$\mathbf{v}_{hid} = \sigma \left(\mathbf{P}_{ih} \mathbf{v}_{in} + \mathbf{P}_{rh} \mathbf{v}_{rec} \right)$$

kde σ bola funkcia hyperbolický tangens:

$$\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Aktivácia výstupnej vrstvy, $\mathbf{v}_{out}$, ktorá reprezentovala ohodnotenie jednotlivých

akcií, sa vyrátala pomocou

$$\mathbf{v}_{out} = \mathbf{P}_{ho}\mathbf{v}_{hid}$$

Vybrala sa akcia s najväčším ohodnotením a tá sa vykonala (ako, to záležalo už na danom probléme). Ďalším krokom bolo trénovanie, kde sa trénovala len prepojovacia matica $\mathbf{P}_{ho}$. Na aké trénovacie hodnoty, a ktoré akcie sa trénovali, už bol problém konkrétnej úlohy a preto ho rozoberám pri jednotlivých úlohách. Ak sa však trénovala niektorá akcia, robilo sa to pomocou tohto vzťahu

$$\forall i : \mathbf{P}_{ho_{i,j}} = \mathbf{P}_{ho_{i,j}} + \alpha_t(d - \mathbf{v}_{out_j})\mathbf{v}_{hid_i}$$

kde α_t je rýchlosť učenia v čase t, ktorú popíšem pri konkrétnej úlohe, pretože som v oboch prípadoch použil rôzne funkcie a d je trénovacia hodnota, ktorú dostáva sieť pri trénovaní podľa toho, ako funguje a symbolizuje odmenu alebo trest.

Pri testovaní úspešnosti som pri výbere akcie postupoval rovnako, ale sieť som následne netrénoval.

3.2 Experiment 1: Tiger za dverami

Mojím prvým experimentom bola implementácia učenia sa jednoduchej hry. Majme svet a v ňom dvoje dverí. Vieme, že za jednými je tiger, ktorý vrčí a za druhými je poklad. Náš agent má 3 možné akcie - otvoriť ľavé dvere, otvoriť pravé dvere a počúvať. Pri počúvaní agent počuje tigra vrčať z jedného zo smerov. S pravdepodobnosťou $p \in \langle 0, 1\rangle$ ho počuje vrčať zo správnej strany. Agent dostáva veľkú odmenu, ak otvorí dvere, za ktorými je poklad, veľký trest, ak otvorí dvere, za ktorými sa nachádza tiger, alebo malý trest, ak počúva. Hra sa končí otvorením jedných z dverí.

Cieľom bolo natrénovať sieť, aby niekoľkokrát počúvala a podľa toho sa správne rozhodla a otvorila dvere s pokladom.

Pri výstupoch, ktoré ďalej prezentujem, som uvažoval $p \geq 0.6$, pretože pri $p \leq 0.5$ sa sieť učila rovnako ako pri $(1 - p)$ a medzi pravdepobnosťou 0.5 a 0.6 už sieť mohla len tipovať a na vstupoch záležalo minimálne.

3.2.1 Vstupy

Sieť mala dva vstupy - hluk sprava a hluk zľava. Na začiatku boli obidve 0, avšak pri počúvaní sa nastavili jeden na hodnotu 0 a druhý na hodnotu 1, podľa pozície tigra. S pravdepodobnosťou $(1 - p)$ však boli vstupy, podľa zadania úlohy, vymenené. Následne som vynásobil vstupný vektor číslom $f = 30/|\mathbf{v}_{hid}|$. Dôvod je vysvetlený v predchádzajúcej časti.

3.2.2 Učenie

Pri trénovaní neurónovej siete pre túto úlohu bolo potrebné, aby sa rýchlosť učenia v čase zmenšovala a teda sieť konvergovala k istému stavu, zvolil som rýchlosť učenia v čase t:

$$\alpha_t = \alpha_{zac} - (\alpha_{zac} - \alpha_{kon}) \left(\frac{t}{t_{max}} \right)$$

pričom parametre $\alpha_{zac} = 0.4$, $\alpha_{kon} = 0.005$ a $t_{max} = 2000$ bol počet hier, počas ktorých sa neuronová sieť učila. Neskôr som si všimol, že α_t nespĺňa Robbins-Monrovo kritérium[1] , ale aj tak sa to bez problémov dokázalo naučiť.

[1] $\alpha_t > 0, \sum_{t=0}^{\propto} \alpha_t = \propto, \sum_{t=0}^{\propto} \alpha_t^2 < \propto$

Po vykonaní akcie som trénoval nasledovne. Ak zvolila:

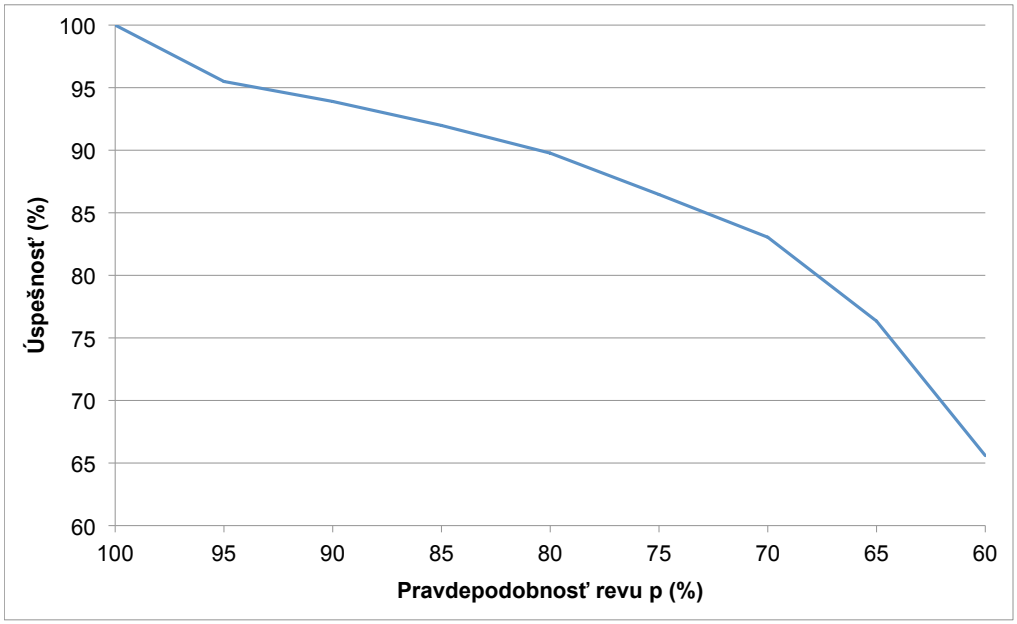
- otvorenie ľavých(pravých) dverí a za nimi našla poklad, trénoval som otvorenie ľavých(pravých) dverí s $d = 5$;
- otvorenie ľavých(pravých) dverí a za nimi našla tigra, trénoval som otvorenie ľavých(pravých) dverí s $d = -150$;
- počúvanie, trénoval som počúvanie s $d = -1$ a otvorenie obidvoch dverí s $d = 0$.

Pri trénovaní siete sa mnohokrát stalo, že ohodnotenie otvorenia oboch dverí sa dostalo nižšie ako napríklad -5 a počúvanie skonvergovalo k svojej hodnote -1. Tým pádom neurónová sieť stále vyberala počúvanie. Ak sa pri trénovaní počúvania trénovali aj otvárania dverí, v prípade, ak sieť príliš dlho počúvala, po istom čase ohodnotenie otvorenia jedných z dverí bolo vyššie a ukončilo hru. Tento problém vznikal, nech som použil akékoľvek d pri trénovaní, takže toto opatrenie bolo nutné.

Nepomer odmeny za otvorenie dverí s pokladom a trestu za otvorenie dverí s tigrom, je tiež úmyselné. Keď boli obidve hodnoty rovnaké (len s opačným znamienkom), neurónová sieť sa správala ako človek závislý na hracích automatoch, ktorý neustále čaká, že vyhrá. Pri mnohonásobne väčšom treste si sieť dávala pozor, aby nespravila chybu. Pri tomto riešení ma inšpirovala výchova malého dieťaťa, kedy za jednotku v škole dostane cukrík, ale ďalší deň už na to zabudne. Ak ale dostane zlú známku, dostane domáce väzenie a musí sa doma učiť, čo má za následok jeho snahu, aby už zlú známku nedostal.

3.2.3 Výsledky

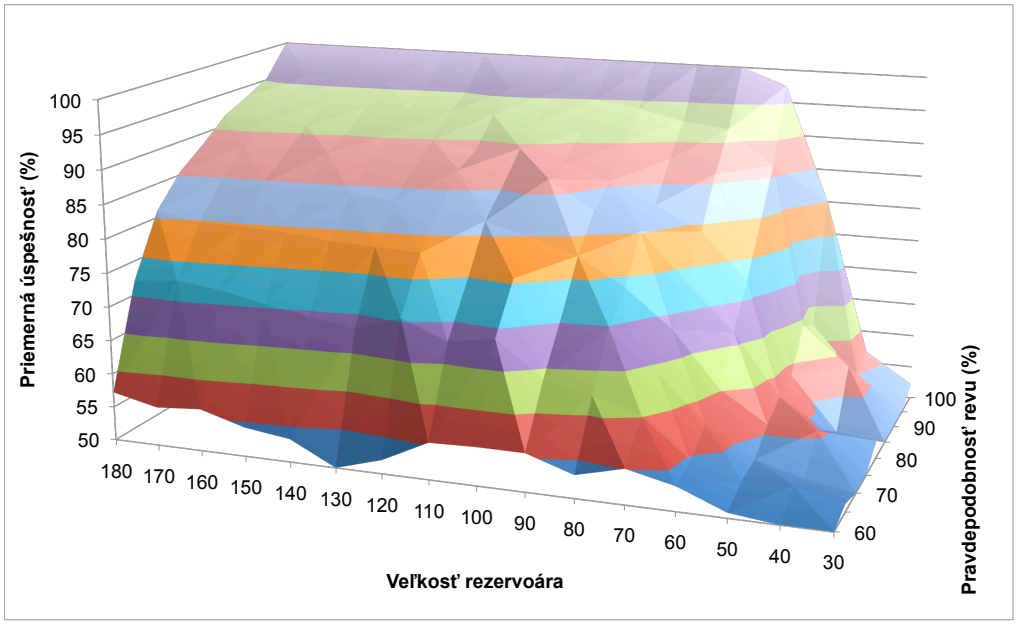
Pri prvých pokusoch som používal počet neurónov v rezervoári 300. Sieť sa naučila riešiť tento problém veľmi dobre – ak by raz počúvala, a podľa toho sa rozhodla hneď otvoriť dvere, mala by úspešnosť p. Moja natrénovaná sieť mala úspešnosť vždy väčšiu, niekedy až o 15%, čo ukazuje graf na obrázku 2. Pre každú pravdepodobnosť p som rátal priemernú úspešnosť 100 neurónovych sietí. Úspešnosť konkrétnej neurónovej siete som vyrátal ako priemer z úspešnosti 300 testovacích hier po natrénovaní danej neurónovej siete.



Obr. 2: Výsledky úspešnosti pri experimente 1: Tiger za dverami, veľkosť rezervoára 300

Trénovanie hore popísanej neurónovej siete trvalo príliš dlho, nakoľko ak je v skrytej vrstve 300 skrytých neurónov, pri každom rozhodovaní sa musí násobiť veľkými maticami, čo je časovo náročné. Preto som sa začal zamýšľať nad otázkou. Aký najmenší rezervoár potrebuje táto neurónová sieť na to, aby sa túto úlohu dokázala dobre naučiť?

Spravil som sériu pokusov a ich výsledok je na obrázku 3. Pre každé p a veľkosť rezervoáru som vyrátal priemernú úspešnosť 100 neurónových sietí. Úspešnosť danej siete som vyrátal ako priemer úspešnosti 300 testovacích hier, ktoré odohrala po natrénovaní. Tieto výpočty boli časovo veľmi náročne – hodnoty pre tento graf sa rátali asi 30 hodín.

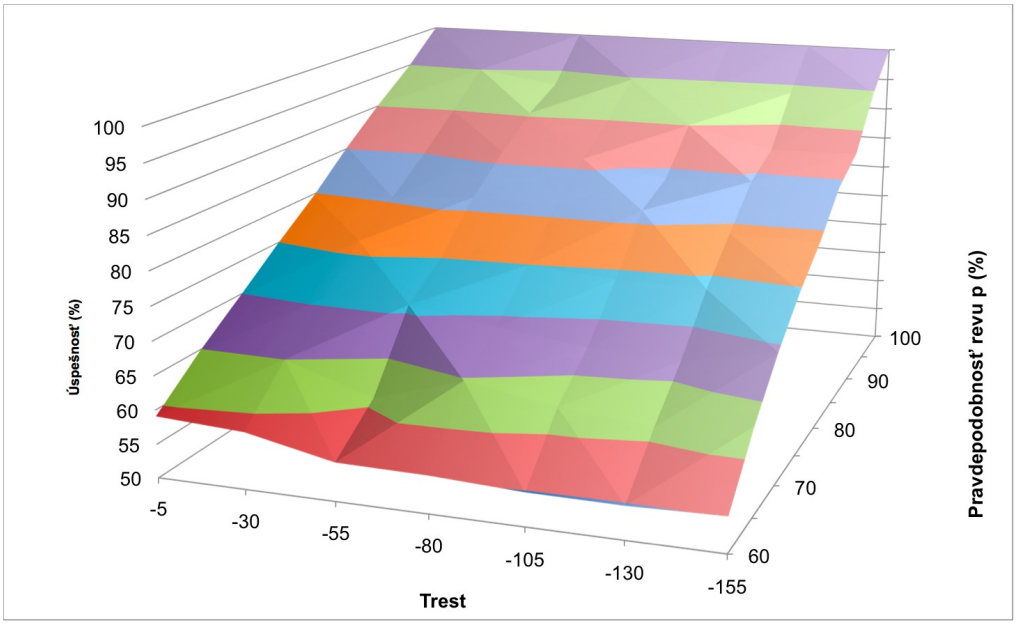


Obr. 3: Výsledky úspešnosti vzhľadom na veľkosť rezervoáru pri experimente 1: Tiger za dverami

Z daných výsledkov som usúdil, že ak má rezervoár veľkosť menšiu ako 40 neurónov, nevie sa vôbec naučiť tento problém. Od veľkosti 40 až po 70 nastáva obrovský nárast úspešnosti. Pri veľkosti 70 neurónov mala úspešnosť ako keby raz počúvala a podľa toho sa rozhodla (priemerná úspešnosť pripomínala lineárnu funkciu). Pri ďalšom zvyšovaní veľkosti rezervoáru graf úspešnosti siete pomaly mení tvar smerom ku grafu zobrazenom na obrázku 2. Predpokladám, že pri ešte väčšej veľkosti rezervoára by sa úspešnosť zvyšovala, ale čím ďalej, tým menej.

Ak by bola naším cieľom sieť, ktorá sa musí rýchlo učiť a rýchlo sa rozhodovať,

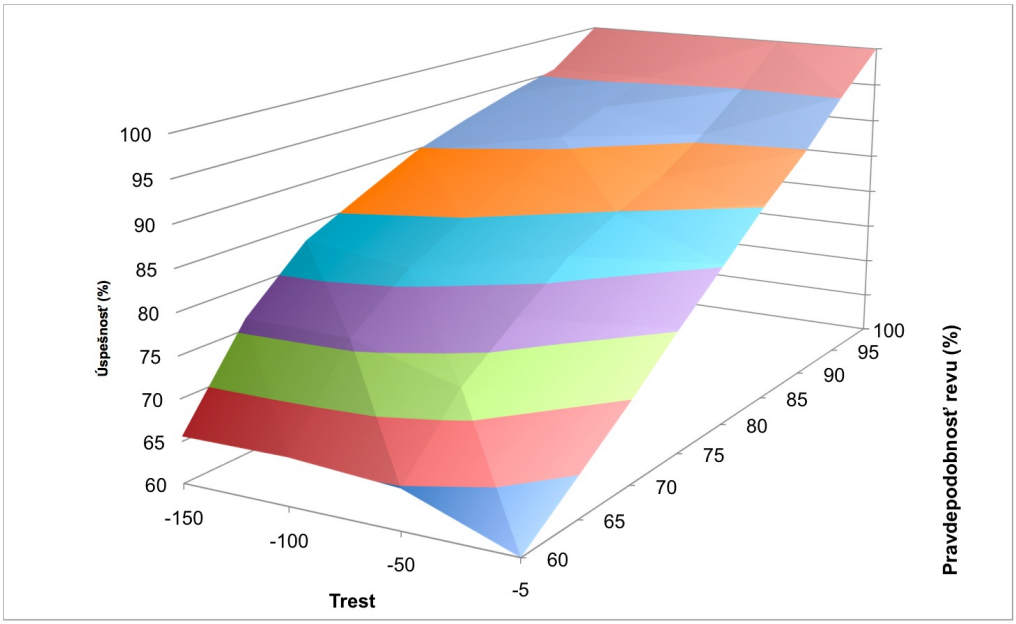
určite by som volil rezervoár veľkosti 180. Ak by nám ale šlo o pár percent úspešnosti navyše a čas by pre nás nebol až tak dôležitý, volil by som rezervoár veľkosti 300, poprípade viac.



Obr. 4: Výsledky úspešnosti vzhľadom na veľkosť trestu s rezervoárom veľkosti 80 pri experimente 1: Tiger za dverami

Vyššie uvedené tvrdenie tvrdiace, že zvýšenie trestu za otvorenie dverí s tigrom zvýši aj úspešnosť siete, som sa rozhodol podporiť experimentom. Pri prvom meraní som použil sieť s veľkosťou rezervoára 80, pričom trénovacia hodnota d nadobúdala hodnoty od -5 (rovnaká ako odmena za dvere s pokladom, len s opačným znamienkom), až po -155. Výsledky sú zobrazené na obrázku 4. Vidíme, že zvyšovanie trestu malo za následok zhoršenie siete. Toto správanie mi pripomína trestanie mladého neinteligentného delikventa, ktorý po ňom škodí ešte viac.

Pri druhom meraní (obrázok 5) som použil sieť s veľkosťou rezervoára 300 a trénovacia hodnota nadobúdala hodnoty -5 (rovnaká ako odmena za dvere s pokladom, len s opačným znamienkom), -50 a -100 ku ktorým som pridal pre porovnanie



Obr. 5: Výsledky úspešnosti vzhľadom na veľkosť trestu s rezervoárom veľkosti 300 pri experimente 1: Tiger za dverami

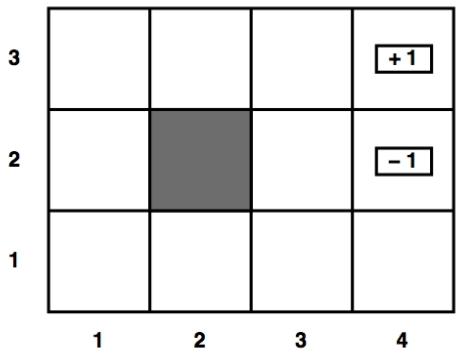
výsledky z obrázka 2. Ako si môžeme všimnúť, so zvyšujúcim sa trestom sieť dosahovala lepšie výsledky. Správala sa tak ako inteligentné dieťa, ktoré sa z trestu snaží poučiť.

Priemerná úspešnosť implementovaného modelu sa empiricky menila nielen na základne veľkosti rezervoára a pomeru veľkosti odmeny a trestu, ale aj na základe dvojice týchto hodnôt. Teoreticky však tento jav zdôvodniť neviem, ale pravdepodobne bude cieľom môjho ďalšieho skúmania.

V porovnaní s výsledkami z práce Szita et al. [3] moja najlepšie natrénovaná sieť bola horšia o maximálne 3%. Verím však, že ak by som mal viac času venovať sa tomuto problému, našiel by som riešenie rovnako úspešné ako mali oni.

3.3 Experiment 2: Bludisko

Tento problém bol pôvodne prevzatý z [1], kde bol ale prezentovaný ako plne pozorovateľný problém. Moja práca sa však zaoberá riešením čiastočne pozorovateľných problémov. Z toho dôvodu bola použtá upravená verzia tohto problému, ktorá bola použitá v [3].



Obr. 6: Bludisko v experimente 2

Náš agent sa nachádza v bludisku 3x4 ako je zobrazené na obrázku 6. Cieľom tejto úlohy je, aby sa sieť z náhodnej pozície vedela dostať pomocou pohybou na sever, juh, východ a západ k políčku $+1$ a vyhla sa políčku -1. Ak sa agent rozhodne ísť niektorým zo smerov, je 20% šanca, že pôjde smerom kolmým na smer ktorý určil (vľavo aj vpravo majú rovnakú pravdepodobnosť), napríklad ak sa rozhodne ísť na sever, je šanca 80%, že pôjde na sever, 10%, že pôjde na západ a 10%, že pôjde na východ. Ak by mal svojím pohybom naraziť do steny, ostáva na mieste, ale ako ťah sa to ráta. Pri každom pohybe dostáva malý trest, pri vstupe na políčko $+1$ dostáva veľkú odmenu a pri vstupe na políčko -1 dostáva veľký trest. Agent má dva senzory, ktorými vie vnímať stenu na západ a na východ od pozície kde práve stojí.

Túto úlohu som ešte doplnil o to, že ak použije viac ako 50 krokov, ráta sa to, ako keby posledným stúpil na políčko -1. Pri prvých testoch sa sieť naučila riešiť tento problém tak, že stále volila pohyb na sever. Spoliehala sa na to, že narazí na severnú stenu a do bodu sa $+1$ dostane vďaka náhodnému kolmému pohybu. Toto riešenie ma prekvapilo a zaujala ma jeho jednoduchosť. Bola ale veľmi neefektívna - poväčšinou túto úlohu splnila za vyše 200 ťahov (zopárkrát aj mnohonásobne viac).

Pohyb agenta v tomto bludisku som realizoval pomocou prechodovej funkcie, kde pre každú pozíciu a každý pohyb bolo povedané na ktorú pozíciu sa dostane. Pre

zložitejšie bludiská, respektíve bludiská meniace sa v čase, by bolo možno prehľadnejšie použitie dvojrozmerného poľa znázorňujúceho hraciu plochu, ale pre takto malú plochu bola prehľadná aj prechodová tabuľka.

3.3.1 Vstupy

Sieť dostávala, ako aj sieť popísaná v práci Szita et. al. [3], na vstup 6 hodnôt. Prvé dva vstupy reprezentovali výskyt steny na západ a východ od pozície agenta. Tretí až šiesty vstup reprezentoval, ktorú akciu agent vykonal v predchádzajúcom kroku. V práci Szita et. al. [3] použili túto techniku, aby sieť stabilizovali. Ja som ju použil tiež, aby som s nimi mohol porovnávať efektivitu spôsobu učenia.

Vstupný vektor bol vstupný číslom $f = 20/|\mathbf{v}_{hid}|$. Dôvod je vysvetlený v časti 3.1. Implementácia.

Takto vybrané vstupy majú za následok ťažkú pozorovateľnosť prostredia. Ak sa agent na začiatku nachádza na pozícii [1,1], [1,3] alebo [3,2] nevie to rozlíšiť. Takisto pozície [2,3] [3,3],[2,1] a [3,1] splývajú do jednej, rovnako ako pozície [4,1], [4,2] a [4,3]. V jednej jedinej pozícii a to v [1,2] si môže byť sieť istá, kde sa presne nachádza.

3.3.2 Učenie

Ako rýchlosť učenia v čase t som v tejto úlohe použil funkciu

$$\alpha_t = \alpha_{kon} + \alpha_{zac}\left(\frac{100}{100+t}\right)$$

Kde $\alpha_{zac} = 0.3$ a $\alpha_{zac} = 0.0004$. V kroku k v rámci jednej hry som pri trénovaní siete použil ako trénovaciu hodnotu d

$$d = 1 - 1.005^k$$

Ak daným krokom vstúpil agent na políčko -1, od trénovacej hodnoty d som odpočítal 1.4, ak na políčko 1, pripočítal som k trénovacej hodnote 1.

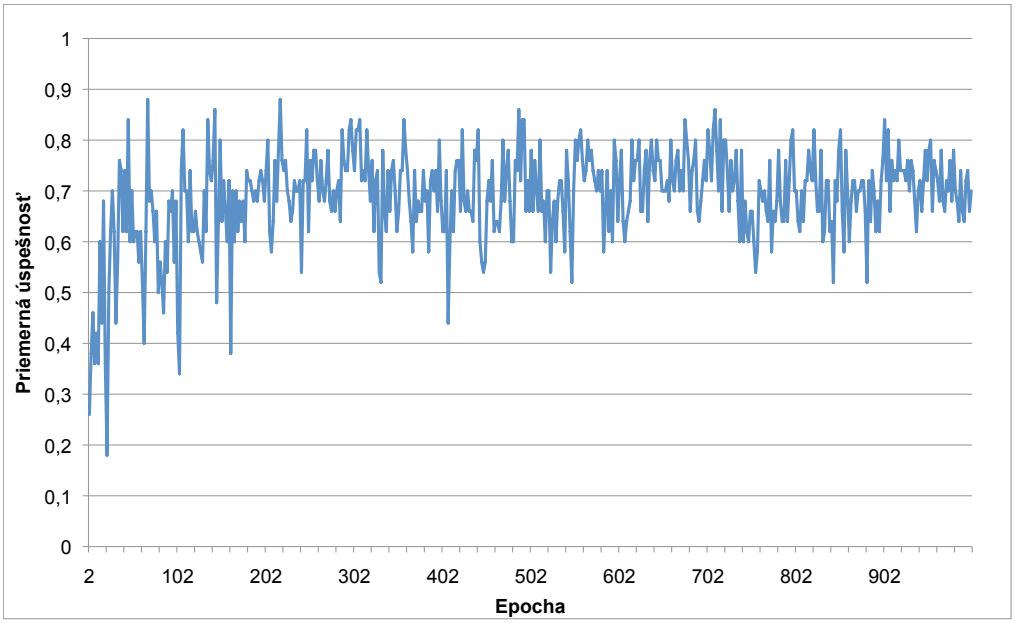
Potreboval som dosiahnuť to, aby mala sieť snahu minimalizovať počet krokov a aby víťazstvo pri použití optimálnej stratégie vedela rozlíšiť od víťazstva pri použití neefektívnej stratégie. Dôvod rozdielu vo veľkosti odmeny a trestu je popísaný v časti 3.2.2.

Hodnota 1.005 je zvolená z toho dôvodu, pretože pri maximálnom počte krokov

(50) nadobúda hodnotu asi 1.28 , ktorá nezatieni odmenu / trest za nájdenie políčka ± 1, ale je dostatočne veľká na to, aby agenta motivovala k väčšej účinnosti.

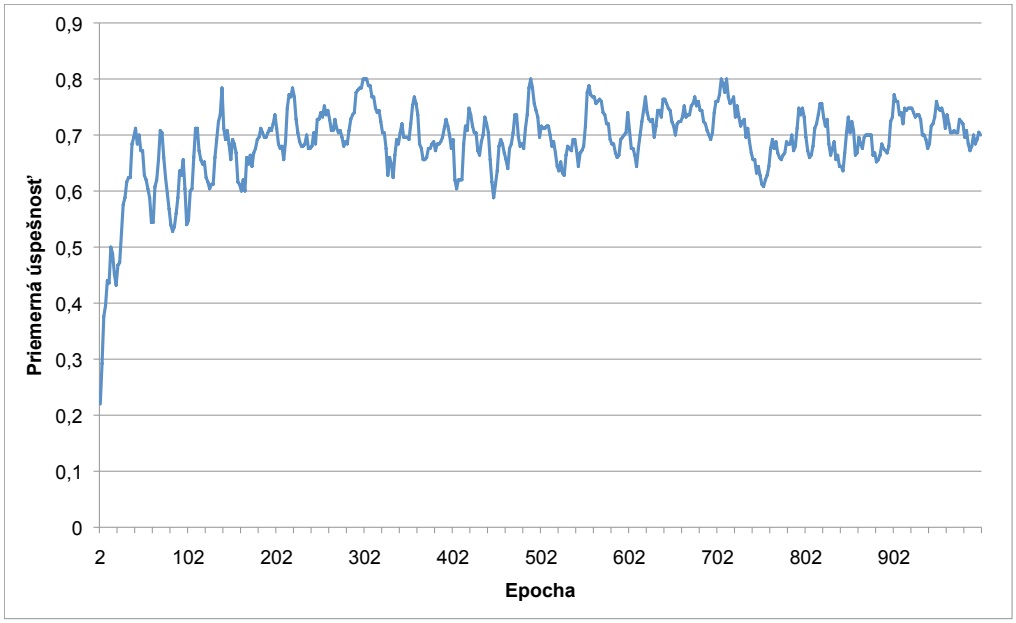
Pomocou hodnoty d som trénoval akciu pohybu, ktorú v danom kroku agent zvolil. S pravdepodobnosťou $(1 - t/4000)$ som navyše trénoval aj ostatné akcie pohybu, ale rýchlosť učenia použitá pri ich trénovaní bola $\alpha_{5(t+10)}$. K tomuto riešeniu som sa musel uchýliť, pretože mnohokrát vôbec neskúmala prostredie, ostala pri neustálom opakovaní jednej akcie, čo malo za následok, že sa nevedela naučiť riešenie. Toto spôsobilo vychýlenie z tohto stavu a sieť sa teda vedela naučiť riešenie.

3.3.3 Výsledky



Obr. 7: Výsledky úspešnosti pri experimente 2: Bludisko

Obrázok 7 vznikol pri trénovaní a zároveň testovaní. Neurónová sieť hrala vždy 2 hry, na ktorých sa učila a potom si zahrala 50 testovacích hier, ktorých úspešnosť sa zakreslila do grafu. Táto neurónová sieť mala veľkosť rezervoáru 500 a trénovala sa

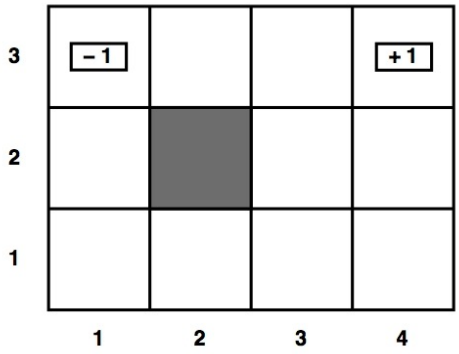


Obr. 8: Výsledky úspešnosti pri experimente 2: Bludisko - vyhladená verzia

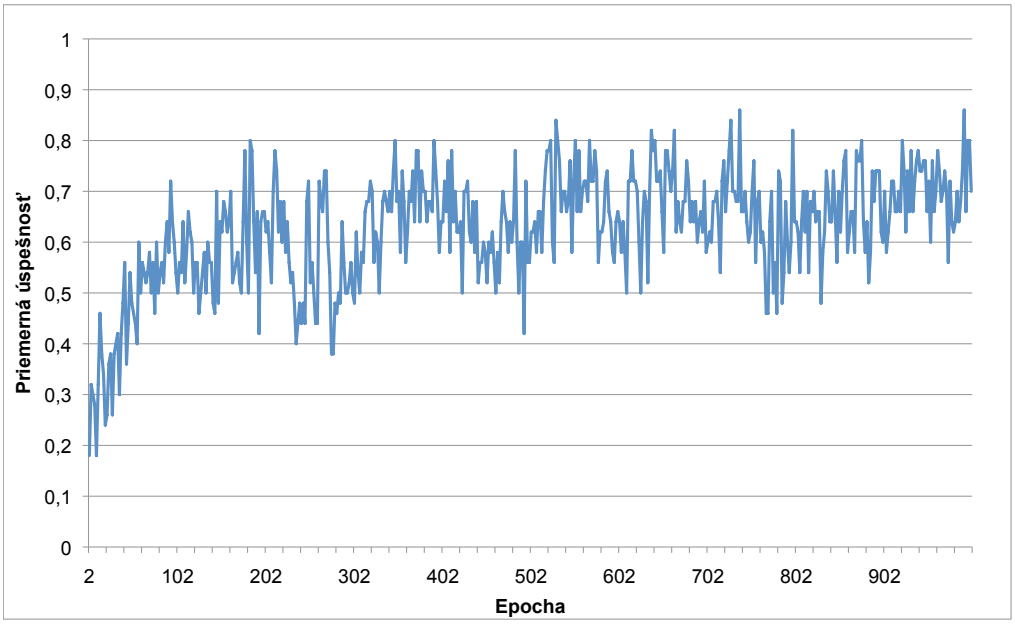
počas 1000 hier. Obrázok 8 vznikol úpravou dát použitých na tvorbu obrázka 4. V tomto grafe úspešnosť v čase t je priemer hodnôt z časov od $t-2$ po $t+2$.

V ďalšom pokuse som v bludisku zmenil pozíciu políčka -1 z pozície [4,2] na pozíciu [1,3]. Bludisko po zmene znázorňuje obrázok 9. Sieť sa trénovala rovnako ako v predchádzajúcom prípade a jej priemernú úspešnosť počas trénovania znázorňuje obrázok 10. Ako je na ňom vidieť, úspešnosť siete sa nezmenila ani pri zmenenom bludisku.

V porovnaní s prácou Szita et. al. [3] musím konštatovať, že úspešnosť ich trénovania bola maximálne 90% a po prekonaní počiatočného učenia sa mala úspešnosť medzi 80% a 85%, raz však klesla na 75%. Moja sieť sa naučila najlepšie na 88%, priemerne sa však držala po počiatočnom učení na 70%, čo je asi o $10-15\%$ horšie.



Obr. 9: Zmenené bludisko v experimente 2



Obr. 10: Výsledky úspešnosti pri experimente 2: Bludisko - zmenené bludisko

4 Záver

Cieľom mojej práce bolo ukázať účinnosť neurónových sietí s echo stavi na riešenie sekvenčných problémov v čiastočne pozorovateľnom prostredí. Po získaní teoretických poznatkov sa mi podarilo implementovať danú sieť a v rámci dvoch experimentov som ju učil hrať dve hry. Postup implementácie som taktiež podrobne popísal.

V prvom experimente som učil sieť hrať hru Tiger za dverami, popísanú v práci Szita et. al. [3]. Už pri implementácii som si všimol, že účinnosť siete závisí od veľkosti rezervoáru, ako aj na pomere veľkosti odmeny za úspech a trestu za neúspech. Pri dvoch meraniach, zameriavajúc sa na túto problematiku, som empiricky zistil, že pri malej veľkosti rezervoáru má zvyšovanie trestu negatívny účinok na priemernú úspešnosť siete, zatiaľ čo pri veľkom rezervoáre malo zvyšovanie trestu účinok pozitívny.

Ďalšími meraniami som sa snažil odpovedať na otázku, ako sa zmení priemerná úspešnosť siete od veľkosti jej rezervoára. Zistil som, že do veľkosti 70 sa tento problém nevedela naučiť, pri veľkosti 70 mala priemernú úspešnosť približne rovnakú aká bola pravdepodobnosť, že správne počuje (viď popis problému) a pri ďalšom zväčšovaní rezervoára sa úspešnosť zvyšovala. Pri veľkosti 300 mala výsledky s úspešnosťou až do 15% vyššiu, ako sieť s veľkosťou 70.

V druhom experimente som učil sieť orientovať sa v malom bludisku, kde jej cieľom bolo dostať sa k políčku $+1$ a vyhnúť sa políčku -1. Sieť sa naučila tento problém riešiť s priemernou úspešnosťou 70% a maximálnou úspešnosťou 88%. Aj pri tomto riešení som použil zmenený pomer veľkosti odmeny za úspech a trestu za neúspech. Testy zaoberajúce sa úspešnosťou pri rôznych veľkostiach rezervoáru a pomere veľkosti odmeny a trestu som, kvôli ich obrovskej náročnosti na strojový čas, nerobil.

Pomocou merania som zistil, že sieť sa dokáže naučiť na približne rovnakú úspešnosť, aj keď som zmenil pozíciu políčka -1.

Počas experimentov som mal najväčšie problémy s obrovskými nárokmi trénovania na strojový čas. Napríklad pokus v prvom experimente zaoberajúci sa úspešnosťou vzhľadom na veľkosť rezervoáru zabral asi 30 hodín pri plnom vyťažení jedného jadra môjho procesora (2.4GHz). Obe jadrá som nevyužíval, nakoľko môj počítač má slabé chladenie a už pri tomto zaťažení sa jadro prehrievalo.

Ak by som mal viac času, zopakoval by som druhý pokus prvého experimentu, ale

s omnoho menšou granulitou parametrov a v druhom experimente by som skúmal, ako sa bude meniť úspešnosť pri zmene veľkosti rezervoára a pomeru odmeny a trestu a ďalej skúmal tento jav aj na iných problémoch.

Literatúra

[1] Russell, S.J., Norvig, P.: Artificial Intelligence: a Modern Approach. Prentice-Hall, Englewood Cliffs, New Jersey (1994)

[2] Jaeger, H.: Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "Echo state network" approach. Technical Report GMD Report 159, German National Research Center for Information Technology (2002)

[3] István Szita, Viktor Gyenes, and András Lőrincz, Reinforcement Learning with Echo State Networks, In: International Conference on Artificial Neural Networks (ICANN), Proceedings, Part I. Lecture Notes in Computer Science ¡http://www.informatik.uni-trier.de/Springer 2006, pp. 830-839.

[4] Sutton, R., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA (1998)

[5] Bakker, P.B.: The State of Mind - Reinforcement Learning with Recurrent Neural Networks. PhD thesis, Universiteit Leiden (2004)

[6] http://en.wikipedia.org/wiki/Markov_decision_process

[7] http://en.wikipedia.org/wiki/Partially_observable_Markov_decision_process

[8] http://en.wikipedia.org/wiki/Q-learning

[9] http://en.wikipedia.org/wiki/SARSA