UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

Strojno učenje iz interakcije

(Machine learning from interaction)

Ime in priimek: Rok Breulj
Študijski program: Računalništvo in informatika
Mentor: doc. dr. Peter Rogelj

Koper, september 2014

Ključna dokumentacijska informacija

Ime in PRIIMEK: Rok BREULJ

Naslov zaključne naloge: Strojno učenje iz interakcije

Kraj: Koper

Leto: 2014

Število listov: 54 Število slik: 16 Število tabel: 7

Število referenc: 37

Mentor: doc. dr. Peter Rogelj

Ključne besede: strojno učenje, okrepitveno učenje, učenje na podlagi časovne razlike, nevronska mreža, hex, namizna igra

Izvleček:
Medtem ko se ljudje v vsakdanjem življenju učimo iz stika iz okolja, naprave nimajo tovrstne sposobnosti. Pod vplivom vedenjske psihologije se je razvilo področje okrepitvenega učenja, pri katerem se oblikuje skupek algoritmov, ki posnemajo način vedenja pri živalih, ko se te srečajo z novimi okoliščinami. Pričujoče delo povzema teoretično ozadje algoritmov in jih uporablja pri problemu igranja igre Hex brez znanja o njenih pravilih. S problemom se zelo dobro spopadejo tabularne metode, ki se ukvarjajo z majhnim številom stanj, medtem ko se nevronskih mreže v povezavi z okrepitvenim učenjem za reševanje problema na večjih igralnih ploščah izkažejo za manj uspešne. Robustnejše metode, ki v povezavi s posploševanjem in nelinearno funkcijsko aproksimacijo prinašajo boljša zagotovila konvergence, so še vedno predmet številnih raziskav in bodo v prihodnje zagotovo prinesle boljše rezultate.

Key words documentation

Name and SURNAME: Rok BREULJ

Title of final project paper: Machine learning from interaction

Place: Koper

Year: 2014

Number of pages: 54 Number of figures: 16 Number of tables: 7

Number of references: 37

Mentor: Assist. Prof. Peter Rogelj, PhD

Keywords: machine learning, reinforcement learning, temporal-difference, neural network, hex, board game

Abstract:
While humans interact and learn from the environment in their everyday lives, machines are not naturally capable of the feat. An area of machine learning called reinforcement learning has been inspired by behavioral psychology to create a set of algorithms to mimic the way animals modify their behavior when exposed to new circumstances. This work summarizes the theory behind these algorithms and applies them to the problem of playing the board game Hex without any knowledge of the games' rules. While tabular methods that deal with small state spaces are found to perform very well, the conjunction of neural networks with reinforcement learning to solve the problem on bigger boards provides less thrilling results. More robust methods that provide better convergence guarantees when combined with generalization and non-linear function approximation are still actively being researched and will undoubtedly give better results in the future.

Zahvala

Rad bi se zahvalil vsem, ki so mi stali ob strani pri pisanju naloge in učenju nasploh.

Hvaležen sem vseh trenutkov in izkušenj, ki so me pripeljale do današnjega dne. Brez navdiha čudovitih ljudi, deljenja znanja in priložnosti, ki sem jih bil deležen, te naloge zagotovo ne bi bilo.

Posebna zahvala gre k družini, prijateljem in sodelavcem za razumevanje, potrpežljivost in podporo. Z vašo pomočjo je bila to mala malica.

Navsezadnje bi se rad zahvalil še mentorju dr. Petru Roglju za neprimerljivo odzivnost.

Kazalo vsebine

Kazalo tabel

Kazalo slik

Seznam kratic

angl.	angleško
ti.	tako imenovan
tj.	to je
npr.	na primer
CR	pogojen odziv (angl. conditioned response)
CS	pogojena spodbuda (angl. conditioned stimulus)
CR	nepogojen odziv (angl. unconditioned response)
DP	dinamično programiranje (angl. dynamic programming)
MC	Monte Carlo
MDP	Markov proces odločanja (angl. Markov decision process)
$SARSA$	stanje, dejanje, nagrada, stanje, nagrada (state, action, reward, state, action)
TD	učenje na podlagi časovne razlike (angl. temporal difference)
US	nepogojena spodbuda (angl. unconditioned stimulus)
$A(s)$	množica možnih dejanj v stanju s
$E[.]$	predvidena vrednost (angl. expected value)
$P(s'\|s,a)$	prehodna funkcija (angl. transition function)
$Q(s,a)$	vrednostna funkcija dejanj (angl. action-value function)
R	pričakovan donos (angl. expected return)
$R(s,a,s')$	nagrajevalna funkcija (angl. reward function)
S	množica vseh stanj
T	končen čas episode
$V(s)$	vrednostna funkcija stanj (angl. state-value function)
a	dejanje (angl. action)
e	sled primernosti (angl. eligibility trace)
r	nagrada (angl. reward)
s	stanje (angl. state)
t	diskreten čas korak (angl. discrete time step)

α	velikost koraka, stopnja učenja (angl. step size, learning rate)
δ	TD napaka (angl. TD error)
ϵ	verjetnost naključnega dejanja
γ	faktor popuščanja (angl. discount factor)
λ	popuščanje sledi (angl. trace-decay)
π	politika vedenja (angl. behavior policy)

1 Uvod

Učimo se skozi naše celotno življenje. Eden izmed osnovnih načinov učenja temelji na podlagi interakcije z okoljem. V računalništvu pogosto radi odidemo po poti poizkušanja in prilagoditev, posebej ko verjamemo, da smo blizu rešitvi. Vendar pa se ni potrebno zazreti tako daleč, kot je računalništvo. Že kot otroci, ko se igramo s kockami in mahamo z rokami, nimamo izrecnega učitelja, imamo pa neposredno fizično povezavo z našo okolico. S svojim vedenjem vplivamo na okolje in naša čutila uporabljamo za pridobitev ogromne količine podatkov o vzrokih in učinkih, o posledicah dejanj in načinih, kako doseči cilje. Skozi naše življenje predstavljajo tovrstne interakcije velik vir znanja o našem okolju in o nas samih. Ko se pogovarjamo z ljudmi, se zavedamo kako se okolje odziva na naša dejanja in iščemo način kako vplivati na rezultat z našim vedenjem.

Čeprav ni ene same standardne definicije inteligence, lahko različne predlagane definicije med seboj primerjamo in hitro najdemo precejšnje podobnosti. V veliko primerih, definicije inteligence vsebuje idejo, da se mora posameznik, ki je inteligenten, znati prilagoditi okoljem, s katerimi se poprej še nikoli ni srečal, in v njih doseči cilje [30]. Za inteligentno vedenje očitno torej potrebujemo način, da ovrednotimo in razvrstimo nove položaje. Da se posameznik lahko uči in prilagaja svoje vedenje, mora znati upoštevati tudi informacije iz okolja in iz njih sklepati. Okrepitveno učenje (angl. reinforcement learning) predstavlja teorijo o učenju povečanja nagrade na voljo v okolici in tako neposredno povečanje možnosti prilagoditve in preživetja. Nekatere naloge so preveč zapletene, da bi se jih opisalo v statičnem računalniškem programu, kar je danes pogost postopek. Način za dinamično učenje in razvijanje programa je pri nekaterih nalogah torej potreben.

Praktično vse aktivnosti živali, podjetij in računalniških programov vključujejo niz dejanj za dosego cilja. Tako pri vožnji z avtomobilom na delo kot tudi med pripravo jutranje kave obstaja cilj in zaporedje dejanj za uspešno opravljen cilj. Prilagodljiv krmilni sistem, ki se zna učiti izvajati takšne sekvenčne naloge odločanja lahko najde vlogo v številnih domenah, kot so krmiljenje proizvodnega procesa, avtonomnih vozilih, letalstvu in pripomočkih za invalide. V pametnih sistemih, ki delujejo v dinamičnih okoljih resničnega sveta, kjer se ni mogoče zanašati na obvladljive pogoje in kjer vladajo negotovost ter časovne omejitve, ima lahko odločanje na podlagi okrepitvenega učenja

bistvene prednosti pred ostalimi vrstami učenja.

Področje okrepitvenega učenja sega v zelo različne discipline in je močno povezano s teorijo krmiljenja (angl. control theory), psihologijo in nevroznanostjo. Teorija krmiljenja pripomore k reševanju problema z analitičnega oziroma matematičnega vidika, medtem ko se psihologija in nevroznanost za odgovore zgledujeta po bioloških procesih. Veliko temeljnih smernic je izpeljanih iz psihologije vedenja in učenja; teorijah, ki se tičejo nagrajevanja in pogojevanja dejanj. Algoritmični pristopi so izpeljani pod podobnimi načeli kot ljudje in živali oblikujemo vedenja glede na odzive iz okolice.

Zamisel o gradnji inteligentni strojev sega v daljno preteklost; o tem so razmišljali že Egipčani. Čez leta se je razvilo veliko teorij, ampak šele z začetkom sodobnega računalnika pred 60-imi leti sta se umetna inteligenca in strojno učenje razvila v samostojno znanstveno področje [2]. Leta 1948 je Claude Elwood Shannon [4] napisal predlog za šahovski program, Arthur Samuel [1] pa je leta 1959 razvil računalniški program, ki se je naučil igrati namizno igro dama z igranjem proti samemu sebi. V zadnjih letih so se raziskave osredotočale bolj na posnemanje bioloških modelov, da be izdelale programe, ki rešujejo probleme in razmišljajo kot ljudje. Nevronske mreže (angl. neural networks), pri katerih gre za zelo poenostavljen model možganov, so bile uspešno uporabljene v vrsti aplikacijah. Po formalizaciji Samuelevega pristopa in oblikovanja učenja na podlagi časovne razlike lambda Richarda Suttona [20] je Richard Tesauro [11] leta 1992 razvil računalniškega igralca za igro Backgammon, ki je tekmoval proti najboljšim človeškim igralcem na svetu. Čeprav je Tesaurova združitev pristopa okrepitvenega učenja in nevronskih mrež pretresla področje umetne inteligence in Backgammonske skupnosti, ni bilo veliko drugih uspehov v namiznih igrah [10, 23, 25]. Prenos Tesaurove rešitve v največje namizne igre na področju umetne inteligence – šah in Go – niso uspele; rezultati so bili slabši, kot pa so jih dosegale konvencionalne metode. Poleg namiznih iger se je okrepitveno učenje uporabljalo tudi v problemih robotike, razporejanja, dinamičnih dodelitev in optimizacije [21].

V nadaljevanju naloge je v razdelku 1.1 pregledan izvor okrepitvenega učenja iz vedenjske plati. Temu sledi pogled s stališča umetne inteligence in inženirstva, razišče pa se tudi računski pristop do učenja iz interakcije. V poglavju 2 je formalno opredeljen celotni problem okrepitvenega učenja, rešitve pa so predstavljene v poglavju 3 in 4. Primerjajo se različni algoritmi, njihove lastnosti, povezave in kombinacije. Ker so cilji in pravila pri abstraktnih namiznih igrah jasno opredeljeni, s čimer se poenostavita model in simulacija, so priročno testno okolje za študijo tovrstnega učenja. Poleg tega pa predstavljajo tudi zahteven in zanimiv problem. V poglavju 5 so rešitve iz okrepitvenega učenja uporabljene v namizni igri Hex in na koncu v poglavju 6 razpravljeni rezultati skupaj s pogledom na prihodnost.

1.1 Psihologija

Okrepitveno učenje ima korenine v psihologiji učenja živali, iz kjer izvira tudi samo ime. Posebej se nanaša na klasično pogojevanje (angl. classical conditioning) in operantno pogojevaje (angl. operant conditioning).

1.1.1 Klasično pogojevanje

Klasično pogojevanje (imenovana tudi Pavlovo pogojevanje) je učenje prek povezav oz. asociacij.

V začetku 20. stoletja je ruski psiholog Ivan Pavlov (1849-1936) med preučevanjem prebavnega sistema psov odkril vedenjski fenomen [13]: psi so se začeli sliniti takoj, ko so laboratorijski tehniki, ki so jih hranili, vstopili v sobo, čeprav psi še niso dobili hrane. Pavlov je spoznal, da so se psi začeli sliniti, ker so vedeli, da bodo dobili hrano; povezali so prihod tehnikov s hranjenjem.

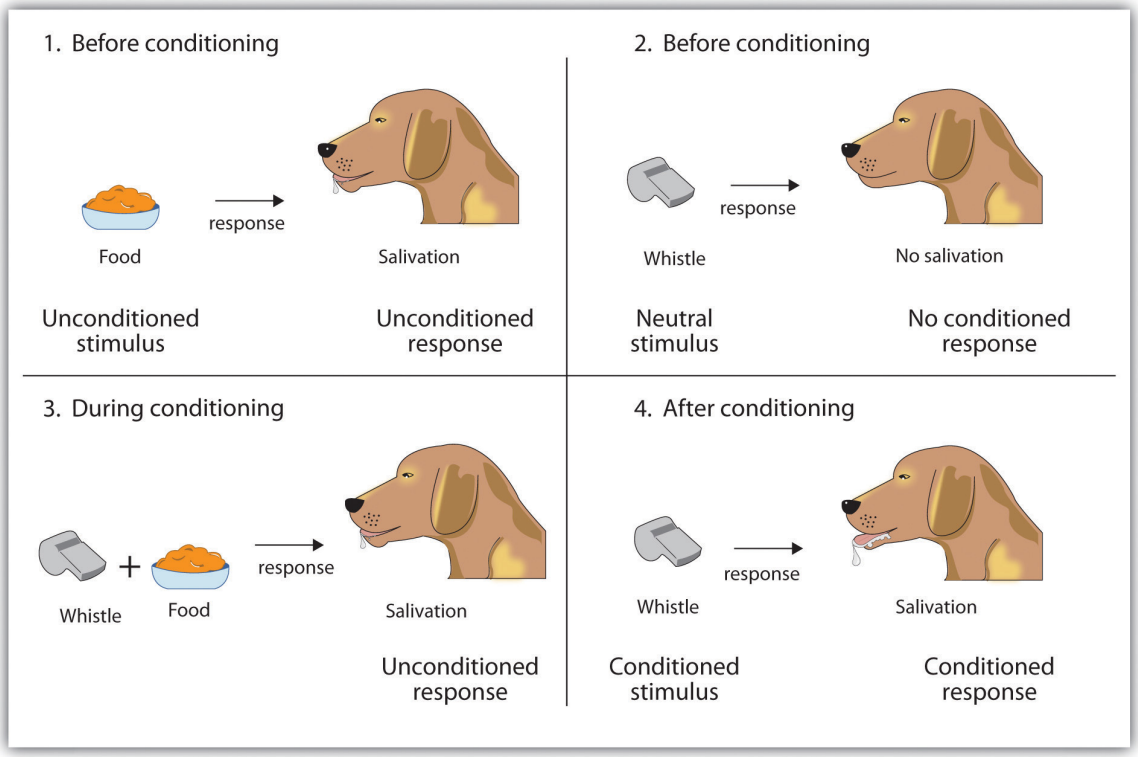
S svojo ekipo je začel proces raziskovati bolj podrobno. Opravil je vrsto eksperimentov, pri katerih so bili psi izpostavljeni zvoku, tik preden so dobili hrano. Sistematično je nadzoroval časovno razliko med pojavom zvoka in hrano ter zabeležil količino sline pri psih. Najprej so se psi slinili samo, ko so hrano videli ali zavohali. Po večkrat predstavljenem zvoku skupaj s hrano pa so se psi začeli sliniti takoj, ko so zaslišali zvok. Zvok so se namreč naučili povezati s hrano, ki mu je sledila.

Pavlov je odkril temeljni asociativni proces imenovan klasično pogojevanje. Gre za učenje, pri katerem postane nevtralna spodbuda (na primer: zvok) povezana s spodbudo, ki vedenje sproži sama po sebi (na primer: hrana). Ko se povezava enkrat nauči, poprej nevtralna spodbuda zadošča za pojav vedenja, ki je v večji meri enakovredno (Pavlov je opazil razliko v sestavi sline [17, 22, 29]).

Prihod tehnikov oz. zvok je Pavlov imenoval pogojena spodbuda (angl. conditioned stimulus CS), ker je njen učinek odvisen od povezave s hrano. Hrano je imenoval nepogojena spodbuda (angl. unconditioned stimulus US), ker njen učinek ni odvisen od predhodnih izkušenj. Podobno gre pri pogojenem odzivu (angl. conditioned response CR) za odziv pogojene spodbude CS in pri nepogojenem odzivu (angl. unconditioned response UR) za odziv nepogojene spodbude US. Pavlov je odkril, da je krajši razmak med zvokom in prikazom hrane povzročil močnejše in hitrejše učenje pogojenega odziva CR psa [34].

Pogojevanje je evolucijsko koristno, ker omogoča organizmom razviti pričakovanja, ki jim pomagajo v dobrih in slabih okoliščinah. Razvidno je na primeru živali, ki zavoha novo hrano, jo poje in posledično zboli. Če se žival zna naučiti povezave vonja (CS) s hrano (US), se bo znala izogibati določeni hrani že po vonju.

Klasično pogojevanje obravnava samo problem napovedovanja, ker odziv živali ne



Slika 1: Klasično pogojevanje piščalke namesto hrane za slinjenje pri psu [5].

vpliva na eksperiment, oziroma na splošno ne vpliva na okolje. Učenje na podlagi časovne razlike (angl. temporal difference learning), ki je opisano pozneje v razdelku 3.3, je bilo prvotno povezano predvsem s klasičnim pogojevanjem in problemom napovedovanja, kjer pogojena spodbuda (CS), ki je vezana na poznejšo nepogojeno spodbudo (US), povzroči potrebo po ovrednotenju časovne razlike vrednostne funkcije. Cilj izračuna je zagotoviti, da postane pogojena spodbuda (CS) po učenju napovednik nepogojene spodbude (US). Osnutek na temo algoritmičnih pristopov do eksperimentov klasičnega pogojevanja sta sestavila Belkenius in Morén [3].

Čeprav je bilo učenje na podlagi časovne razlike sprva namenjeno reševanju problema napovedovanja, se uporablja tudi za reševanje problema optimalnega krmiljenja (glej razdelek 3.3) [21].

1.1.2 Operantno pogojevanje

V klasičnem pogojevanju se organizem nauči povezati nove spodbude z naravnim biološkim odzivom, kot je slinjenje ali strah. Organizem sam se ne nauči ničesar novega, ampak se v prisotnosti novega signala začne vesti na že obstoječi način. Po drugi strani pa gre pri operantnem pogojevanju za učenje, ki se zgodi glede na posledice vedenja in lahko vsebuje nova dejanja. Med operantno pogojevanje sodi primer, ko se pes na

ukaz vsede, ker je v preteklosti za to dejanje dobil pohvalo. Za operantno pogojevanje gre tudi, ko nasilnež v šoli grozi sošolcem, ker lahko tako doseže svoje cilje, ali ko otrok domov prinese dobre ocene, ker so mu starši v nasprotnem primeru zagrozili s kaznijo. Pri operantnem pogojevanju se organizem uči iz posledic svojih dejanj.

Psiholog Edward L. Thorndike (1874-1949) je bil prvi, ki je sistematično preučil operantno pogojevanje. Izdelal je škatlo, katero je bilo mogoče odpreti samo po rešitvi preproste uganke. Vanjo je spustil mačko in opazoval njeno vedenje. Sprva so mačke praskale in grizle naključno, sčasoma pa so slučajno potisnile na ročico in odprle vrata, za katerimi je stala nagrada – ostanki ribe. Ko je bila mačka naslednjič zaprta v škatlo, je poizkusila manjše število neučinkovitih dejanj, preden se je uspešno osvobodila. Po več poizkusih se je mačka naučila skoraj takoj pravilno odzvati. [8]

Opazovanje tovrstnih sprememb v mačjem vedenju je Thorndikeju pomagalo razviti njegov zakon o učinku, pri katerem gre za princip, da se odzivi, ki v določeni situaciji navadno pripeljejo do prijetnega izida, bolj verjetno pojavijo ponovno v podobni situaciji, medtem ko je za odzive, ki tipično pripeljejo do neprijetnega izida, manj verjetno, da se ponovno pojavijo v tej situaciji. [9]

Vedenjski psiholog B. F. Skinner (1904-1990) je omenjene ideje razširil in jih povezal v bolj dovršen sistem, ki opredeljuje operantno pogojevanje. Zasnoval je operantne komore (tako imenovane Skinner škatle) za sistemično preučevanje učenja, pri katerih gre za zaprto strukturo z dovolj prostora za manjšo žival, v kateri je slednja s pritiskom na palico ali gumb prišla do nagrade v obliki vode ali hrane. Struktura je poleg tega vsebovala tudi napravo za grafični zapis odzivov živali. [5]

Najosnovnejši eksperiment je Skinner izvedel na način, ki je zelo podoben Thorndikejevem poizkusu z mačkami. V škatlo je spustil podgano, katera se je sprva odzvala po pričakovanjih - hitela je naokrog, vohljala in praskala po tleh ter stenah. Čez čas je slučajno naletela na gumb in ga pritisnila ter s tem prišla do koščka hrane. Naslednjič je že potrebovala manj časa in z vsakim novim poizkusom je hitreje pritisnila na gumb. Kmalu je pritiskala na gumb, čim je lahko jedla hrano. Kot pravi zakon o učinku, se je podgana naučila ponavljati dejanje, ki ji je pomagalo priti do hrane, in prenehala z dejanjem, ki so se bila izkazala za neuspešna. [5]

Skinner je preučeval, kako živali spreminjajo svoje vedenje v odvisnosti od okrepitve (angl. reinforcement) in kaznovanja (angl. punishment). Določil je izraze, ki razlagajo proces operantnega učenja (glej tabelo 1). Okrepitev je opredelil kot dogodek, ki utrdi ali zviša verjetnost nekega vedenja, kaznovanje pa je označil za dogodek, ki oslabi ali zniža verjetnost nekega vedenja. Uporabil je še izraza pozitivno in negativno za opredeliti, če je spodbuda predstavljena ali odvzeta. Pozitivna okrepitev torej utrdi odziv s predstavitvijo nečesa prijetnega in negativna okrepitev utrdi odziv z znižanjem ali odvzemom nečesa neprijetnega. Pohvala otroka za opravljeno domačo

nalogo tako na primer sodi v pozitivno okrepitev, medtem ko predstavlja jemanje aspirina za zniževanje glavobola negativno okrepitev. V obeh primerih okrepitev zviša verjetnost, da se vedenje v prihodnosti ponovi. [5]

Tabela 1: Vpliv pozitivne in negativne okrepitve in kaznovanja na vedenje.

Izraz	**Opis**	**Izid**	**Primer**
Pozitivna okrepitev	Predstavljena ali povečana prijetna spodbuda	Vedenje je utrjeno	Otrok dobi slaščico, potem ko pospravi sobo
Negativna okrepitev	Zmanjšana ali odvzeta neprijetna spodbuda	Vedenje je utrjeno	Starši se prenehajo pritoževati, potem ko ko otrok pospravi sobo
Pozitivno kaznovanje	Predstavljena ali povečana neprijetna spodbuda	Vedenje je oslabljeno	Učenec dobi dodatno domačo nalogo, potem ko nagaja v razredu
Negativno kaznovanje	Zmanjšana ali odvzeta prijetna spodbuda	Vedenje je oslabljeno	Otrok izgubi privilegij računalnika, potem ko pride pozno domov

Čeprav je razlika med okrepitvijo (povišanje verjetnosti vedenja) in kaznovanjem (znižanje verjetnosti vedenja) navadno jasna, je v nekaterih primerih težko določiti, ali gre za pozitivno ali negativno. V vročem poletnem dnevu je lahko svež veter zaznan kot pozitivna okrepitev (ker prinese hladnejši zrak) ali pa negativna okrepitev (ker odvzame vroč zrak). V nekaterih primerih je lahko okrepitev hkrati pozitivna in negativna. Za odvisnika, jemanje drog hkrati prinese užitek (pozitivna okrepitev) in odstrani neprijetne simptome umika (negativna okrepitev).

Pomembno se je tudi zavedati, da okrepitev in kaznovanje nista zgolj nasprotna pojma. Spreminjanje vedenja s pomočjo pozitivne okrepitve je skoraj vedno učinkovitejše od kaznovanja, ker pozitivna okrepitev pri osebi ali živali izboljša razpoloženje in pripomore k vzpostavitvi pozitivnega razmerja z osebo, ki predstavlja okrepitev. Med vrste pozitivne okrepitve, ki so učinkovite v vsakdanjem življenju, sodijo izrečene pohvale in odobritve, podelitev statusa in prestiža ter neposredno denarno izplačilo. Pri kaznovanju pa je po drugi strani bolj verjetno, da bodo spremembe vedenja samo začasne, ker temelji na prisili in vzpostavi negativno ter nasprotujoče razmerje z osebo, ki predstavlja kazen. Ko se oseba, ki kazen predstavi, iz okolja umakne, se neželeno vedenje najverjetneje vrne. [5]

Operantno pogojevanje je metoda učenja, ki se uporablja pri treniranju živali za izvajanje različnih trikov. V filmih in na predstavah so živali, od psov do konjev in

delfinov, naučene dejanj z uporabo pozitivnih okrepitev – skačejo čez ovire, se vrtijo, pomagajo osebi pri vsakdanjih opravilih in izvajajo podobna zanje neobičajna dejanja. [5]

Velikokrat se pri učenju hkrati izvajata klasično in operantno pogojevanje. Učitelji imajo s seboj napravo, ki proizvede določen zvok. Učenje se začne z nagrajevanjem želenega enostavnega dejanja s hrano (operantno pogojevanje) in hkrati z vzpostavljanjem povezave med hrano in zvokom (klasično pogojevanje). Hrana se tako lahko po intervalih izpusti in pred dejanjem se doda še zvočni ukaz, na katerega želimo vezati učeno dejanje (klasično pogojevanje). Tako z nagrado hrane povežemo samo zvočni ukaz, ki je predstavljen pred dejanjem. Kompleksnejša dejanja se naučijo postopoma iz enostavnejših z nadaljnjo povezavo spodbud, kar se imenovuje proces oblikovanja. [5]

Tudi domači ljubljenčki se obnašanja naučijo na podlagi teh konceptov; ne samo na ukaz, ampak tudi samega vedenja na povodcu, do tujcev itd. S to metodo se živali lahko nauči celo razlikovati med podobnimi vzorci, s čimer lahko znanstveniki na živalih preverjajo sposobnost učenja. Porter in Neuringer [7] sta golobe na primer naučila, da so razlikovali med različnimi vrstami glasbe, Watanabe, Sakamoto in Wakita [33] pa so jih naučili razlikovati med različnimi stili umetnosti.

Operantno pogojevanje se od klasičnega razlikuje v tem, da spremeni vedenje do okolja. Ne obravnava več samo problema napovedovanja, ampak širši problem krmiljenja.

2 Problem okrepitvenega učenja

Okrepitveno učenje (angl. reinforcement learning) po [21] je učenje, kaj narediti oziroma kako izbirati dejanje, da povečamo številčni nagrajevalni signal. Učencu niso nikoli predstavljena pravilna ali optimalna dejanja, kot pri večini oblik strojnega učenja. Katera dejanja prinesejo največjo nagrado mora sam odkriti s poizkušanjem. Skozi interakcijo z okoljem se uči posledic svojih dejanj. V najbolj zanimivih in težavnih primerih imajo dejanja vpliv ne le na takojšnjo nagrado ampak tudi na naslednji položaj in posledične nagrade. Te dve značilnosti, iskanje s poizkušanjem in zakasnjene nagrade, so dve najpomembnejši lastnosti okrepitvenega učenja.

Okrepitveno učenje se od nadzorovanega učenja (angl. supervised learning) razlikuje v tem, da nima izobraženega zunanjega nadzornika, ki učencu predloži primere in rezultate. Nadzorovano učenje je pomemben tip učenja, vendar pa ni primerno za učenje iz interakcije. V interaktivnih problemih je velikokrat nepraktično pridobiti primere želenega vedenja, ki so pravilni in hkrati predstavljajo vsa stanja v katerih mora učenec delovati. V neznanem okolju, kjer bi si lahko predstavljali, da je učenje najbolj koristno, se mora učenec učiti iz svojih izkušenj.

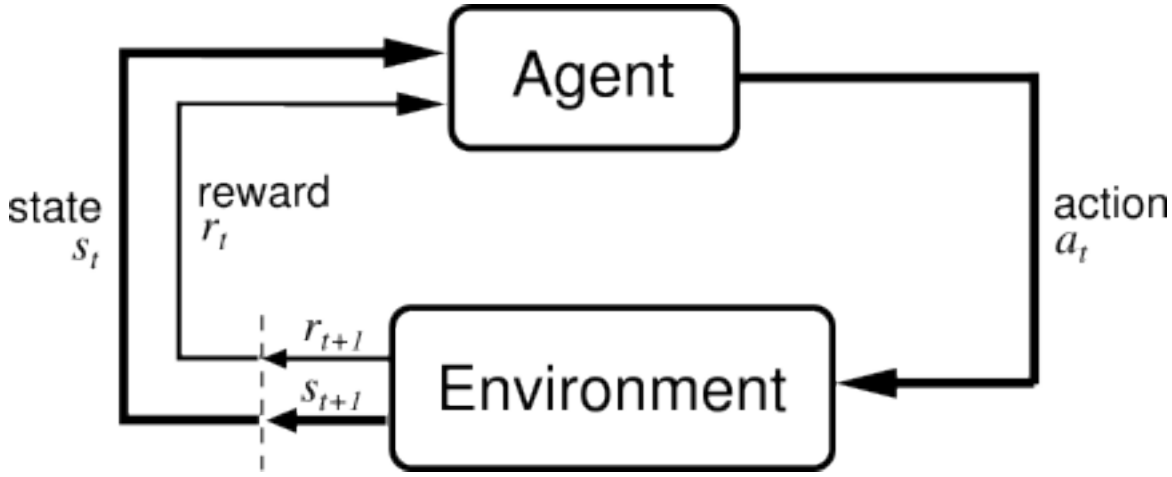
To poglavje formalno definira dele okrepitvenega učenja ter določi predpostavke, ki so potrebne za opis rešitev v sledečih poglavjih.

2.1 Elementi okrepitvenega učenja

Cilj algoritmov okrepitvenega učenja je optimizirati vedenje *učenca (angl. agent)*. Učenec je tisti, ki se skozi interakcijo odloča o *dejanjih (angl. action)* za rešitev zadane *naloge (angl. task)*.

Učenec se s svojimi dejanji vede na *okolje (angl. environment)*. Česar učenec ni zmožen poljubno spremeniti se smatra kot izven učenca in pripada okolju. Učenec in okolje neprestano vplivata drug na drugega; učenec izbira dejanja in okolje se nanje odziva s predstavitvijo novih *stanj (angl. state)* učencu. Okolje ob prehodih na nova stanja oddaja tudi številčne *nagrade (angl. reward)*, katere učenec skuša povečati skozi čas. Natančneje, učenec in okolje sta v interakciji v vsakem koraku diskretnega zaporedja časa $t = 0, 1, 2, 3, \ldots$. V vsakem koraku učenec prejme predstavitev stanja okolja, $s_t \in S$, kjer je S množica vseh možnih stanj. Glede na stanje se odloči za

dejanje, $a_t \in A(s)$, iz množice možnih dejanj. En časovni korak pozneje prejme učenec kot posledico svojega dejanja številčno nagrado, $r_{t+1} \in R$, in se znajde v novem stanju, s_{t+1}. Slika 2 prikazuje opisan potek interakcije med učencem in okoljem. Tovrstna opredelitev elementov okrepitvenega učenja ustreza številnim težavam. Ni nujno, da časovni koraki predstavljajo fiksne intervale v resničnem času, lahko se nanašajo na poljubne zaporedne faze odločanja oziroma delovanja. Dejanja so lahko na zelo nizkem nivoju, kot na primer napetosti, ki krmilijo roko robota, ali pa na visokem nivoju, ko gre na primer za izbiranje, v katero šolo se vpisati ali pa kakšno hrano pripraviti za večerjo. Stanja so lahko tudi v zelo različnih predstavitvah, od nizkonivojskih odčitkov senzorjev do abstraktnih simboličnih opisov sob. Na splošno lahko med dejanja sodijo katerekoli odločitve, o katerih se želimo učiti, med stanja pa vse, kar nam lahko pomaga pri teh odločitvah. Edini cilj učenca je, da poveča prejete nagrade čez čas.



Slika 2: Interakcija med učencem in njegovim okoljem [21].

Politiko (angl. policy) π imenujemo pravilo po katerem se učenec odloča katero dejanje izvesti v vsakem od stanj. Z drugimi besedami: politika preslikuje stanja v dejanja. Sama po sebi zadostuje za popolno definirano vedenje. V času t predstavlja $\pi_t(a_t|s_t)$ verjetnost, da učenec izvede dejanje a_t v stanju s_t. V psihologiji koncept politike ustreza povezavam spodbud z odzivi. V splošnem so politike lahko stohastične.

Nagrajevalna funkcija (angl. reward function) opredeljuje cilje v nalogi okrepitvenega učenja, saj predstavlja večanje nagrad čez čas edini učenčev cilj. V grobem, nagrajevalna funkcija stanja okolja preslikuje v realno število, r_t, nagrado, ki predstavlja trenutno zaželenost stanja. Pozitivne nagrade spodbujajo obiske stanj, negativne pa jih odvračajo. Oddane nagrade kažejo, kako dobro se učenec vede v okolju; nagrade opredeljujejo dobre in slabe dogodke. Uporaba nagrajevalnega signala za formalizacijo ideje cilja je ena izmed najbolj značilnih lastnosti okrepitvenega učenja. Čeprav se tovrstno oblikovanje ciljev sprva morda zdi omejujoče, se je v praksi izkazalo za zelo

fleksibilno in primerno. Številne nagrade so pogosto enostavno definirane kot $+1$ ali -1. Bistveno je, da nagrade točno določajo zadan cilj. Nagrajevalni signal ni primerno mesto za podeljevanje učenca s predhodnim znanjem kako doseči cilj. Učenca se pri igri šaha praviloma nagradi samo v primeru zmage, ne pa za dosego podciljev, kot je zavzemanje nasprotnikovih figur. Če nagrajujemo tovrstne podcilje, se zna zgoditi, da učenec najde pot, kako doseči podcilje, ne da bi dosegel končen cilj. V primeru šaha, bi lahko našel način zavzemanja nasprotnikovih figur tudi na račun izgubljene igre. Nagrajevalna funkcija je način komuniciranja z robotom kaj naj doseže, ne pa o samem načinu, kako naj to doseže. V biološkem sistemu se nagrade lahko opredelijo kot užitek ali bolečina. V psihologiji so to okrepitve ali kaznovanje. Nagrajevalna funkcija mora obvezno biti del okolja in učenec je ne sme biti zmožen spreminjati. Na splošno so nagrajevalne funkcije lahko stohastične.

Medtem ko nagrajevalna funkcija označuje kaj je dobro v takojšnjem smislu, *vrednostna funkcija (angl. value function)* V določa kaj je dobro na dolgi rok. Vrednostna funkcija izraža tako takojšnjo kot dolgoročno nagrado, ki se lahko pričakuje iz obiska stanja, in sicer izraža skupno količino nagrade, za katero lahko učenec predvidi, da jo bo čez čas prejel z začetkom v določenem stanju. Vrednosti upoštevajo stanja, ki si najverjetneje sledijo, in nagrade teh stanj. Vrednostna funkcija V preslikuje stanja s v vrednosti v. Najpomembnejši del skoraj vseh algoritmov okrepitvenega učenja je učinkovito ocenjevanje vrednosti. Tudi v vsakdanjem življenju velikokrat ocenjujemo in napovedujemo dolgoročno vrednost situacij, kar nam predstavlja nemajhen izziv. Stanja z visoko takojšnjo nagrado so lahko dolgoročno slaba in imajo nižjo vrednost kot alternativna. Slaščica ima na primer kratkoročen užitek, ampak pogosto ni najboljša izbira hrane kar se tiče našega telesnega zdravja. Veljati zna tudi obratno, in sicer ima stanje lahko zelo nizko takojšnjo nagrado, ampak se sčasoma izkaže za najboljšo izbiro. Naša pot do službe je lahko hitrejša, če ne izberemo najkrajše poti po dolžini, ampak upoštevamo promet do katerega nas zadana pot pripelje. Tudi živali se pri operantnem učenju učijo vrednosti in ne le takojšnjih nagrad. Hitro se naučijo, da na dolgi rok nagrade prejmejo hitreje, če se pravilno vedejo, kot pa, če se ne. Medtem ko so nagrade primarne, so vrednosti sekundarne. Brez nagrad ne bi bilo vrednosti, a vedemo se glede na ocene vrednosti. Razlika je tudi v tem, da nagrade prihajajo iz okolja, vrednosti pa moramo neprestano ocenjevati učenci sami. Vrednostna funkcija določa politiko vedenja, saj želimo s slednjim povečati nagrade, katere opisuje funkcija opisuje.

Vrednostna funkcija dejanj (angl. action-value function) Q je enakovredna vrednostni funkciji z razliko, da stanja s slika neposredno v dejanja a. Vrednostna funckija V določa vrednost v stanja s ($V(s) = v$), medtem ko vrednostna funkcija dejanj določa vrednost v dejanja a ($Q(a) = v$).

Nekateri algoritmi znajo uporabiti tudi model okolja, pri drugih je pa opuščen.

2.2 Končni Markov proces odločanja

V okrepitvenem učenju se učenec vede na podlagi signala iz okolja, ki ga imenujemo tudi stanje okolja. V tem razdelku je opisano kaj se zahteva od signala stanja in kakšne informacije je smiselno pričakovati od signala.

Po eni strani lahko signal stanja pove veliko več, kot samo trenutne meritve. Stanja so lahko predstavljena z močno obdelanimi originalnimi meritvami ali pa s zapletenimi strukturami, ki so zgrajene skozi čas. Ko na primer slišimo odgovor "da", se znajdemo v zelo različnih stanjih odvisno od predhodnega vprašanja, ki ga ne slišimo več.

Po drugi strani pa ne smemo predpostavljati, da nam signal stanja zna povedati vse o okolju, ali celo vse kar nam bi prišlo prav za odločanje. Če igramo igro s kartami ne smemo predvidevati, da bomo izvedeli kaj imajo drugi igralci v rokah ali pa katera je naslednja karta na vrhu kupa. Če učenec odgovori na telefon, ne smemo predpostavljati, da ve kdo ga kliče vnaprej. V obeh primerih obstajajo skrite informacije stanja, ki jih učenec ne more vedeti, ker jih ni nikoli prejel.

Idealno si želimo signal stanja, ki povzame vse uporabne predhodne informacije. Za to je navadno potrebna več kot samo trenutna informacija, ampak nikoli več kot celotna preteklost vseh prejetih informacij. Za signal stanja, ki zadrži vse uporabne predhodne informacije pravimo, da je *Markov* oziroma, da ima *Markovo lastnost*. Na primer, pri igri štiri v vrsto je trenutna konfiguracija vseh polj Markovo stanje, ker povzame vse pomembne informacije o poteku igre. Čeprav je veliko informacije o poteku igre izgubljene, je vse pomembno še vedno na voljo.

Pri končnem številu stanj in nagrad, je v splošnem dinamika okolja definirana samo s popolno porazdelitvijo verjetnosti

$$Pr\{r_{t+1} = r, s_{t+1} = s' | s_0, a_0, r_1, \ldots, s_{t-1}, a_{t-1}, r_t, s_t, a_t\} \tag{2.1}$$

na odziv okolja v času $t+1$, na dejanje v času t in za vse vrednosti r, s' in prejšnjih dogodkov $s_0, a_0, r_1, \ldots, s_{t-1}, a_{t-1}, r_t, s_t, a_t$. Če ima signal stanja *Markovo lastnost*, pa je odziv okolja v času $t+1$ odvisen samo od stanja in dejanja v času t in lahko dinamiko okolja definiramo z določitvijo le porazdelitve verjetnosti

$$Pr\{r_{t+1} = r, s_{t+1} = s' | s_t, a_t\}, \tag{2.2}$$

za vse r, s', s_t in a_t. Povedano drugače, signal stanja ima *Markovo lastnost* in je *Markovo stanje*, če in samo če je (2.1) enako (2.2) za vse s', r, in preteklosti $s_0, a_0, r_1, \ldots, s_{t-1}, a_{t-1}, r_t, s_t, a_t$. V tem primeru pravimo, da ima celotno okolje in naloga Markovo lastnost.

Če ima okolje Markovo lastnost, lahko iz enostavne dinamike predhodnega stanja (2.2) napovedujemo naslednje stanje in naslednjo nagrado za trenutno stanje in dejanje. V tem okolju lahko napovedujemo vsa stanja in nagrade v prihodnosti enako dobro kot bi lahko s popolno preteklostjo do trenutnega časa. Sledi enako, da je najboljša politika izbire dejanj v Markovem stanju enako dobra kot najboljša politika izbire dejanj s trenutnim stanjem in popolno zgodovino.

Čeprav Markova lastnost velikokrat ne drži popolnoma v nalogah okrepitvenega učenja, je vseeno zelo primerno razmišljati o stanju v okrepitvenem učenju kot približnemu Markovemu stanju, saj name omogoča nam razmišljati o odločitvah in vrednostih na podlagi trenutnega stanja.

Naloga okrepitvenega učenja, ki izpolnjuje Markovo lastnost, se imenuje *Markov proces odločanja (angl. Markov decision process – MDP)*. Če gre pri prostoru stanj in dejanj za končen prostor, temu pravimo *končen Markov proces odločanja (angl. finite MDP)*.

Končen MDP je torej popolnoma definiran s:

- končno množico dosegljivih stanj S,
- končno množico izvedljivih dejanj A,
- prehodno funkcijo, definirano na vseh stanjih iz S in za vsa dejanja iz A, ki je za prehod v stanje $s' \in S$ odvisna samo od trenutnega stanja $s \in S$ in dejanja $a \in A$:
$$P(s'|s,a) = Pr\{s_{t+1} = s'|s_t = s, a_t = a\}, \tag{2.3}$$
- nagrajevalno funkcijo, ki je, posledično od prehodne funkcije, tudi odvisna samo od trenutnega stanja in trenutnega dejanja:
$$R(s,a,s') = E[r_{t+1}|s_t = s, a_t = a, s_{t+1} = s']. \tag{2.4}$$

2.3 Diskretno okrepitveno učenje

Ta razdelek povzema okrepitveno učenje v diskretnem primeru, v katerem je prostor stanj okolja diskreten in končen, čas pa je razdeljen v diskretne korake.

Politika π slika stanje v dejanje, kot je omenjeno v razdelku 2.1. Za končne MDP lahko definiramo tudi *optimalno politiko (angl. optimal policy)* $\pi^\star$. Naj bosta π in $\pi^\star$ politiki in V_π vrednostna funkcija politike π ter $V_{\pi^\star}$ vrednostna funkcija politike $\pi^\star$. Politika $\pi^\star$ je optimalna, če ima vrednostno funkcijo $V_{\pi^\star}$ z naslednjo lastnostjo

$$V_{\pi^\star}(s) >= V_\pi(s), \forall s, \tag{2.5}$$

za vse možne politike π.

Kar je bilo do sedaj navedeno kot pričakovana dolgoročna nagrada, se pogosto imenuje *pričakovan donos (angl. expected return)*. Formalna definicija donosa nekega stanja v času t za poslednje nagrade $r_{t+1}, r_{t+2}, r_{t+3} \dots$ je

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}, \tag{2.6}$$

kjer je $0 \leq \gamma \leq 1$. Donos je vsota vseh nadaljnjih nagrad, ki jih pričakujemo po času t do končnega stanja v času T. V končnem stanju definiramo, da je prehod možen samo v isto stanje in nagrada ob prehodu vedno ničelna. S tem lahko poenotimo *epizodične* in *neskončne naloge* z uvedbo konstante γ, ki predstavlja *faktor popuščanja (angl. discount factor)*, s tem da je lahko $T = \infty$ ali $\gamma = 1$, ampak ne oboje hkrati. Pri neskončnih nalogah, ki jih ne moremo razdeliti na epizode, je $T = \infty$, saj se nikoli ne končajo, hkrati pa mora biti $\gamma < 1$, drugače lahko donos postane neskončen. Faktor popuščanja določa, koliko želimo upoštevati prihodnje nagrade. Z $\gamma = 0$ se osredotoči samo za trenutno nagrado. Pri epizodičnih nalogah, kot so igre, je navadno $\gamma = 1$.

Za končne MDP lahko *vrednost stanja s po politiki π, ti. vrednostno funkcijo $V_\pi(s)$*, definiramo kot pričakovan donos iz stanja s z nadaljnjim upoštevanjem politike π, formalno:

$$V_\pi(s) = E_\pi[R_t | s_t = s] = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s \right], \tag{2.7}$$

kjer $E_\pi[.]$ predstavlja pričakovano vrednostjo, če učenec sledi politiki π.

Podobno lahko *vrednost dejanja a v stanju s* po politiki π, ti. vrednostno funkcijo dejanj $Q_\pi(s, a)$, definiramo kot pričakovan donos iz stanja s ob dejanju a in nadaljnjim upoštevanjem politike π, formalno:

$$Q_\pi(s, a) = E_\pi[R_t | s_t = s, a_t = a] = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a \right]. \tag{2.8}$$

2.4 Raziskovanje in izkoriščanje

Eden od izzivov okrepitvenega učenja, ki jih ne najdemo v ostalih oblikah strojnega učenja, je kompromis med raziskovanjem (angl. exploration) in izkoriščanjem (angl. exploitation). Med učenjem, ko učenec uporablja približek optimalne vrednostne funkcije za svoje vedenje, mu to omogoča pridobiti največjo znano nagrado, ampak nikjer ni zagotovil, da je ta znana politika tudi v splošnem najboljša. Boljša rešitev bi mogoče lahko bila najdena, če bi učenec imel dovoljenje raziskovati dejanja, ki jih še ni poizkusil.

Spodaj opisana ϵ-požrešna izbira dejanj je zelo preprosta in se v praksi pogosto uporablja. Obstaja še veliko drugih metod, nekaterih bolj kompleksnih od drugih. Več primerov je v delu Thrun [27] ter Sutton in Barto [21].

2.4.1 ϵ-požrešna izbira dejanj

Eden izmed najbolj enostavnih pristopov k izbiri dejanja za ravnovesje med raziskovanjem in izkoriščanjem je uvod parametra ϵ, ki določi verjetnost izbire naključnega dejanja. Po tej metodi učenec ob vsakem koraku izbere naključno dejanje z verjetnostjo ϵ in požrešno dejanje z verjetnostjo $1 - \epsilon$.

Velikokrat je koristno izbrati veliko naključnih dejanj ob začetku učenja in nato, ko učenje napreduje, znižati pogostost naključnih dejanj. S tem v fazi največjega učenja čimbolj raziščemo prostor stanj. Znižanje vrednosti ϵ logično zniža stopnjo raziskovanja in zviša stopnjo izkoriščanja. Težava pri *ϵ-požrešni metodi izbiranja dejanj (angl. ϵ-greedy action selection*) je v tem, da ne obstaja preprost način za izbiro vrednosti ϵ. V veliko primerih je težko izbrati, kdaj povečati ali znižati število naključnih dejanj, ki naj jih učenec izbere.

3 Tabularne rešitve

V tem poglavju so opisani ključni algoritmi okrepitvenega učenja v njihovih enostavnih oblikah – ko je prostor stanj in dejanj dovolj majhen za predstavitev ocenjene vrednostne funkcije s poljem ali tabelo. V teh primerih znajo algoritmi najti optimalno vrednostno funkcijo in optimalno politiko vedenja. To je v nasprotju z aproksimacijskimi metodami opisanimi v naslednjem poglavju, katere znajo najti le približne rešitve, ampak jih lahko učinkovito uporabljamo na veliko večjih problemih.

3.1 Dinamično programiranje

Richard Bellman je avtor *dinamičnega programiranja (angl. dynamic programming – DP)*, izraza, ki se nanaša na zbirko algoritmov, katere lahko uporabimo za izračunati optimalno politiko za MDP. Algoritmi dinamičnega programiranja za delovanje potrebujejo popoln in natančen model okolja. Vsi se nanašajo na Bellmanovo enačbo [18]

$$\begin{aligned} V_\pi(s) &= E_\pi[R_t|s_t = s] \\ &= E_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\middle| s_t = s\right] \\ &= E_\pi\left[r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}\middle| s_t = s\right] \\ &= \sum_a \pi(a|s)\sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma E_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}\middle| s_{t+1} = s'\right]\right] \\ &= \sum_a \pi(a|s)\sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V_\pi(s')\right]. \end{aligned} \tag{3.1}$$

Bellmanova enačba pravi, da je možno izračunati vrednost vseh stanj s, torej $V(s)$ z rekurzivnimi koraki čez vsa stanja, ki so *povezana* s stanjem s. Izraz povezana je tukaj definiran na naslednji način: dve stanji s in s' sta povezani, če in samo če je $P(s'|s,a) > 0$ pri vseh $a \in A$. Postopek se ponovi za vsak s' povezan s s, dokler ne pridemo do končnega stanja. Verjetnost $P(s'|s,a)$ določa, kolikšen vpliv ima izraz $R(s,a,s') + \gamma V_\pi(s')$ vpliv na vrednost stanja s. Pomembno se je zavedati, da potrebujemo za izračun enačbe celotno prehodno funkcijo P. Če je P znan, se enačba razširi na sistem

enačb, ki ga lahko rešimo enostaven način.

Bellman je opozoril na težavo, da se število izračunov, ki so potrebni za rešitev sistema enačb, veča eksponentno z številom vhodnih dimenzij. Poimenoval jo je *prekletstvo dimenzionalnosti (angl. the curse of dimensionality).* Gre za dobro znano težavo na področju strojnega učenja, izvira pa iz dejstva, da se prostor stanj veča eksponentno s številom vhodnih dimenzij.

Teorem izboljšanja politike (angl. policy imporvement theorem) navaja, da ko spremenimo pot politike na pot, ki je po vrednostni funkciji boljša, dobimo izboljšano politiko. Ta lastnost se uporablja tako v metodah dinamičnega programiranja kot tudi v veliki meri drugih algoritmov okrepitvenega učenja, kamor sodijo tudi algoritmi, ki so opisani v nadaljevanju.

3.2 Metode Monte Carlo

Za razliko od metod DP, metode *Monte Carlo (MC)* vrednostno funkcijo ocenjujejo na podlagi interakcije z okoljem in ne potrebujejo popolnega znanja o okolju. Metode Monte Carlo potrebujejo samo izkušnje – vzorčna zaporedja stanj, dejanj in nagrad – ki jih pridobijo med interakcijo z okoljem ali pa iz simuliranih interakcij z okoljem. Za presenetljivo veliko aplikacij je enostavno simulirati vzorčne epizode, čeprav je težko zgraditi celoten ekspliciten model prehodnih verjetnosti, ki jih potrebujejo DP metode.

Metode Monte Carlo so način reševanja problema okrepitvenega učenja z povprečenjem donosov stanj iz celotnih epizod. Ker so vrednosti stanj enostavno samo predviden donos – predvidena nabrana zakasnjena nagrada – z začetkom v tem stanju, lahko za oceno iz izkušenj preprosto povprečimo pridobljene donose iz tega stanja. Postopoma s pridobitvijo večje količine donosov povprečje konvergira k pričakovani vrednosti. Samo po zaključku epizode se ocenjene vrednosti in politika spremenijo. Učenje torej poteka postopoma iz epizode do epizode in ne iz enega stanja v drugega.

Če želimo oceniti $V_\pi(s)$, tj. vrednost nekega stanja s po politiki π, imenujemo vsak pojav stanja s v enem od epizod *obisk (angl. visit)* stanja s. *Metoda every-visit MC* oceni $V_\pi(s)$ kot povprečje donosov po vsakem obisku stanja s v vsaki epizodi, *metoda first-visit MC* pa uporablja samo povprečje donosov po prvem obisku stanja s v vsaki epizodi. Oba algoritma po pravilu o velikih številih konvergirata k pričakovani vrednosti $V_\pi(s)$, ko gre število obiskov stanja s v neskončnost [31].

Naivna implementacija povprečenja donosov je hranitev vsakega prejetega donosa R_t za vsako stanje. Ko nato potrebujemo oceno vrednosti stanja, lahko to enostavno izračunamo z

$$V_t(s) = \frac{R_1 + R_2 + \cdots + R_{K_s}}{K_s}, \tag{3.2}$$

kjer so $R_1, \ldots, R_{K_s}$ vsi prejeti donosi v stanju s pred časom t in K število prejetih donosov. Problem pri tej preprosti implementaciji je, da se spominske in računske zahteve čez čas neomejeno povečujejo. Vsak nov donos po obisku stanja potrebuje več spomina za shrambo in izračun $V_t(s)$ potrebuje več časa.

Takšna implementacija seveda ni potrebna. Enostavno je oblikovati postopno pravilo posodobitve, ki potrebuje konstanten čas in spomin. Naj bo V_k za neko stanje ocena vrednosti za k-ti donos, to je povprečje prvih $k-1$ donosov. Povprečje vseh k donosov lahko potem izračunamo z

$$\begin{aligned} V_{k+1} &= \frac{1}{k}\sum_{i=1}^{k} R_i \\ &= \frac{1}{k}\left(R_k + \sum_{i=1}^{k-1} R_i\right) \\ &= \frac{1}{k}\left(R_k + (k-1)V_k + V_k - V_k\right) \\ &= \frac{1}{k}\left(R_k + kV_k - V_k\right) \\ &= V_k + \frac{1}{k}\left[R_k - V_k\right], \end{aligned} \tag{3.3}$$

kar drži tudi za $k = 1$.

Ta oblika postopnega pravila posodobitve (3.3) se pogosto uporablja v algoritmih okrepitvenega učenja, kateri svoje ocene posodabljajo čez čas. Splošna oblika tega pravila je

$$NovaOcena \leftarrow StaraOcena + VelikostKoraka\left[Cilj - StaraOcena\right]. \tag{3.4}$$

Izraz $[Cilj - StaraOcena]$ je *napaka (angl. error)* v oceni vrednosti. Napaka se zniža s korakom proti *cilju (angl. target)*, ki predstavlja zaželeno smer, v katero bi se radi premaknili; v zgornjem primeru je to k-ti donos.

Parameter *velikosti koraka (angl. step-size)* se v zgornji metodi spreminja v vsakem časovnem koraku ($\frac{1}{k}$). V literaturi ga pogosto zaznamuje α in je podan v intervalu $[0, 1]$. Imenuje se tudi *stopnja učenja (angl. learning rate)*, saj določa hitrost, s katero se pomikamo proti ciljni vrednosti. V praksi je velikokrat konstanten ali pa se niža čez čas.

Enostavno pravilo posodobitve vrednosti nekega stanja s_t na podlagi every-visit MC metode lahko torej opišemo z

$$V(s_t) \leftarrow V(s_t) + \alpha[R_t - V(s_t)], \tag{3.5}$$

kjer je R_t dejanski donos. Monte Carlo metode počakajo dokler ni znan celoten donos stanja, preden ga uporabijo za oceno $V(s_t)$ – počakajo torej do konca epizode.

Pomemben dejavnik pri metodah MC je ta, da so ocene vsakega stanja neodvisne. Ocena enega stanja ne gradi na oceni kateregakoli drugega stanja, kot velja pri DP metodah. Lastnost grajenja ene ocene vrednosti na podlagi druge ocene vrednosti imenujemo *zagon (angl. bootstrapping).*

Z delnim modelom okolja so vrednostne funkcije zadostne za izbiro dejanj; enostavno pogledamo naprej, v katero stanje nas dejanje pelje. Brez modela pa vrednostne funkcije niso zadostne za krmiljenje. Eksplicitno moramo oceniti vrednost vsakega dejanja v vsakem stanju, da se bomo znali vesti. Ocena vrednostne funkcije dejanj, $Q(s, a)$ je v glavnem enaka, kot za $V(s)$:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_t - Q(s_t, a_t)]. \tag{3.6}$$

MacKay [6] povzema Monte Carlo metode v splošnem, Sutton in Barto [21] pa v aplikaciji na okreptivenem učenju.

3.3 Učenje na podlagi časovne razlike – TD(0)

Učenje na podlagi časovne razlike (angl. temporal-difference – TD) je ena izmed osrednjih idej v okrepitvenem učenju. Medtem ko metode Monte Carlo za izračun posodobitve $V(s_t)$ čakajo do konca epizode, metode TD počakajo en sam časovni korak. V času $t + 1$ že tvorijo ciljno vrednost in jo skupaj z zaznano nagrado r_{t+1} in oceno $V(s_{t+1})$ uporabijo za posodobitev ocene prejšnjega stanja $V(s_t)$. Najenostavnejša TD metoda, znana kot *TD(0)* je

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]. \tag{3.7}$$

Pri posodobitvi MC metode je ciljna vrednost R_t, pri TD pa $r_{t+1} + \gamma V_t(s_{t+1})$. Cilj posodobitvenega pravila za TD izhaja neposredno iz Bellmanove enačbe (3.1).

Tako kot metoda DP, tudi TD uporablja zagonsko lastnost grajenja ocene vrednosti prejšnjega stanja na podlagi ocene vrednosti trenutnega stanja. Hkrati pa ima TD tudi lastnost MC metode, da ne potrebuje modela okolja. Ključna prednost TD metod je, da posodabljajo ocene iz koraka v korak napram MC metodam, ki morajo počakati na celoten donos, še posebej ko so epizode dolge.

Število 0 v TD(0) se nanaša na dejstvo, da algoritem posodobi samo vrednost enega prejšnjega stanja. Obstaja seveda tudi zelo podoben algoritem za oceno vrednosti dejanj:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)], \tag{3.8}$$

z imenom *SARSA(0)* $(State_t, Action_t, Reward_{t+1}, State_{t+1}, Action_{t+1})$.

Obe metodi konvergirata k optimalni politiki, če je α dovolj majhen oziroma se zmanjšuje čez čas in če so vsa stanja oziroma vsi pari (stanje, dejanje) obiskani neskončnokrat [32].

3.4 Združitev metod – TD(λ)

TD(0) glede na obiskano stanje posodobi samo vrednost prejšnjega stanja. Novost pri *TD(λ)* algoritmu je sposobnost, da razširi znanje nazaj v poljubno število vrednosti prejšnjih stanj.

Za dosego tega se uvaja faktor λ v intervalu $[0, 1]$, ki določa koliko daleč nazaj se znanje širi – koliko prejšnjih stanj bomo posodabljali oziroma s kolikšnim vplivom jih bomo posodabljali. Z $\lambda = 0$ dobimo TD(0) metodo (znanje se razširi samo na prejšnje stanje), z $\lambda = 1$ pa MC metodo (znanje se razširi na vsa prejšnja stanja). TD(λ) predstavlja most med metodo MC in TD(0).

Splošna ciljna vrednost za TD(λ) je

$$R_t^{\lambda} = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)}, \tag{3.9}$$

kjer je donos $R_t^{(n)}$ razširjen iz ocene vrednosti enega koraka v prihodnosti

$$R_t^{(1)} = r_{t+1} + \gamma V(s_{t+1}) \tag{3.10}$$

na oceno vrednosti n-tega koraka v prihodnosti

$$R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n}). \tag{3.11}$$

Za primerjavo se pri metodi MC uporablja točen donos

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T. \tag{3.12}$$

Ciljna vrednost predstavlja povprečje donosov za vse različne dolžine korakov, kjer je vsak donos utežen z λ^{n-1}. Normalizacijski faktor $1 - \lambda$ poskrbi, da se uteži seštevajo v vsoto 1. Donos enega koraka dobi največjo utež, $1 - \lambda$; donos drugega koraka dobi drugo največjo utež $(1 - \lambda)\lambda$; donos tretjega koraka dobi utež $(1 - \lambda)\lambda^2$ in tako dalje. V vsakem koraku se utež zniža za λ.

Splošno pravilo posodobitve vrednostne funkcije lahko torej podamo z

$$V(s_t) \leftarrow V(s_t) + \alpha[R_t^{\lambda} - V(s_t)]. \tag{3.13}$$

Za praktično implementacijo algoritma potrebujemo dodatno spremenljivko za vsako stanje, ki določa kolikšen vpliv bo imelo neko prihodnje dejanje na vrednost tega stanja.

Te spremenljivke imenujemo *sledi primernosti (angl. eligibility traces)* e. Posodabljamo jih po pravilu *akumulacijskih sledi (angl. accumulating traces)*

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & \text{če } s \neq s_t; \\ \gamma\lambda e_{t-1}(s) + 1 & \text{če } s = s_t, \end{cases} \tag{3.14}$$

ali pa *zamenjalnih sledi (angl. replacing traces)*

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & \text{če } s \neq s_t; \\ 1 & \text{če } s = s_t, \end{cases} \tag{3.15}$$

kjer je γ faktor popuščanja, kot je opisano v predhodnem razdelku. Na vsakem koraku se sledi primernosti znižajo za $\gamma\lambda$, sled obiskanega stanja pa se zviša za 1 ali pa ponastavi na 1. Parameter λ se iz tega razloga imenuje tudi *parameter popuščanja sledi (angl. trace-decay parameter)*. Z akumulacijskimi sledmi se sled poveča za 1 ob vsakem obisku, kar jo lahko dvigne nad vrednostjo 1, medtem ko se pri zamenjalnih sled vedno ponastavi na 1. V poljubnem času sledi določajo, katera stanja so bila nedavno obiskana. $TD(1)$ z akumulacijskimi sledmi je logično povezan z every-visit MC, $TD(1)$ z zamenjavnimi sledmi, pa z first-visit MC [31]. Čeprav je razlika majhna, lahko zamenjalne sledi pripeljejo do znatno hitrejšega učenja [21].

Celoten algoritem $TD(\lambda)$ za napoved vrednosti z zamenjalnimi sledmi je predstavljen v tabeli 2. Napaka v oceni vrednosti pri TD metodah je

$$\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t) \tag{3.16}$$

katera se prenese na posodobitve vseh vrednosti stanj, ki imajo neničelne sledi:

$$\Delta V_t(s) = \alpha\delta_t e_t(s), \quad \forall s \in S. \tag{3.17}$$

TD algoritem za krmiljenje *SARSA(λ)* se je razvil iz TD(λ) preprosto z zamenjavo $V_t(s)$ za $Q_t(s,a)$ in $e_t(s)$ za $e_t(s,a)$ (tabela 3), podobno kot pri SARSA(0).

Singh in ostali so dokazali, da TD(λ) konvergira k optimalni rešitvi [32].

Tabela 2: Algoritem TD(λ) z zamenjalnimi sledmi.

Inicializiraj V poljubno
Za vsako epizodo:
  Inicializiraj $e(s) = 0$, $\forall s \in S$
  Inicializiraj s
  Za vsak korak:
    $a \leftarrow$ dejanje po π za s
    Izvedi dejanje a, opazuj nagrado r in naslednje stanje s'
    $\delta \leftarrow r + \gamma V(s') - V(s)$
    $e(s) \leftarrow 1$
    Za vsak $s \in S$:
      $V(s) \leftarrow V(s) + \alpha\delta e(s)$
      $e(s) \leftarrow \gamma\lambda e(s)$
    $s \leftarrow s'$
  dokler ni s končno stanje

Tabela 3: Algoritem SARSA(λ) z zamenjalnimi sledmi.

Inicializiraj Q poljubno
Za vsako epizodo:
  Inicializiraj $e(s, a) = 0$, $\forall s \in S, \forall a \in A(s)$
  Inicializiraj s, a
  Za vsak korak:
    Izvedi dejanje a, opazuj r, s'
    $a \leftarrow$ dejanje po π za s
    Izberi dejanje a' iz stanja s' po politiki iz Q (npr. ϵ-požrešno)
    $\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
    $e(s, a) \leftarrow 1$
    Za vsak $s \in S$ in $a \in A(s)$:
      $Q(s, a) \leftarrow Q(s, a) + \alpha\delta e(s, a)$
      $e(s, a) \leftarrow \gamma\lambda e(s, a)$
    $s \leftarrow s'; a \leftarrow a'$
  dokler ni s končno stanje

4 Posploševanje in funkcijska aproksimacija

Vsi algoritmi do sedaj so predpostavljali, da ocenjene vrednosti hranimo v tabeli, torej z enim vpisom za vsako vrednost. Težava je v tem, da je to mogoče le pri majhnemu številu stanj in dejanj. Problem ni samo v količini potrebnega spomina, ampak tudi v času in količini podatkov, ki so potrebni za pravilno ocenjevanje celotne tabele. Potrebujemo način za posploševanje znanja iz izkušenj omejenega števila stanj.

Posploševanje je izčrpno preučeno v obliki funkcijskih aproksimacij na področju nadzorovanega učenja – umetnih nevronskih mrežah, prepoznavi vzorcev in statističnem prilagajanju krivulj. Načeloma lahko katerokoli izmed teh metod uporabljamo v okrepitvenem učenju – posebej popularne so nevronske mreže in metode gradient descent.

4.1 Nevronske mreže

Eden izmed najzanimivejših in najbolj raziskanih funkcijskih aproksimatorjev se zgleduje po najspretnejšem posploševalcu v naravi – možganih. *Umetne nevronske mreže (angl. artificial neural network)* so v splošnem opredeljene kot sistem povezanih modelov nevronov, ki znajo iz več vhodnih vrednosti izračunati eno izhodno.

Nevroni se logično povzemajo kot linearna funkcija $y(\mathbf{x})$, kjer je vsaka vrednost iz vhodnega vektorja $\mathbf{x}$ utežena z vrednostjo iz *utežnega vektorja* $\mathbf{w}$:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \omega_0. \tag{4.1}$$

Če je vhodni vektor dvodimenzionalen, potem utežni vektor določa orientacijo ravnine in ω *(angl. bias)* določa pravokotno razdaljo ravnine od središča prostora. Na funkcijo lahko gledamo tudi s stališča funkcijske aproksimacije, kjer pri vhodu $\mathbf{x}$ spreminjamo $\mathbf{w}$, da dosežemo želen izhod $y(\mathbf{x})$. Pogosto se rezultat spusti še skozi monotonično nelinearno *aktivacijsko oziroma preklopno funkcijo* g:

$$y(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x} + \omega_0). \tag{4.2}$$

Čeprav je g nelinearna funkcija, je odločitvena meja še vedno linearna, ker je g monotonična.

Najpogostejši aktivacijski funkciji sta sigmoidni zaradi enostavnosti izračuna njunih odvodov: logistična funkcija (zaloga vrednosti $[0, 1]$)

$$g(a) = \frac{1}{1 + e^{-a}} \tag{4.3}$$

in hiperbolična tangentna funkcija (zaloga vrednosti $[-1, 1]$)

$$\tanh(a) = \frac{1 - e^{-2a}}{1 + e^{-2a}}. \tag{4.4}$$

Zaradi simetrije in na splošno boljših rezultatov se pri učenju priporoča uporaba tanh [37].

Za učenje nevrona – da vemo v katero smer moramo spremeniti utež – moramo definirati neko napako E glede na izhod nevrona:

$$\mathbf{w_{t+1}} = \mathbf{w_t} + \alpha E \mathbf{x_t}. \tag{4.5}$$

Posamezni nevroni se niso zmožni naučiti nelinearno ločljivih vzorcev, kot je na primer logična funkcija XOR [24]. Če pa nevrone povežemo v *večnivojsko feedforward nevronsko mrežo (angl. multilayer feedforward neural network)* z vsaj tremi nivoji in dovolj velikim številom nevronov v srednjem nivoju, postane mreža univerzalni aproksimator – poljubno natančno lahko aproksimira poljubno zvezno funkcijo, ki slika iz realnih števil v realna števila [16]. Feedforward pomeni, da mreža nima usmerjenih ciklov oziroma, da informacija iz vhodov na izhode potuje samo v eno smer.

Med probleme nevronskih mrež sodi veliko število spremenljivk, ki jih moramo določiti empirično (število nivojev, število nevronov v vsakem nivoju, stopnje učenja), neenotna predstavitev vhodov, začetne vrednosti uteži, počasna hitrost učenja, prekomerno prilagajanje (overfitting ali overtraining) in konvergiranje k lokalnem minimumu.

4.2 Združitev TD(λ) z nevronskimi mrežami

Učenje na podlagi časovne razlike TD(λ) lahko združimo z metodo *gradient descent* za izračun napake in uporabimo *backpropagation* algoritem za razširitev napake po nevronski mreži [2].

Posodobitveno pravilo za utež nevrona v tem primeru postane

$$\mathbf{w_{t+1}} = \mathbf{w_t} - \alpha(R_t^\lambda - V(s_t)) \frac{\partial V(s_t)}{\partial \mathbf{w_t}}. \tag{4.6}$$

Izraženo s sledmi primernosti:

$$\mathbf{w_{t+1}} = \mathbf{w_t} - \alpha \delta_t e_t, \tag{4.7}$$

kjer je δ_t *TD-napaka (angl. TD-error)*

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{4.8}$$

in $\mathbf{e_t}$ je vektor sledi primernosti, ena sled za vsako utež v vektorju $\mathbf{w_t}$,

$$\mathbf{e_t} = \gamma\lambda\mathbf{e_{t-1}} + \frac{\partial V(s_t)}{\partial \mathbf{w_t}} \tag{4.9}$$

z $\mathbf{e_0} = \mathbf{0}$.

Če politika vedenja oziroma vzorčenje sledi dinamiki MDP-ja v interesu, potem linearni funkcijski aproksimatorji s TD(λ) dokazano konvergirajo z verjetnostjo 1 [15]. Za offline vzorčenje in nelinearne funkcijske aproksimatorje, kot so nevronske mreže z backpropagation, pa ni zagotovljena konvergenca; številni primeri kažejo, da divergirajo [14], čeprav obstajajo tudi empirični uspehi, kot je TD-Gammon [11].

5 Učenje na namizni igri Hex

V tem poglavju so metode okrepitvenega učenja uporabljene na računalniški simulaciji namizne igre Hex. Najprej je opisana igra in njena pravila ter lastnosti, nato so predstavljene implementacije različnih metod. Na koncu poglavja so primerjani še rezultati iz učenja.

5.1 Ozadje

Hex je abstraktna potezna strategija za dva igralca. Igra se na heksagonalni mreži, katera je navadno urejena v obliki romba. Velikost mreže je poljubna. Medtem ko je v namizni različici pogosto v velikosti 11×11, so v tem primeru zaradi velikega števila stanj raziskane predvsem velikosti 3×3 in 4×4.

Vsak od igralcev ima dodeljeno eno barvo, pri čemer so pogoste zlasti rdeča in modra ali bela in črna. Igralci izmenično zavzemajo polja eno po eno, dokler eden izmed igralec ne poveže nasprotni stranici svoje barve. Zmaga tisti igralec, ki prvi vzpostavi povezavo med svojima stranicama.

Hex spada v kategorijo končnih (tj. konča se v končnem številu korakov) determinističnih (tj. brez naključnih dejavnikov) iger za dve osebi s popolno informacijo o stanju. Igra se nikoli ne more končati neodločno – nasprotniku lahko povezavo stranic preprečimo samo tako, da povežemo svoji stranici. Ko so stranice mreže enako dolge, ima prednost prvi igralec. Ker je Hex končna igra s popolno informacijo, ki se ne more končati neodločeno, sledi, da ima ali prvi ali drugi igralec zmagovalno strategijo. Če predpostavimo, da ima drugi igralec zmagovalno strategijo, jo lahko prvi igralec "ukrade" s tem, da naredi poljubno potezo in nato sledi zmagovalni strategiji drugega igralca. Dodatna poteza za bodisi igralca v kateremkoli položaju lahko samo izboljša položaj tega igralca, kar pomeni, da je prvi igralec v prednosti in ima on zmagovalno strategijo. Formalen dokaz je Maarup opisal v [35]. Ta argument samo dokazuje obstoj zmagovalne strategije in je ne opisuje eksplicitno.

Even in Tarjan [28] sta dokazala, da je ugotavljanje, ali je določen položaj v igri Hex zmagovalen, PSPACE-polno. Splošno se domneva, da se PSPACE-polnih problemov ne da rešiti z učinkovitimi algoritmi (tj. v polinomskem času).

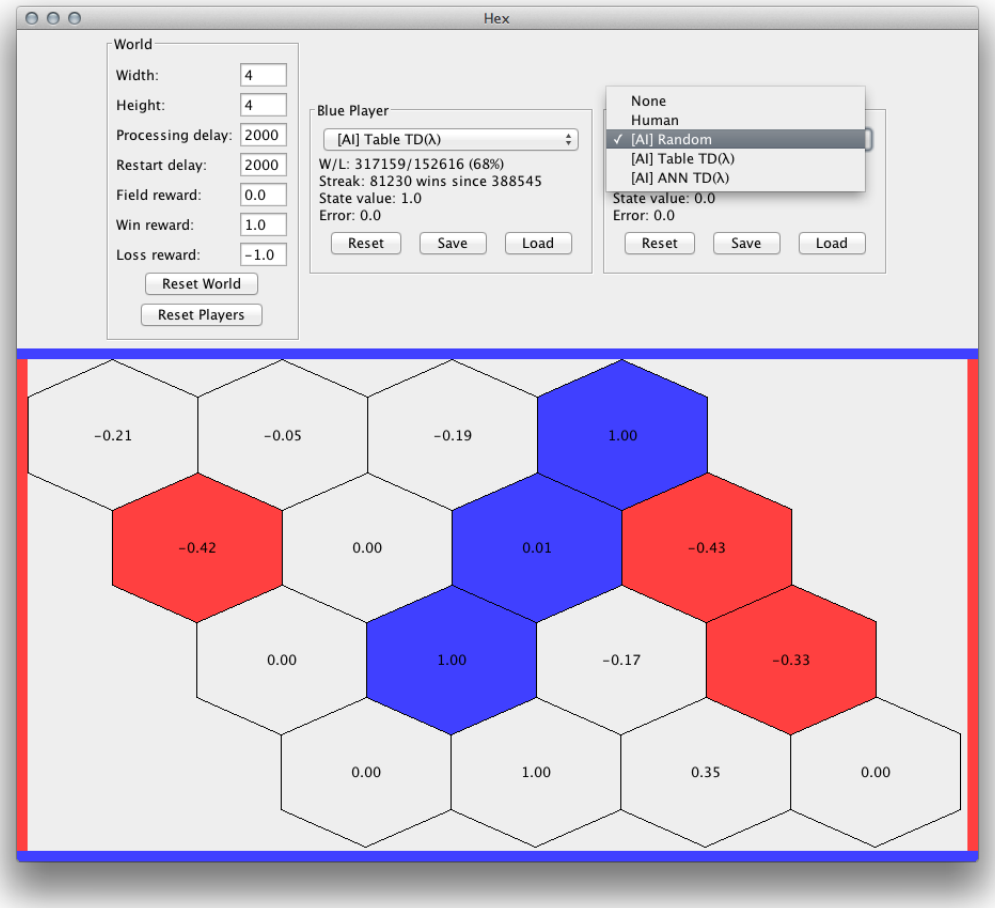
5.2 Implementacija

Igra – tako okolje kot igralci – je implementirana v programskem jeziku Java za izkoristek močne standardne knjižnice in enostavne izvedbe grafičnega vmesnika za vse tri glavne operacijske sisteme.

Celotna naloga, vključno z izvorno kodo in LaTeX dokumenti, je dostopna na spletnem naslovu `https://github.com/SephiRok/Hex`.

5.2.1 Grafični vmesnik

Spremljajoče k igralni logiki in algoritmom učenja je dodelan grafični vmesnik.



Slika 3: Grafični vmesnik implementacije igre Hex.

V zgornjem delu okna najdemo vrsto nastavitev, katere lahko spreminjamo med delovanjem programa. Na levi so podani parametri okolja: velikost mreže, zakasnitev pri obdelavi, zakasnitev po končani igri, nagrada za posamezno potezo, nagrada za zmago, nagrada za poraz in gumbi za ponastavitev okolja ali igralcev. Na desni pa

so podani parametri igralcev. Za oba igralca – modrega in rdečega – lahko izbiramo med implementiranimi učenci. Modri igralec ima prvi korak pri vsaki igri. Na voljo so naslednje možnosti: brez igralca, človeški igralec, naključni igralec, TD(λ) tabularni učenec in TD(λ) učenec z nevronsko mrežo. Implementacija učencev je podrobneje opisana v sledečih razdelkih. Pod imenom učenca najdemo tudi nekaj njegove statistike: razmerje med zmagami in porazi, število zaporednih zmag, ocena vrednosti trenutnega stanja in ocena napake. Učence (njihove naučene vrednosti ter statistiko) lahko tudi ponastavimo, shranimo za poznejšo rabo ali pa naložimo v program.

V spodnjem delu okna se nahaja igralna mreža v obliki romba. Stranice igralne površine so v barvi posameznega igralca. Za zmago mora igralec povezati stranici svoje barve. Če je določen človeški igralec, lahko s klikom na posamezno polje zavzamemo prosta polja. Pred potezo igralca z umetno inteligenco pa je v sredini posameznega polja prikazana njegova ocenjena vrednost za zavzem tega polja.

Grafični vmesnik in računski del sta v ločenih nitih. Med seboj komunicirata s pošiljanjem sporočil.

5.2.2 Okolje

Okolje igre je zgrajeno na osnovnih načelih okrepitvenega učenja, ki so opisani v razdelku 2.1.

Osnovna zanka okolja je opisana s psevdokodo v tabeli 4.

Tabela 4: Osnovna zanka okolja igre Hex.

```
Okolje ponastavi na začetno stanje
Sporoči začetek igre
Ponavljaj:
  Če ni zmagovalca:
    Naslednjemu igralcu sporoči stanje in ga vprašaj za potezo
    Potezo sporoči grafičnemu vmesniku
    Preveri, če je poteza zmagovalna
    Igralcu sporoči nagrado za potezo, zmago ali poraz
  drugače:
    Sporoči zaključek igre in zmagovalca
    Okolje ponastavi na začetno stanje
    Sporoči začetek igre
```

Stanje je s strani okolja opisano preprosto z opredelitvijo imetnika vsakega polja.

Nagrajevalna funkcija je določena na naslednji način:

$$r(s) = \begin{cases} 1.0 & \text{za zmago;} \\ -1.0 & \text{za poraz;} \\ 0.0 & \text{drugače,} \end{cases} \tag{5.1}$$

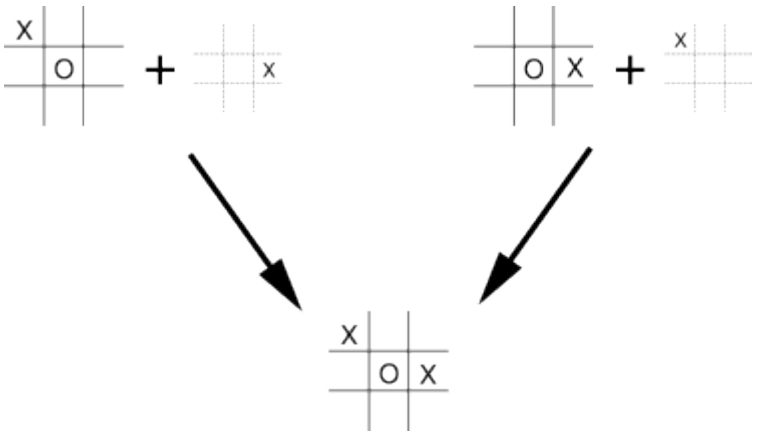
za učenje najkrajše poti za povezavo dveh stranic pa je zanimiva tudi funkcija

$$r(s) = \begin{cases} 1.0 & \text{za zmago;} \\ -1.0 & \text{za poraz;} \\ -1.0 & \text{drugače.} \end{cases} \tag{5.2}$$

5.2.3 TD(λ) tabularni učenec

Namesto učenja vrednosti stanj ali parov stanje-dejanje *pred* potezo kot je običajno, ta učenec ocenjuje vrednosti stanj *po* svoji potezi. Sutton in Barto [21] takšna stanja imenujeta *afterstate*, njihovo vrednostno funkcijo pa *afterstate value function*. Takšen pristop je uporaben, ko vemo, kakšna bo neposredna posledica našega dejanja – imamo delni model okolja – kot navadno velja pri igrah. Pri igri Hex vemo, kakšno bo stanje igre za vsako naše možno dejanje. Afterstate ocene so naravna pot za izrabo tega znanja in privedejo do učinkovitejše metode učenja.

Razlog, zakaj je algoritem učinkovitejši, če je zasnovan na stanjih afterstate, je razviden iz slike 4. Konvencionalne metode slikajo stanja *in* poteze v ocene vrednosti, ampak veliko teh parov doprinese do enakega sledečega stanja. Afterstate ocene ti dve enakovredni oceni združijo v eno, kar privede do hitrejšega učenja – učenje o enem dejanju se avtomatično prenese na drugo dejanje.



Slika 4: Dva različna para stanja in dejanja privedeta do enakega posledičnega stanja [21].

Pri implementaciji ocenjevanja afterstate vrednosti pa nastopi tudi težava. Algoritma TD(λ) (tabela 2) ne moremo uporabiti neposredno, ker je namenjeno napovedo-

vanju vrednosti stanja pred dejanjem in ne po dejanju – nagrade pri TD(λ) prispevajo k oceni stanja pred dejanjem. Težavo lahko nazorno prikažemo s primerom poteze do končnega stanja pri igri Hex. Predpostavljajmo, da smo v stanju s_t eno potezo pred zmago. Ko izvedemo zmagovalno potezo a_t preidemo v končno stanje s_{t+1} in dobimo nagrado $r_{t+1} = 1$. Po TD(λ) se ta nagrada upošteva za posodobitev vrednosti stanja pred zmagovalno potezo s_t, kar ni ustrezno za naš primer. Če nagrado tako upoštevamo, nagrada sploh ne bo vplivala na stanje ki je pravzaprav povzročilo zmago s_{t+1}. S tem povsem izgubimo vrednotenje končnega stanja; nagrade so v splošnem nepravilno zamaknjene za en časovni korak:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]. \tag{5.3}$$

Če se zgledujemo po krmilnem algoritmu SARSA(λ) (tabela 3), opazimo, da ocenjuje vrednost samega dejanja $Q(s, a)$. Vrednost izvedenega dejanja $Q(s_t, a_t)$ se posodobi z uporabo vrednosti naslednjega dejanja $Q(s_{t+1}, a_{t+1})$. Nagrada se za naš primer pravilno uporabi za posodobitev izvedenega dejanja a_t. Težava za afterstate vrednosti v tem primeru nastopi pri pridobitju vrednosti za naslednje stanje. Ker je po našem dejanju na vrsti za potezo naslednji igralec, stanja po tem v katerega preidemo ne moremo predvideti:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)]. \tag{5.4}$$

Ključ rešitve je ravno pri zamiku nagrade. V času t za predkončno stanje s_t moramo uporabiti nagrado r_t namesto nagrade r_{t+1}. Po končani igri pa moramo zato sprožiti učenje še za končno stanje s_{t+1} in uporabiti nagrado r_{t+1}. Z drugimi besedami: za posodabljanje prejšnje afterstate vrednosti moramo uporabiti prejšnjo nagrado:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_t + \gamma V(s_{t+1}) - V(s_t)]. \tag{5.5}$$

Celoten algoritem učenca je podan v tabeli 5. Vrednosti stanj in sledi primernosti se v tabelo dodajo ob prvem obisku v življenju učenca, privzeto imajo neobiskana stanja vrednost $V(s) = 0$ in sledi primernosti $e(s) = 0$. $A(s)$ je množica vseh možnih dejanj v stanju s, množica P pa vsebuje vsa obiskana stanja v posamezni igri. Ostale oznake so enake kot v prejšnjih poglavjih.

Parametri so bili določeni empirično. Faktor popuščanja γ je nastavljen na 1, saj gre za epizodični problem, stopnja učenja α na 0.95, parameter popuščanja sledi λ na 0.6 in ϵ na 0.

5.2.4 TD(λ) učenec z nevronsko mrežo

Ta učenec uporablja tudi afterstate vrednosti, kot je opisano v prejšnjem razdelku 5.2.3. Algoritem se drugače nanaša povsem na TD(λ), metodo gradient descent in na algoritem backpropagation, kot so opisani v 4. poglavju.

Tabela 5: Algoritem tabularnega TD(λ) učenca za igro Hex.

Inicializiraj obiskana stanja v tej igri $P = \emptyset$
Za vsako igro:
  $e(s) \leftarrow 0, \forall s \in P$
  $s \leftarrow$ stanje iz okolja
  $P \leftarrow \{s\}$
  $r \leftarrow 0$
  Za vsako zahtevano potezo:
    $s \leftarrow$ stanje iz okolja
    Ponastavi Q
    Za vsak $a \in A(s)$:
      Simuliraj dejanje a, opazuj naslednje stanje s'
      $Q(a) \leftarrow$ V(s')
    $a \leftarrow \max_a Q(a)$ oz. naključno dejanje ϵ-verjetno
    Izvedi dejanje a in opazuj naslednje stanje s'
    $\delta \leftarrow r + \gamma V(s') - V(s)$
    $e(s) \leftarrow 1$
    Za vsak $s \in P$:
      $V(s) \leftarrow V(s) + \alpha\delta e(s)$
      $e(s) \leftarrow V(s) + \gamma\lambda e(s)$
    $P \leftarrow P \cup \{s\}$
    $r \leftarrow$ nagrada iz okolja
  dokler se igra ne zaključi
  $\delta \leftarrow r + 0 - V(s)$
  $e(s) \leftarrow 1$
  Za vsak $s \in P$:
    $V(s) \leftarrow V(s) + \alpha\delta e(s)$
    $e(s) \leftarrow V(s) + \gamma\lambda e(s)$

Nevronska mreža je sestavljena iz treh polnopovezanih slojev: vhodni, skriti in izhodni. V vhodnem sloju je širina × višina × 2 nevronov ter bias nevron. Ti predstavljajo vhode na enostaven način: vsako polje je predstavljeno z dvema nevronoma. Eden od njiju dobi vrednost 1, če ima prvi igralec to polje, drugače 0. Drugi dobi vrednost 1, če ima drugi igralec isto polje, drugače 0. V skritem sloju je število vhodnih nevronov × 1, 5 nevronov ter bias nevron. Izhodni nevron je samo eden in določa vrednost stanja oziroma stanja afterstate. Bias nevroni držijo konstantno vrednost 1. Aktivacijska funkcija za vse nevrone razen izhodnega je hiperbolična tangentna funkcija (4.4). Izhodni nevron ima linearno aktivacijsko funkcijo.

Faktor popuščanja γ je nastavljen na 1, saj gre za epizodični problem, globalna stopnja učenja α na $0,001$, parameter popuščanja sledi λ na $0,1$ in ϵ na 0,2.

5.2.5 Naključni igralec

Naključni igralec preprosto izbira naključno dejanje med možnimi dejanji $A(s)$ iz vsakega stanja $s \in S$.

5.3 Rezultati

Ker je lahko prvi igralec v igri Hex s pravilno strategijo vedno zmagovalec, temeljijo vsi preizkusi učenja na prvem oziroma modrem igralcu. Cilj se je potemtakem naučiti optimalne strategije za vse možne položaje. Za učenje se bo v eksperimentih prvi igralec pomeril s tabularnim učencem in nato z naključnim učencem. Najprej bodo raziskani primeri za velikost 3×3 in nato še za 4×4.

Vsi podatki so dobljeni iz same aplikacije s pomočjo izpisovanja statistike v tekstovno datoteko ob koncu vsake igre. Povprečne napake v oceni vrednosti stanja $\overline{\delta}$ so izračunane z enostavnim drsečim povprečjem (3.3) TD(λ) napak δ_t:

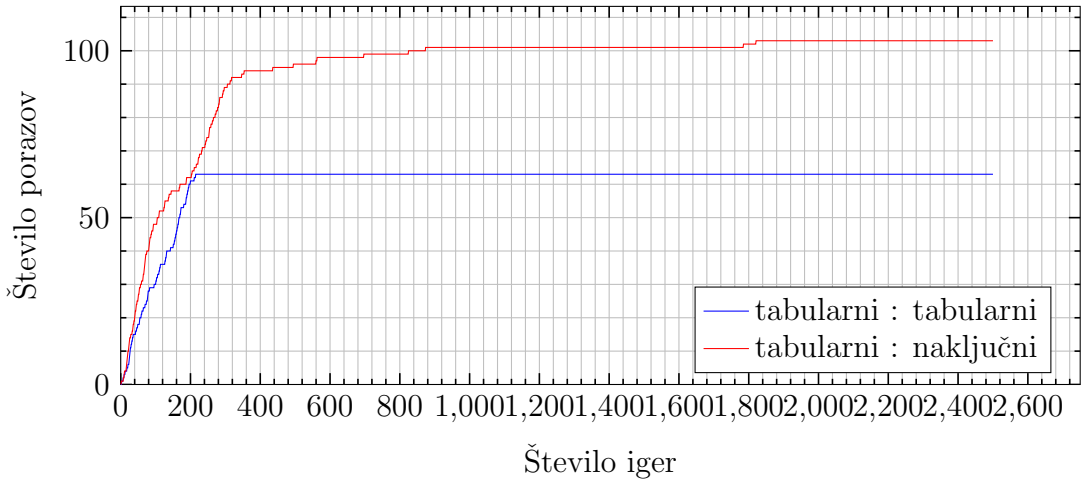
$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t), \tag{5.6}$$

$$\overline{\delta_t} = \overline{\delta_{t-1}} + \frac{1}{t}\left[\delta_t - \overline{\delta_{t-1}}\right]. \tag{5.7}$$

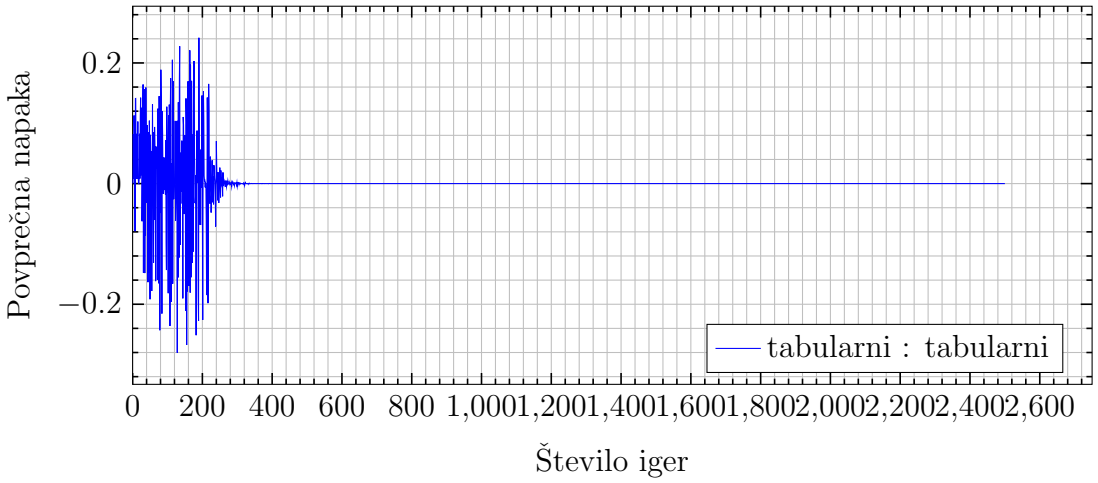
5.3.1 Hex 3×3

Pri velikosti mreže 3×3 se najboljše izide tabularni učenec, ko se ima možnost učiti proti enakemu algoritmu tudi v vlogi drugega igralca (slika 5). V tem primeru gre za teoretično dokazan algoritem, ki konvergira k optimalni rešitvi, kar učenca pomaga naučiti brezhibnega igranja v samo 215 igrah (63 porazih). Če pa tabularnemu učencu prepustimo samo naključnega nasprotnika, ta porabi znatno več časa, da se nauči

pravilno ocenjevati vrednosti stanj, saj je kakovost potez nasprotnika veliko nižja. V eksperimentih je za popolno napoved vrednosti stanj potreboval 1821 iger (103 poraze). V grafih na slikah 6 in 7 vidimo, da je učenje v primeru tabularnega proti tabularnemu veliko bolj zgoščeno.

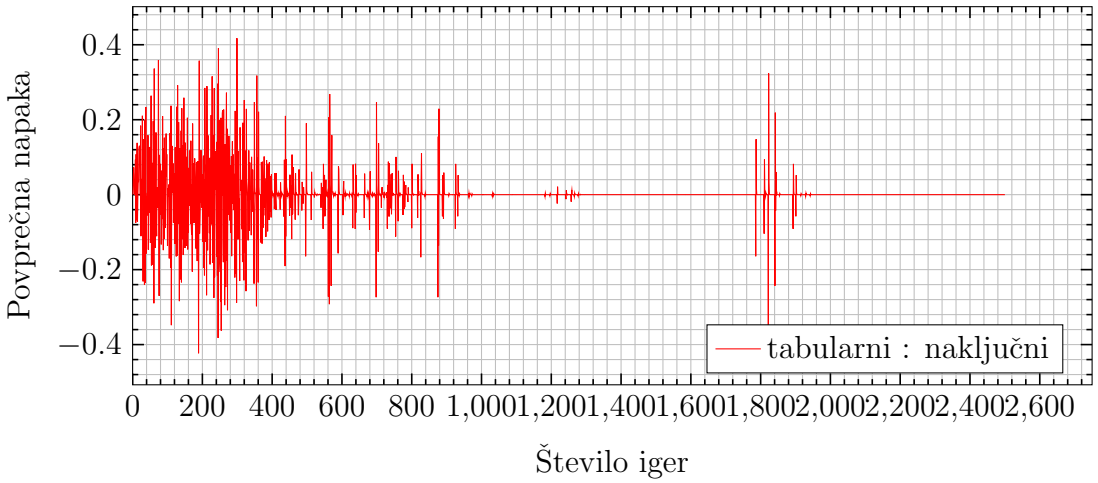


Slika 5: Graf porazov pri učenju TD(λ) tabularnega učenca za igro Hex 3×3.

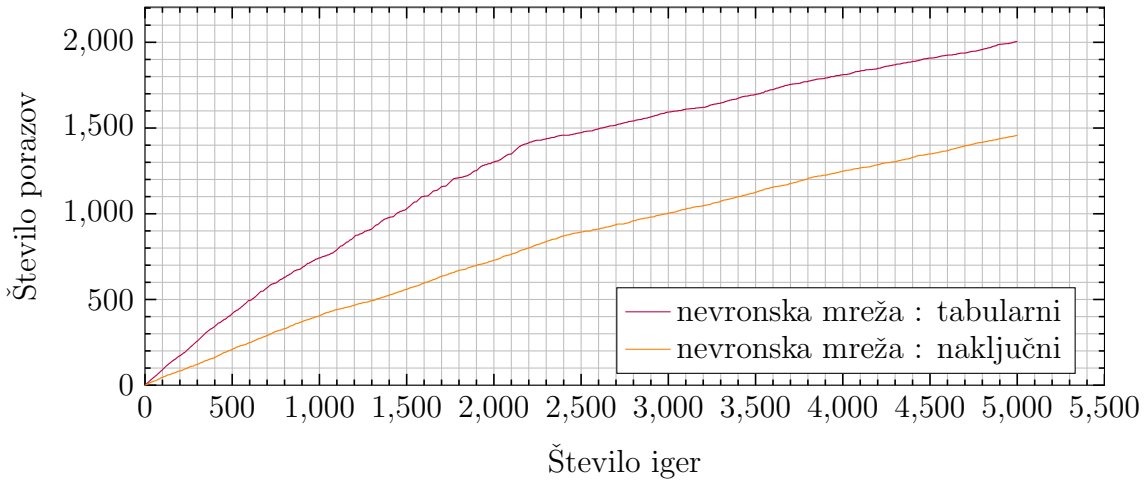


Slika 6: Graf povprečne napake pri učenju TD(λ) tabularnega učenca proti TD(λ) tabularnemu učencu za igro Hex 3×3.

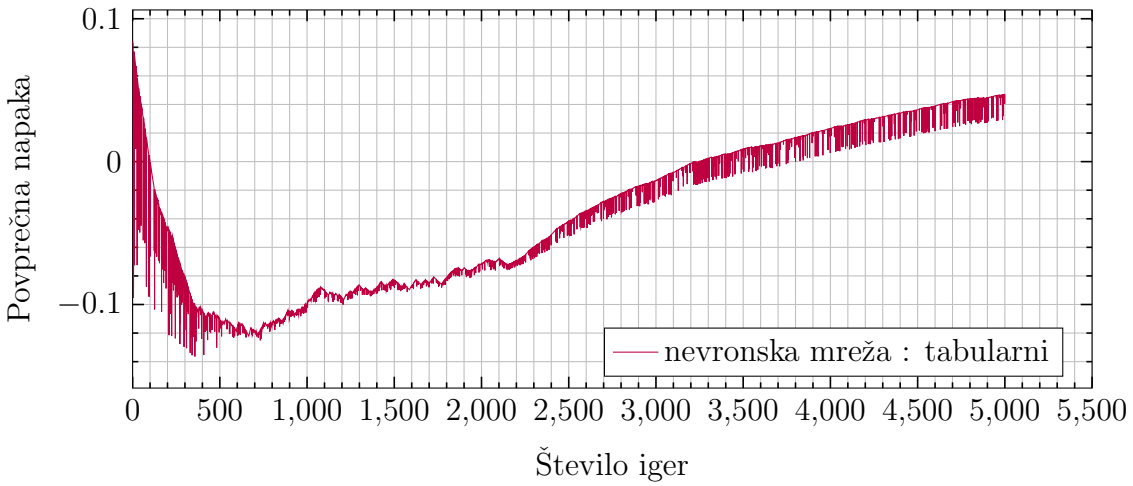
Pri učencu z nevronsko mrežo pa je zgodba drugačna. Čeprav se učenec nauči zadostno količino dobrih potez za zmagati približno 75 % iger proti naključnemu učencu in približno 60 % iger proti tabularnemu učencu (slika 8), algoritem ne konvergira k pravilni rešitvi in se nikoli ne nauči optimalne strategije (sliki 9 in 10).



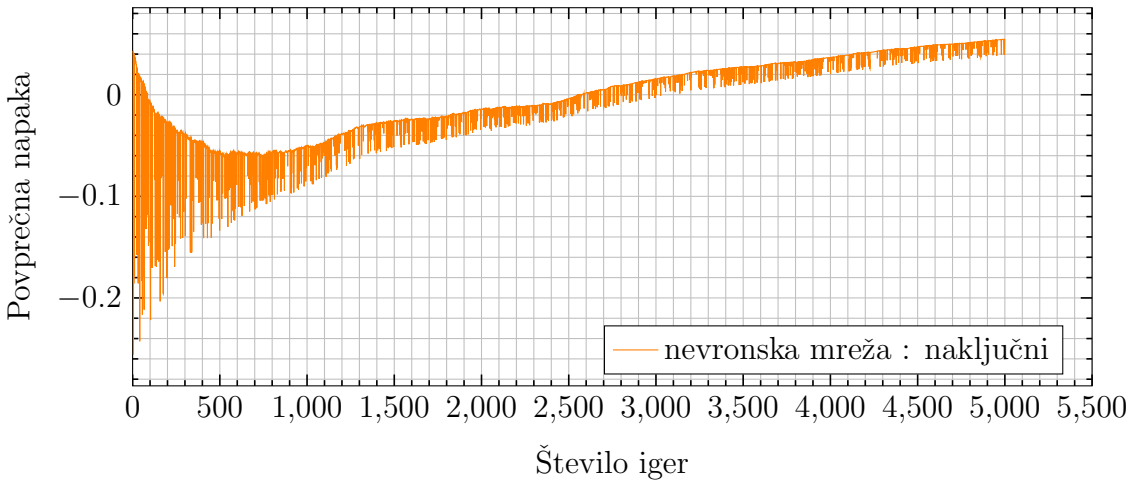
Slika 7: Graf povprečne napake pri učenju TD(λ) tabularnega učenca proti naključnemu igralcu za igro Hex 3×3



Slika 8: Graf porazov pri učenju TD(λ) učenca z nevronsko mrežo za igro Hex 3×3.



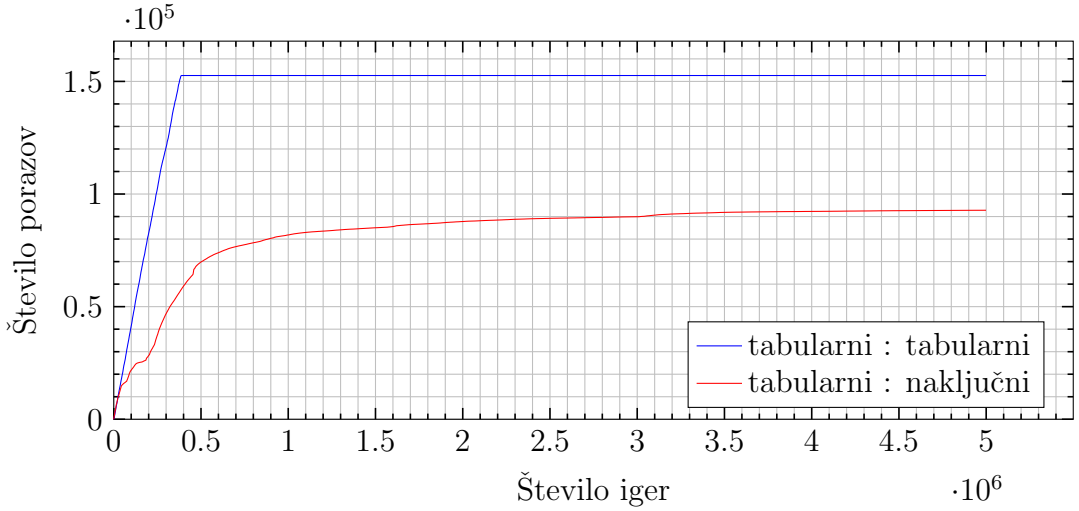
Slika 9: Graf povprečne napake pri učenju TD(λ) učenca z nevronsko mrežo proti TD(λ) tabularnemu učencu za igro Hex 3×3



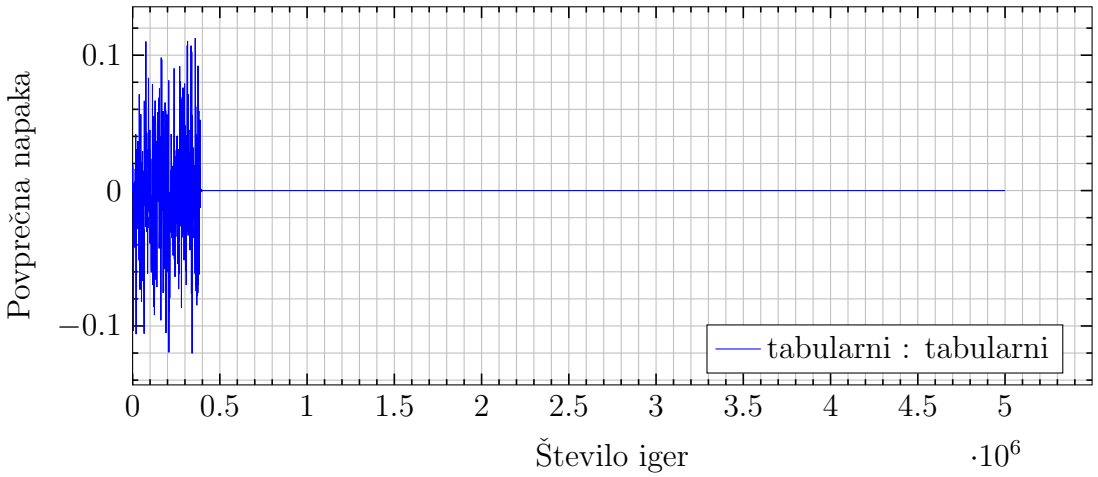
Slika 10: Graf povprečne napake pri učenju TD(λ) učenca z nevronsko mrežo proti naključnemu učencu za igro Hex 3×3

5.3.2 Hex 4×4

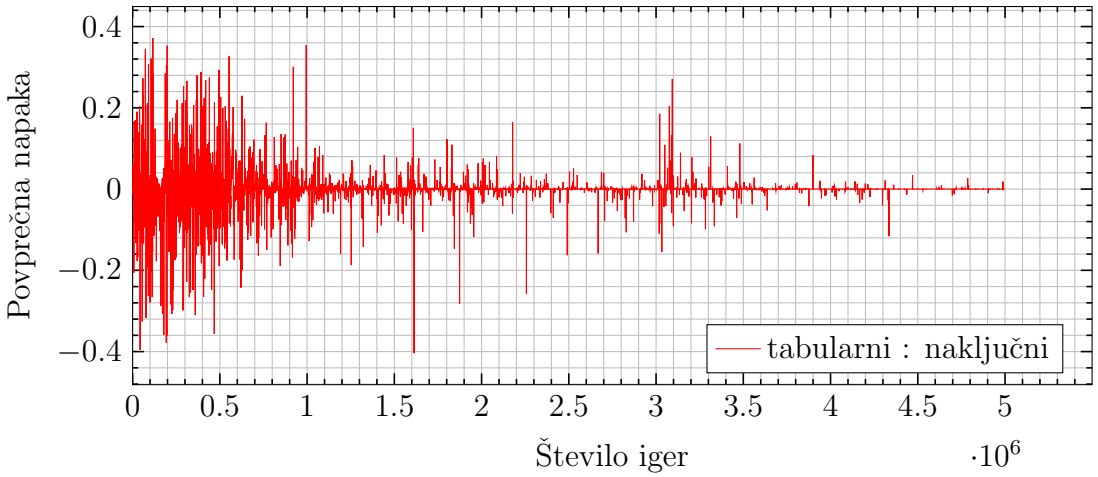
Tabularni učenec pri velikosti 4×4 ponovno dokazuje konvergentne lastnosti, vendar zaradi veliko večje količine stanj potrebuje veliko več iger, da se nauči vseh kombinacij (slika 11). V boju proti tabularnemu igralcu se vseh možnosti nauči v 400.000 igrah (150.000 porazih), v boju proti naključnemu igralcu pa se pravilnim vrednostim zelo približa v 5.000.000 igrah (100.000 porazih), vendar se še vedno pojavljajo tudi stanja pri katerih predvidi napačno vrednost, kar privede do redkih porazov (sliki 12 in 13). Slabost tabelnega pristopa pri velikosti 4×4 postane očitna, nesposobnost posploševanja privede do dolgotrajnega učenja ter velike količine potrebnega delovnega spomina (tabeli 6 in 7).



Slika 11: Graf porazov pri učenju TD(λ) tabularnega učenca za igro Hex 4×4.

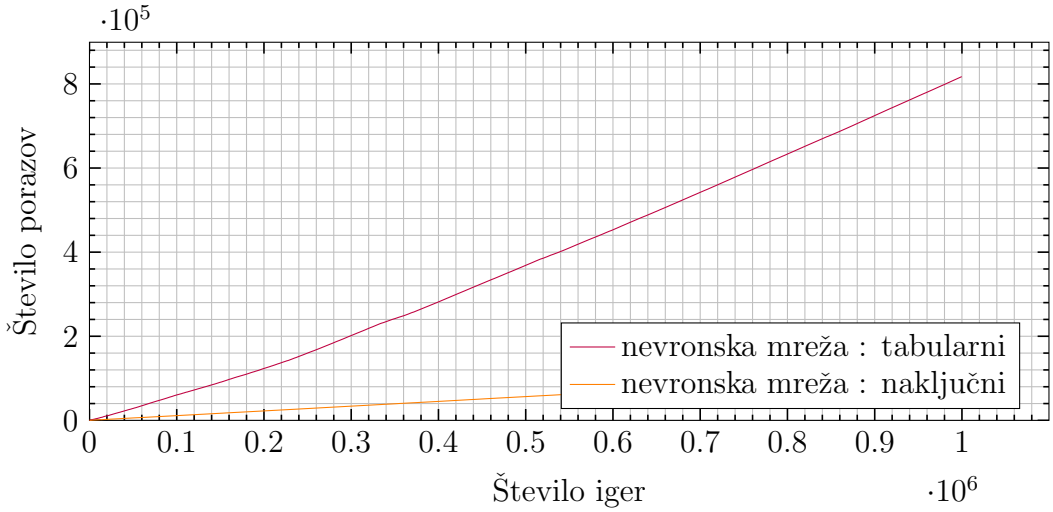


Slika 12: Graf povprečne napake pri učenju TD(λ) tabularnega učenca proti TD(λ) tabularnemu učencu za igro Hex 4×4



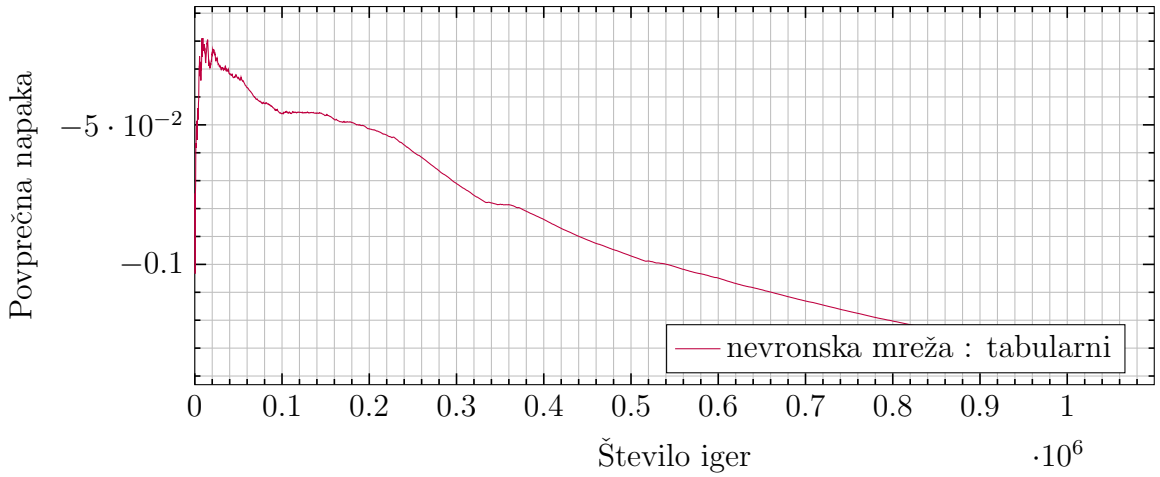
Slika 13: Graf povprečne napake pri učenju TD(λ) tabularnega učenca proti naključnemu učencu za igro Hex 4×4

Medtem ko ima pri manjšem številu stanj učenec z nevronsko mrežo občutno slabost proti tabelnemu učencu, lahko pri 4×4 vidimo prednosti pri posploševanju, uporabi spomina ter hitrosti učenja. Na žalost pa opazimo tudi nadaljevanje slabih zagotovil o konvergentnosti (slika 14). V primeru proti tabularnemu učencu divergira (slika 15), proti naključnemu pa obstane v lokalnem minimumu (slika 16).

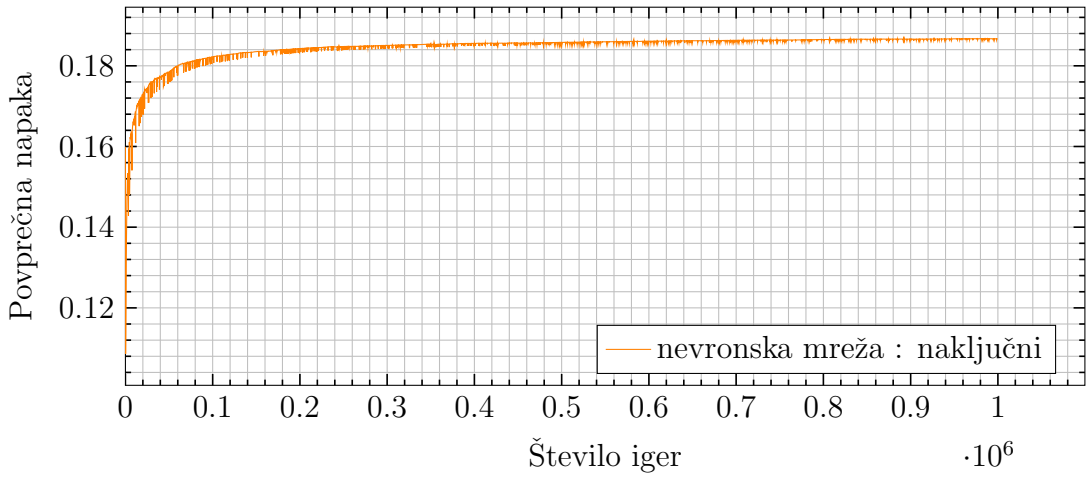


Slika 14: Graf porazov pri učenju TD(λ) z nevronsko mrežo za igro Hex 4×4.

Prostor za predstavitev stanj in njihovih vrednosti zavzema v vseh primerih, razen pri 4×4 tabularnem učencu, zanemarljivo količino prostora, čeprav v Java okolju zaradi grafičnega vmesnikom to predstavlja 80 MB za zanemarljivo količino podatkov.



Slika 15: Graf povprečne napake pri učenju TD(λ) učenca z nevronsko mrežo proti TD(λ) tabularnemu učencu za igro Hex 4×4



Slika 16: Graf povprečne napake pri učenju TD(λ) učenca z nevronsko mrežo proti naključnemu učencu za igro Hex 4×4

Tabela 6: Število iger in porazov do ničelne povprečne napake učencev igre Hex.

Učenec : nasprotnik	**Napaka**	**Število iger**	**Število porazov**
3×3 tabularni : tabularni	0	215	63
3×3 tabularni : naključni	0	1.821	103
3×3 nevronska mreža : tabularni	$0,0003$	3.414	1.679
3×3 nevronska mreža : naključni	$0,0003$	2.550	1.651
4×4 tabularni : tabularni	0	≈ 428.000	≈ 150.000
4×4 tabularni : naključni	≈ 0	$\approx 5.000.000$	≈ 100.000
4×4 nevronska mreža : tabularni	Divergira	/	/
4×4 nevronska mreža : naključni	$0,18$	69.000	7.845

Tabela 7: Prostorske zahteve učencev igre Hex.

Učenec	**Delovni spomin za učenje**	**Shranjen na disku**	**Shranjen na disku (stisnjeno)**
3×3 tabularni	80 MB	66 kB	5 kB
3×3 z nevronsko mrežo	80 MB	6 kB	9 kB
4×4 tabularni	900 MB	148 MB	9 MB
4×4 z nevronsko mrežo	80 MB	27 kB	17 kB

6 Zaključek

Iz okrepitvenega učenja so se razvili trdni matematični temelji in impresivne aplikacije. Računska študija okrepitvenega učenja je sedaj obsežna, z aktivnimi raziskovalci v raznolikih disciplinah, kot so psihologija, teorija krmiljenja (angl. control theory), operacijske raziskave (angl. operations research), umetna inteligenca in nevroznanost. Posebej pomembne so zveze z optimalnim nadzorom in dinamičnim programiranjem. Celoten problem učenja iz interakcije za dosego ciljev še zdaleč ni rešen, vendar se je naše razumevanje na tem področju bistveno izboljšalo. Sedaj lahko postavimo sestavne ideje, kot so učenje na podlagi časovne razlike, dinamično programiranje in funkcijske aproksimacije skladno s celotnim problemom.

Eden večjih trendov v okrepitvenem učenju je večji stik med umetno inteligenco in ostalimi inženirskimi disciplinami. Nedolgo nazaj se je umetna inteligenca dojemala kot nekaj popolnoma ločenega od teorije nadzora in statistike [21]. Imela je opravka z logiko in simboli, ne pa s števili. Umetno inteligenco so v preteklosti namesto linearne algebre, diferencialne enačbe ali statistike sestavljali obširni LISP programi. V zadnjih desetletjih se je ta pogled spremenil. Sodobni raziskovalci umetne inteligence sprejemajo statistične in nadzorne algoritme kot pomembne konkurenčne metode ali pa enostavno kot orodja. Prej prezrta področja med umetno inteligenco in konvencionalnega inženirstva so danes med najbolj aktivnimi, vključno z nevronskimi mrežami, pametnim nadzorom in okrepitvenim učenjem. V okrepitvenem učenju se ideje optimalne teorije nadzora in stohastične aproksimacije razširijo za nasloviti širše in bolj ambiciozne cilje na področju umetne inteligence.

Pri implementaciji okrepitvenega učenja na iskanju optimalnih rešitev pri namizni igre Hex se hitro pokaže največji izziv na področju strojnega učenja – ogromna količina stanj. Čeprav je nenadzorovana rešitev za igralne plošče majhnih dimenzij elegantna in razumljena, se pri posploševanju in funkcijski aproksimaciji hitro zatakne. Neposredno prilagajanje učenja na podlagi časovne razlike na nevronsko mrežo nima dovolj strogih zagotovil o konvergentnosti k optimalni rešitvi, da bi v splošnem ustrezalo za rešitev igre Hex. Težava je toliko bolj očitna, ker človeški igralci hitro vidijo najpreprostejšo pot od ene stranice do druge, medtem ko okrepitveni učenec tega ne zna sklepati iz okolja brez predhodnega znanja. Karkoli manj od optimalne rešitve se zaradi tega človeškemu igralcu zdi zelo pomanjkljivo ter zlahka presegljivo.

Niso pa vsi problemi tako občutljivi na točen odgovor. Okrepitveno učenje je bilo zelo uspešno uporabljeno v komercialnih videoigrah, kjer je največji uspeh široko znan v igri Black & White. Rešitev problema v Black & White je veliko bolj približen. Predstavljeni smo z bitjem, ki mu lahko spreminjamo vedenje z okrepitvijo in kaznovanjem. Cilje, ki si jih lahko sami zadamo med igro so v veliki meri bolj odprte narave, zaradi česar smo tudi bolj sprejemljivi do delnih rezultatov.

Razvijalec umetne inteligence igre Black & White ter ustanovitelj podjetja DeepMind Technologies Demis Hassabis je s svojimi kolegi v [36] predstavil prvi model *globokega učenja (angl deep learning)*, ki se je sposoben naučiti politike krmiljenja neposredno iz visokodimenzijskih senzoričnih vhodov. Uporabili so naprednejši tip nevronske mreže, ki je naučena na izvornih točkah zaslona z okrepitvenim učenjem za napovedovanje nagrad. Metodo so izvedli na sedmih Atari 2600 igrah in prekašali vse prejšnje rešitve v šestih primerih, zmožnost človeškega mojstra pa so presegli v treh. Sedaj je Hassabisovo podjetje DeepMind Technologies del podjetja Google.

Sutton in drugi so nedavno razvili nekaj posodobljenih algoritmov, ki dokazljivo konvergirajo k lokalni rešitvi z nelinearnimi funkcijskimi aproksimatorji, kot so nevronske mreže, vendar jih še razvijajo na probleme krmiljenja in potrebujejo več empiričnih podatkov iz obsežnejših realnih aplikacij [26, 12].

Ray Kurzweil – izumitelj, futurist, vodja oddelka za inženirstvo pri podjetju Google – v svoji knjigi "The Singularity Is Near: When Humans Transcend Biology" [19] opisuje svoj zakon o pospeševanju donosov, ki napoveduje eksponentno povečanje v tehnologijah, kot so računalništvo, genetika, nanotehnologija, robotika in umetna inteligenca. Predvideva, da bo do leta 2020 na voljo računalnik za tisoč ameriških dolarjev, ki bo imel računsko zmožnost posnemati človeško inteligenco. Po tem, pričakuje, da bo tehnologija optičnega zajemanja človeških možganov pripomogla k oblikovanju učinkovitega modela človeške inteligence do okoli leta 2025. Ta dva elementa bi računalnikom omogočala opraviti Turingov preizkus do leta 2029. Do zgodnjih 2030 naj bi količina nebiološkega računanja prekoračila zmožnost vse žive biološke inteligence človeštva. Končno eksponentno povečanje v računski zmožnosti pravi, da bi privedlo do dogodka singularnosti – močno in moteče preoblikovanje v človeški sposobnosti – leta 2045.

7 Literatura

[1] A. L. Samuel, *Some Studies In Machine Learning Using the Game of Checkers*, IBM Journal on Research and Development, 1959. *(Citirano na strani 2.)*

[2] A. Persson, *Using Temporal Difference Methods In Combination With Artificial Neural Networks to Solve Strategic Control Problems*, KTH Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden, 2004. *(Citirano na straneh 2 in 23.)*

[3] C. Balkenius, J. Morén, *Computational Models of Classical Conditioning: A Comparative Study*, From animals to animats 5: proceedings of the fifth international conference on simulation of adaptive behavior. MIT Press/Bradford Books: Cambridge, MA, 1998. *(Citirano na strani 4.)*

[4] C. E. Shannon, *A Mathematical Theory of Communication*, Bell Sys. Tech. Journal, vol. 27, 1948. *(Citirano na strani 2.)*

[5] C. Stangor, *Introduction to Psychology*, MIT Press, Cambridge, MA, 2011. *(Citirano na straneh VIII, 4, 5, 6 in 7.)*

[6] D. J. C. MacKay *Introduction to Monte Carlo Methods*, Learning in graphical models. Springer Netherlands, 1998. *(Citirano na strani 18.)*

[7] D. Porter, A. Neuringer, *Music Discrimination By Pigeons*, Journal of Experimental Psychology: Animal Behavior Processes 10.2, 1984. *(Citirano na strani 7.)*

[8] E. L. Thorndike, *Animal Intelligence: An Experimental Study of the Associative Processes In Animals*, Psychological Monographs: General and Applied 2.4, 1898. *(Citirano na strani 5.)*

[9] E. L. Thorndike, *Animal Intelligence: Experimental Studies*, The Journal of Nervous and Mental Disease 39.5, 1912. *(Citirano na strani 5.)*

[10] G. Markkula, *Playing risk aversive go on a large board using local neural network position evaluation functions*, Department of physical resource theory and complex systems group, Chalmers university of technology Göteborg, 2004. *(Citirano na strani 2.)*

[11] G. Tesauro, *Practical issues in temporal difference learning*, Machine Learning 4, 1992. *(Citirano na straneh 2 in 24.)*

[12] H. R. Maei, R. S. Sutton, *GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces*, Proceedings of the Third Conference on Artificial General Intelligence, Vol. 1, 2010. *(Citirano na strani 40.)*

[13] I. Pavlov, *Conditioned Reflexes*, Courier Dover Publications, 2003. *(Citirano na strani 3.)*

[14] J. Boyan, A. W. Moore, *Generalization in Reinforcement Learning: Safely Approximating the Value Function*, Advances in neural information processing systems, 1995. *(Citirano na strani 24.)*

[15] J. N. Tsitsiklis, B. Van Roy, *An analysis of temporal-difference learning with function approximation*, Automatic Control, IEEE Transactions on 42.5, 1997. *(Citirano na strani 24.)*

[16] K. Hornik, M. Stinchcombe, H. White, *Multilayer feedforward networks are universal approximators*, Neural networks 2.5, 1989. *(Citirano na strani 23.)*

[17] R. A. Rescorla, *Pavlovian Conditioning: It's Not What You Think It Is*, American Psychologist, 1988. *(Citirano na strani 3.)*

[18] R. Bellman, *On the Theory of Dynamic Programming*, Proceedings of the National Academy of Sciences of the United States of America 38.8 1952. *(Citirano na strani 15.)*

[19] R. Kurzweil, *The Singularity Is Near: When Humans Transcend Biology*, Penguin, 2005. *(Citirano na strani 40.)*

[20] R. S. Sutton, *Learning to predict by the methods of temporal difference*, Machine Learning 3, 1988. *(Citirano na strani 2.)*

[21] R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998. *(Citirano na straneh VIII, 2, 4, 8, 9, 14, 18, 20, 28 in 39.)*

[22] M. E. Bouton, *Learning and Behavior: A Contemporary Synthesis*, Sinauer Associates, 2007. *(Citirano na strani 3.)*

[23] M. Ito, T. Yoshioka, S. Ishii, *Strategy acquisition for the game "Othello" based on reinforcement learning*, Technical report, Nara Institute of Science and Techology, 1998. *(Citirano na strani 2.)*

[24] M. L. Minsky, S. A. Papert, *Perceptrons - Expanded Edition: An Introduction to Computational Geometry*, Boston, MA:: MIT press, 1987. *(Citirano na strani 23.)*

[25] N. N. Schraudolf, P. Dayan, T. Sejnowski, *Learning to evaluate go positions via temporal difference methods*, Vol. 62 of Studies in fuzziness and soft computing, Springer Verlag, 2001. *(Citirano na strani 2.)*

[26] S. Bhatnagar, D. Precup, D. Silver, R. S. Sutton, H. R. Maei, C. Szepesvári, *Convergent temporal-difference learning with arbitrary smooth function approximation*, Advances in Neural Information Processing Systems, 2009. *(Citirano na strani 40.)*

[27] S. B. Thrun, *The Role of Exploration in Learning Control*, Department of Computer Science, Carnegie-Mellon University, 1992. *(Citirano na strani 14.)*

[28] S. Even, R. E. Tarjan, *A combinatorial problem which is complete in polynomial space*, Journal of the ACM (JACM) 23.4, 1976. *(Citirano na strani 25.)*

[29] S. J. Shettleworth, *Cognition, Evolution and Behavior*, Oxford University Press, 2009. *(Citirano na strani 3.)*

[30] S. Legg, M. Hutter, *A Collection of Definitions of Intelligence*, Frontiers in Aritificial Intelligence and Applications, 2007. *(Citirano na strani 1.)*

[31] S. P. Singh, R. S. Sutton, *Reinforcement Learning with Replacing Eligibility Traces*, Machine learning 22.1-3, 1996. *(Citirano na straneh 16 in 20.)*

[32] S. P. Singh, T. Jaakkola, M. I. Jordan, *Learning Without State-Estimation in Partially Observable Markovian Decision Processes*, ICML, 1994. *(Citirano na straneh 19 in 20.)*

[33] S. Watanabe, J. Sakamoto, M. Wakita, *Pigeons' Discrimination of Paintings by Monet and Picasso*, Journal of the experimental analysis of behavior 63.2, 1995. *(Citirano na strani 7.)*

[34] T. L. Brink, *Psychology: A Student Friendly Approach*, San Bernardino Community College, 2008. *(Citirano na strani 3.)*

[35] T. Maarup, *Everything you always wanted to know about Hex but were afraid to ask*, Diss. Master's thesis, University of Southern Denmark, 2005. *(Citirano na strani 25.)*

[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, *Playing Atari with Deep Reinforcement Learning*, arXiv preprint arXiv:1312.5602, 2013. *(Citirano na strani 40.)*

[37] X. Glorot, Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, International Conference on Artificial Intelligence and Statistics, 2010. *(Citirano na strani 23.)*