ADAPTIVNI SISTEMI MAŠINSKOG UČENJA

Jaroslav E. Poliščuk[1]

Ključne riječi: *Učenje sa ojačanjem, Markovljev proces odlučivanja, Belmanova jednačina, učenje trenutnih razlika, tragovi aktivnosti, algoritam TR*(0), *algoritam TR*(*Lambda*).

SAŽETAK:
U radu je analizirana metoda mašinskog učenja sa ojačanjem, kod koje se definiše predmet učenja. Suština ove metode je biranje akcija postupkom probe i greške i dodjela odložene nagrade. Ako stanje okruženja posjeduje Markovljevu osobinu, onda dinamika "jednog koraka" omogućava predviđanje slijedećeg stanja i slijedeće nagrade na osnovu poznatog, trenutnog, stanja i akcije, odnosno provođenje Markovljevog procesa odlučivanja. Relacija između vrijednosti trenutnog stanja i vrijednosti mogućeg slijedećeg stanja je definisana Belmanovom jednačinom. Diskutovana je metoda učenja trenutnih razlika, mehanizam tragova aktivnosti, kao i njihovi algoritmi TR(0) i TR(Lambda). Teorijska razmatranja su ilustrovana praktičnim istraživanjima, odnosno implementacijom algoritma Sarsa(Lambda), sa jediničnim tragovima aktivnosti i Epsilon gramzivom politikom.

1. UVOD

Mogućnost prilagođavanja novim i nepoznatim situacijama, kao i situacijama koje se brzo mijenjaju, po pravilu, imaju samo inteligentni sistemi. Drugim riječima, inteligencija je sposobnost sistema da se prilagodi promjenama u okruženju i što je ta sposobnost veća sistem je inteligentniji.

Prilagođavanje je proces kroz koji sistem unapređuje svoje mogućnosti. Taj proces, u zavisnosti od vrste prilagođavanja, omogućavaju različite karakteristike inteligencije, u koje, prema uobičajenoj klasifikaciji, spadaju: imitacije dijaloga, rješavanje svih varijanti problema, rješavanje netrivijalnih zadataka, ekstrapolacija i učenje.

Računari predstavljaju jedinu tvorevinu ljudske ruke koji su u stanju da pretenduju na mogućnost sticanja elemenata i karakteristika inteligencije. Pri tome treba zanemariti sociološka, kulturološka, psihološka i druga razmatranja i sagledavati tehničke pretpostavke za ostvarenje navedenog cilja. Karakteristike tako nastale vještačke inteligencije bi bile:

[1]Elektrotehnički fakultet, Univerzitet Crne Gore, Podgorica.

prijem, spremanje i brzina obrade podataka, efikasnost i promjenljivost računarskih programa, mogućnost učenja, ekstrapolacija i rješavanje netrivijalnih zadataka.

Sistemi vještačke inteligencije treba da rade u kompleksnom okruženju koje je teško, veoma često i nemoguće, u potpunosti definisati i modelirati. Kao primjeri se mogu navesti: vožnja automobila, kontrola leta, upravljanje složenim tehnološkim procesima, itd. U ovakvim slučajevima standardne metode programiranja daju, po pravilu, aproksimativna i parcijalna rješenja, koja su često i neadekvatna. Mašinsko učenje treba da daje drugačiji pristup rješavanju problema, tj. da omogući sistemu da nauči kako da riješi neki problem, a zatim da ga rutinski rješava.

Za računarski program, mašinu ili sistem se može reći da uči ako kroz iskustvo poboljšava svoje performanse u rješavanju nekog zadatka ili nekog skupa zadataka u datom okruženju. Pod poboljšanjem performansi se podrazumjeva, mada ga u cjelosti ne sačinjava, prikupljanje i proširivanje znanja. Sistem treba biti sposoban da usvaja nova znanja i da ta znanja efikasno koristi. Mora biti definisan zajedno sa svojim okruženjem i dovoljno robusan da prihvati njegove dinamičke promjene. Ovakve zahtjeve mogu da ispune samo sistemi koji proces učenja baziraju na interakciji učenika sa okruženjem.

U radu je analizirana metoda *mašinskog učenja sa ojačanjem (reinforcement learning)* [1], [2], [3], [7], [8], koja razmatra učenje kroz interakciju određenu zadanim ciljem učenika. Ova metoda se razlikuje od klasičnih metoda *učenja sa supervizorom*, kod kojih se učeniku eksplicitno kaže šta treba da radi u svakom stanju okruženja.

Suština ideje učenja sa ojačanjem se ogleda u uvođenju principa nagrađivanja učenika, odnosno učeniku se vraća signal nagrade dovođenjem sistema u određeno stanje, tj. njegovo učenje se ojačava. Pri tome se ne govori koje akcije treba preduzeti, već se dozvoljava da on sam otkrije one koje mu donose najveću nagradu. U najizazovnijim slučajevima akcije ne utiču samo na trenutnu nagradu, već kroz slijedeće situacije i na sve buduće nagrade. Navedene dvije karakteristike, biranje akcija postupkom probe i greške i dodjela odložene nagrade, predstavljaju najvažnije osobine učenja sa ojačanjem.

Kod učenja sa ojačanjem se ne definiše algoritam učenja, nego predmet učenja. To znači da će bilo koji algoritam napravljen da riješi dati problem biti algoritam učenja sa ojačanjem. *Agent* (agent, širi pojam od pojma učenika, jer osim što uči ima zadatak i da donosi odluke) treba da sagleda najvažnije aspekte realnog problema sa kojim se suočava u toku postizanja zadanog cilja.

Agent mora da:

- sazna *stanje* okruženja,
- preduzima *akcije* koje će uticati na to stanje,
- ima zadani *cilj* ili *ciljeve* koji su usko povezani sa stanjem okruženja.

Osim agenta i okruženja mogu se identifikovati i četiri podelementa učenja sa ojačanjem, a to su [3], [7]:

- *politika*, koja preslikava sagledano stanje okruženja u akciju koja će biti izvedena u tom stanju i koja definiše način ponašanja agenta;
- *funkcija nagrade* definiše cilj koji agent treba da postigne. Ukratko, ova funkcija preslikava sagledano, trenutno, stanje okruženja u jedan skalar, nagradu, koja agentu pruža saznanje o valjanosti tog stanja;

- *vrijednosna funkcija* određuje dugotrajnu valjanost određenog stanja okruženja, uzimajući u obzir stanja koja ga naslijeđuju i moguće nagrade u tim stanjima i, opciono,
- *model okruženja*, odnosno element koji na izvjestan način oponaša okruženje, omogućava planiranje, gdje se pod planiranjem podrazumjeva odlučivanje o akciji uzimajući u obzir moguća, buduća, stanja prije nego se ona dostignu.

2. OKRUŽENJE SISTEMA

Agent je u neprestalnoj interakciji sa okruženjem, slika 1. Kompletna specifikacija okruženja definiše *zadatak*, koji je instanca problema učenja sa ojačanjem.

Vremenska osa interakcije je diskretna, tj. agent i njegovo okruženje međusobno komuniciraju u diskretnim vremenskim trenucima $t = 0, 1, 2, \ldots$. U svakom trenutku t agent prima prikaz stanja u kome se nalazi okruženje, $s_t \in S$, gdje je S skup svih mogućih stanja okruženja. Na isti način se odabira akcija $a_t \in A(s_t)$, gdje je $A(s_t)$ skup svih mogućih akcija u tom trenutku. U slijedećem vremenskom intervalu $t+1$ okruženje mijenja svoje stanje u s_{t+1}, kao posljedicu izvedene akcije, i agentu šalje numeričku nagradu $r_{t+1} \in R$.

U svakom koraku agent vrši odabiranje akcije na osnovu politike π_t, gdje je $\pi_t(s,a)$ vjerovatnoća da je, za stanje okruženja $s_t = s$, akcija $a_t = a$. Algoritmi učenja sa ojačanjem prikazuju kako agent mijenja svoju politiku sa sticanjem iskustva. Potrebno je naglasiti da je osnovni cilj agenta da maksimizira primljene nagrade u dužem vremenskom periodu.

Granica između agenta i okruženja se postavlja na osnovu opšteg pravila da je sve, na šta agent ne može direktno uticati, dio okruženja. U praksi se ova granica postavlja nakon što su određena stanja, akcije i nagrade, odnosno kada je definisan zadatak.

3. NAGRAĐIVANJE I OSOBINA NEZAVISNOSTI PUTA

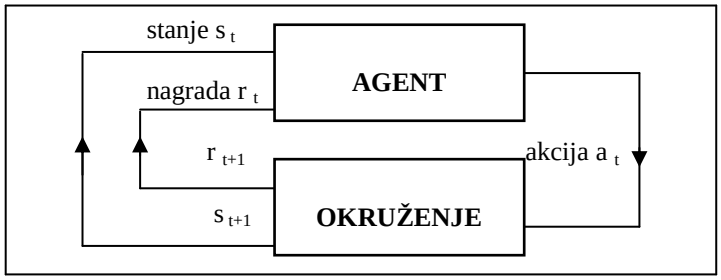
Agent tokom vremena biva nagrađivan za svoje akcije, uz osnovnu težnju da ukupni iznos nagrade učini maksimalnim. Nagrada je skalarna veličina, odnosno broj, čija vrijednost varira u svakom koraku interakcije između agenta i okruženja. Navedeni smisao nagrade određuje mjesto njenog izračunavanja, a to je okruženje i agent ne smije da ima uticaj na njeno određivanje.

Pretpostavka je da je agent nakon trenutka t primio slijedeću sekvencu nagrada: r_{t+1}, r_{t+2}, r_{t+3}, ... i želi da maksimizira *ukupno očekivanu nagradu* R_t. Ukupno očekivana nagrada je funkcija navedene sekvence i u najjednostavnijem slučaju je zbir:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_T, \tag{1}$$

gdje je T posljednji vremenski interval.

Ovakav pristup ima smisla samo u problemima koji po svojoj prirodi imaju definisan posljednji vremenski interval, odnosno kada se interakcija između agenta i okruženja razbija u epizode, kao npr. u igrama. Svaka epizoda ima konačni završetak, nakon čega dolazi povratak u početno stanje. Zadaci koji imaju epizode u navedenom smislu su *epizodni zadaci*.



Slika 1. Interakcija između agenta i okruženja.

Drugi tip zadataka su *kontinualni zadaci*. Primjeri ove vrste zadataka su kontrole procesa. Funkcija očekivane ukupne nagrade se u ovom slučaju izračunava koristeći koncept *faktora popusta* na slijedeći način:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{2}$$

gdje je γ faktor popusta i $0 \le \gamma \le 1$.

Faktor popusta određuje vrijednost koju imaju buduće nagrade u sadašnjem trenutku, odnosno nagrada dobijena u *k*-tom koraku vrijedi γ^{k+1} puta manje nego da je primljena u ovom trenutku.

Okruženje u svakom trenutku predstavlja agentu svoje stanje, koji na osnovu tog skupa informacija donosi potrebne odluke. Idealni signal stanja bi, osim trenutnog snimka, morao da omogući uvid i u prethodna stanja, bez da degradira sposobnost agenta da predvidi buduće ponašanje okruženja. Za ovakav signal se može reći da ima *Markovljevu osobinu* ili *osobinu nezavisnosti puta*, jer su, na određeni način, sve relevantne informacije sadržane u trenutnom signalu stanja.

Na pitanje kako okruženje u trenutku *t+1* može odgovoriti na akciju preduzetu u trenutku *t*, u najopštijem, kauzalnom, slučaju slijedi zaključak da odgovor zavisi od svega što se desilo prije trenutka *t+1*.

Na osnovu navedenog može se reći da stanje posjeduje Markovljevu osobinu ako je vjerovatnoća prelaska iz jednog stanja u drugo [3], [4], [7]:

$$\begin{aligned} &Vjer.\{s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0|\} = \\ &= Vjer.\{s_{t+1} = s', r_{t+1} = r | s_t, a_t|\} \end{aligned} \tag{3}$$

za svako *s´, r* i za sve moguće vrijednosti prethodnih događaja.

Ako okruženje, odnosno njegovo stanje, posjeduje Markovljevu osobinu, onda takva dinamika "jednog koraka" omogućava da se predvidi slijedeće stanje i slijedeća nagrada na osnovu poznatog, trenutnog, stanja i akcije.

Učenje sa ojačanjem, koje zadovoljava Markovljevu osobinu, poznato je u literaturi kao *Markovljev proces odlučivanja (Markov decision process)* [4], [7]. Ako su prostori stanja i akcija konačni i određeni dinamikom "jednog koraka" datog okruženja, onda je takav Markovljev proces odlučivanja (MPO) *konačni MPO*.

Za proizvoljno dato stanje *s* i akciju *a* vjerovatnoća mogućeg slijedećeg stanja *s´* je:

$$P_{ss'}^{a} = Vjer.\{s_{t+1} = s' | s_t = s, a_t = a\}. \quad (4)$$

Takođe, za dato trenutno stanje *s* i akciju *a*, zajedno sa bilo kojim slijedećim stanjem *s´*, očekivana vrijednost nagrade *E* je:

$$R_{ss'}^{a} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\}. \quad (5)$$

Ove dvije vrijednosti, $P_{ss'}^{a}$ i $R_{ss'}^{a}$, u potpunosti određuju najvažnije osobine dinamike konačnog MPO.

4. FUNKCIJE VRIJEDNOSTI STANJA I AKCIJA I NJIHOVA OPTIMIZACIJA

Funkcije vrijednosti su definisane u odnosu na određenu politiku koju agent prati. Politika π predstavlja preslikavanje stanja $s \in S$ i akcija $a \in A(s)$ u vjerovatnoću $\pi(s,a)$ odabiranja akcija *a* kada se okruženje nalazi u stanju *s*.

Drugim riječima, pod *funkcijom vrijednosti stanja za politiku* π, oznaka $V^{\pi}(s)$, se podrazumjeva ukupna vrijednost nagrade koju očekuje agent polazeći od stanja *s* i doslijedno prateći politiku π.

Za MPO funkcija vrijednosti stanja $V^{\pi}(s)$ se definiše:

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\}, \quad (6)$$

gdje E_{π} označava očekivanu vrijednost nagrade kada agent prati politiku π.

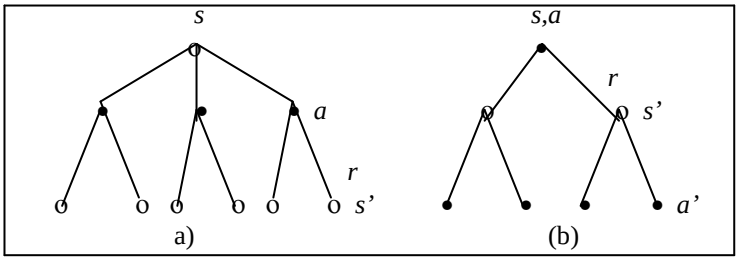
Na sličan način se definiše i *funkcija vrijednosti akcije za politiku* π:

$$Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\} \quad (7)$$

Osnovna osobina funkcija vrijednosti je da one zadovoljavaju slijedeću rekurentnu relaciju. Za bilo koju politiku π i bilo koje stanje *s* postoji relacija između vrijednosti trenutnog stanja *s* i vrijednosti mogućeg slijedećeg stanja, odnosno:

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\{ R_t | s_t = s \} \\ &= E_{\pi}\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \} \\ &= E_{\pi}\{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s \} \\ &= \sum_{a} \pi(s,a) \sum_{s'} P_{ss'}^{a} [R_{ss'}^{a} + \gamma E_{\pi}\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s' \}] \\ &= \sum_{a} \pi(s,a) \sum_{s'} P_{ss'}^{a} [R_{ss'}^{a} + \gamma V^{\pi}(s')] \end{aligned} \quad (8)$$

Navedena jednačina je *Belmanova jednačina (Bellman equation)* za funkciju vrijednosti stanja [2]. Predstavlja odnos između vrijednosti stanja i vrijednosti naslijednika tog stanja, što se može predstaviti tzv. *"backup"* dijagramima, slika 2.



Slika 2. *"Backup"* dijagrami: (a) za funkciju *V*, (b) za funkciju *Q*.

Politika π je jednaka ili bolja od politike π' ako joj je ukupna očekivana nagrada jednaka ili veća. Tačnije, $\pi \geq \pi'$ ako je $V^{\pi}(s) \geq V^{\pi'}(s)$ za svako $s \in S$. Uvijek postoji bar jedna politika koja zadovoljava ovaj uslov i ta politika se zove *optimalna politika*, a označava sa π^*. U istom trenutku može da postoji više od jedne optimalne politike, međutim sve one dijele samo jednu *optimalnu funkciju vrijednosti stanja* sa oznakom *V**:

$$V^*(s) = \max_{\pi} V^{\pi}(s) \tag{9}$$

za svako $s \in S$.

Isto vrijedi i za *optimalnu funkciju vrijednosti akcije* sa oznakom *Q**:

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \tag{10}$$

za svako $s \in S$ i $a \in A(s)$.

Za svaki par *stanje – akcija (s,a)* navedena funkcija daje ukupno očekivanu nagradu za izvršavanje akcije *a* u stanju *s* i, nakon toga, podvrgavanje optimalnoj politici. Iz tog razloga optimalna funkcija vrijednosti akcije se može definisati optimalnom funkcijom vrijednosti stanja:

$$Q^*(s,a) = E\{\ r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\ \}. \tag{11}$$

Belmanove jednačine za optimalne funkcije vrijednosti dobijaju slijedeći oblik:

$$V^*(s) = \max_{a \in A(s)} \sum P^a_{ss'} [\ R^a_{ss'} + \gamma V^*(s')\ ] \tag{12}$$

$$Q^*(s) = \sum P^a_{ss'} [\ R^a_{ss'} + \gamma \max_{a'} Q^*(s', a')\ ] \tag{13}$$

Prednost ovih jednačina je u jedinstvenom rješenju i u jednostavnijem određivanju optimalne politike.

Korisno je analizirati osnovne algoritme koji donose odluke i koncepte koji to omogućavaju, polazeći od praktičnih rezultata, bez ulaženja u teorijske rasprave o konvergentnosti poznatih metoda i njihovoj izračunljivosti, što će biti i urađeno.

Izračunavanje optimalnih funkcija vrijednosti se izvodi na osnovu Belmanovih jednačina optimalnosti (12) i (13). Pri tome se javljaju dva fundamentalna ograničavajuća faktora:

- faktor potrebnog vremena i
- faktor potrebnog prostora.

Faktor potrebnog vremena se odnosi na vrijeme koje je potrebno da se izvrši proračun, dok se faktor potrebnog prostora odnosi na memorijski prostor neophodan pri tom izračunavanju. Ukoliko je *n* broj mogućih stanja i *m* broj mogućih akcija u tim stanjima, onda Belmanove jednačine definišu sistem od *nm* nelinearnih jednačina sa *nm* nepoznatih veličina. Za imalo kompleksnije probleme, brojevi *n* i *m* mogu biti veliki, pa je neposredno rješavanje ovog sistema nekom od poznatih metoda praktično neprihvatljivo. Iz navedenih razloga se pribjegava aproksimativnom načinu rješavanja.

Kao iterativna metoda za mašinsko učenje sa ojačanjem se može koristiti *metoda učenja trenutnih razlika (Temporal difference learning),* odnosno *TR metoda (TD Learning)* [6]. Ova metoda na prihvatljiv način rješava problem faktora vremena, međutim i ova metoda koristi bekap mehanizam, pa je problem faktora prostora još uvijek prisutan.

Sve iterativne metode za određivanje politike se sastoje od dva simultana iterativna procesa:

- prvi vrši iteraciju funkcije vrijednosti konzistentno sa trenutnom politikom – evaluacija politike, dok
- drugi čini politiku gramzivom u odnosu na trenutnu funkciju vrijednosti – poboljšanje politike.

Politika je gramziva ako je:

$$\pi'(s) = \arg_a \max Q^{\pi}(s,a), \tag{14}$$

gdje $\arg_a\max$ notacija označava da je za akciju *a* navedeni izraz maksimalan.

Agent se u svakom trenutku suočava sa problemom kako da na osnovu stanja koje mu je na raspolaganju odredi optimalnu politiku, odnosno odabere optimalnu akciju iz skupa mogućih akcija.

Prva metoda je jednostavno odabiranje akcije koja ima najveću vrijednost. Vrijednost akcije se određuje procjenom kroz iterativni postupak. Ova metoda se zove *gramzivom*, jer prati gramzivu politiku, i ima svojstvo isključive eksploatacije budući da ne dovodi do novih saznanja.

Malim proširenjem navedena metoda se može obogatiti određenim nivoom istraživanja. Intervencija se sastoji u tome da sa određenom, malom, vjerovatnoćom ε, umjesto da se odabere najbolja akcija, "slučajno" odabere neka druga. Ova metoda se često koristi u praksi i zove se ε - *gramzivom*. Biranje se vrši po zakonu uniformne raspodjele vjerovatnoće, što u nekim slučajevima može dovesti do nezadovoljavajućih rezultata.

Rješenje navedenih nedostataka je da se pri odabiranju akcija primjeni drugačiji zakon raspodjele. Metode koje koriste drugačiji zakon raspodjele su *softmax metode*, a najčešće su zasnovane na Bolcmanovom zakonu raspodjele:

$$Vjer.(s,a^*) = \frac{e^{Q(s,a^*)/T}}{\sum_{a\in A} e^{Q(s,a)/T}}, \tag{15}$$

gdje se parametar $T > 0$ zove *temperatura* i on, indirektno, kontroliše nivo istraživanja. Kada $T \to 0$ softmax metoda se izjednačava sa gramzivom, dok za visoke vrijednosti temperature istraživanje postaje dominantno. Sofisticiraniji, heuristički algoritmi, osim što su znatno kompleksniji, utiču i na određene promjene arhitekture učenja [5].

5. ALGORITMI METODE TRENUTNIH RAZLIKA

Osnovna osobina metode trenutnih razlika je iterativnost postupka procjene funkcija vrijednosti. Iterativni postupak prati ideju generalnog procesa iteracije politike, a proceduralno se sastoji od dva dijela. U prvom dijelu se vrši inicijalizacija aproksimativnih proizvoljno odabranih funkcija vrijednosti, dok se u drugom dijelu, nakon ažuriranja procjenjenih vrijednosti signalom greške, aproksimativna vrijednost približava stvarnoj. Ovo se može napisati na slijedeći način:

NovaProcjena ← *StaraProcjena* + *korak**(*Cilj* – *StaraProcjena*),

gdje je *(Cilj – StaraProcjena)* greška procjene.

Pravilo po kome se vrši ažuriranje vrijednosti neposredno proizilazi iz Belmanovih jednačina. TR(0) je najjednostavniji TR algoritam i njegovo pravilo ažuriranja je:

$$V(s_t) \leftarrow V(s_t) + \alpha\left[\ r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\ \right] \tag{16}$$

Izraz $r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ se zove *TR greška.*

Zapis algoritma u pseudo kodu je:

INICIJALIZIRAJ *V(s),* π
PONAVLJAJ *(*za svaku epizodu*)*
    INICIJALIZIRAJ *s*
    PONAVLJAJ *(*za svaki korak u epizodi*)*
        a ← *akcija određena politikom* π *za s*
        IZVEDI AKCIJU *a,* ***OSMOTRI NAGRADU*** *r i slijedeće stanje s'*
        $V(s) \leftarrow V(s) + \alpha\,[r + \gamma(s') - V(s)]$
        $s \leftarrow s'$
    DO POSLJEDNJEG STANJA *s*
KRAJ

U opštem slučaju postoje dvije varijante metode trenutnih razlika [1], [3], [7]. Prva varijanta unapređuje onu politiku koju prati pri selekciji akcija. Najznačajniji predstavnik ove varijante je algoritam Sarsa. Druga varijanta prati jednu politiku pri selekciji akcija, najčešće gramzivu, a poboljšava drugu politiku, onu koja je naslijeđuje. Predstavnik ove varijante je Q algoritam.

Algoritam Sarsa ne koristi funkciju vrijednosti stanja *V(s)* u iterativnom procesu, nego funkciju vrijednosti akcija *Q(s,a)*. Međutim, pravilo ažuriranja je veoma slično prethodnom, obzirom na izraz (11), i ima oblik:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[\ r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\ \right] \tag{17}$$

Zapis algoritma je:

INICIJALIZIRAJ *Q(s,a)*
PONAVLJAJ *(*za svaku epizodu*)*
INICIJALIZIRAJ *s*

a ← *akcija određena politikom* π *za s*
PONAVLJAJ *(*za svaki korak u epizodi*)*
IZVEDI AKCIJU *a*, ***OSMOTRI NAGRADU*** *r i slijedeće stanje s'*
a' ← *akcija određena politikom* π *za s'*
$Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma Q(s',a') - Q(s,a)]$
$s \leftarrow s'$, $a \leftarrow a'$
DO POSLJEDNJEG STANJA *s*
KRAJ

Algoritam Sarsa konvergira sa vjerovatnoćom jedan optimalnoj politici i optimalnoj funkciji vrijednosti akcija, ako su svi parovi *stanje – akcija* posjećeni beskonačan broj puta i ako politika konvergira gramzivoj politici. Ovo se postiže korišćenjem npr. ε - gramzive politike, gdje je $\varepsilon = 1/t$.

Q algoritam predstavlja izuzetan iskorak u mašinskom učenju sa ojačanjem. Ovaj algoritam neposredno aproksimira optimalnu funkciju vrijednosti akcija, nezavisno od praćene politike. Njegovo pravilo ažuriranja je:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[\; r_{t+1} + \gamma \max_a Q(s_{t+1},a_t) - Q(s_t,a_t) \;\right] \qquad (18)$$

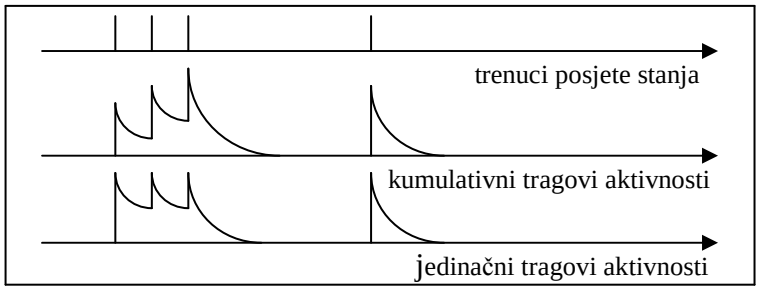
Zapis algoritma je:

INICIJALIZIRAJ *Q(s,a)*
PONAVLJAJ *(*za svaku epizodu*)*
INICIJALIZIRAJ *s*
PONAVLJAJ *(*za svaki korak u epizodi*)*
a ← *akcija određena politikom* π *za s*
IZVEDI AKCIJU *a*, ***OSMOTRI NAGRADU*** *r i slijedeće stanje s'*
$Q(s,a) \leftarrow Q(s,a) + \alpha\left[\; r + \gamma \max_{a'} Q(s',a') - Q(s,a) \;\right]$
$s \leftarrow s'$
DO POSLJEDNJEG STANJA *s*
KRAJ

Jedini uslov za konvergenciju Q algoritma je da svi posjećeni parovi *uslov – akcija* budu korektno ažurirani, što je, inače, zahtjev za sve metode mašinskog učenja sa ojačanjem.

6. MEHANIZMI TRAGOVA AKTIVNOSTI

Mehanizmi tragova aktivnosti *(activity traces, eligibility traces)* [2] su jedan od osnovnih mehanizama mašinskog učenja sa ojačanjem. Ideja je da se za svako stanje odvoji memorijska lokacija u kojoj će biti vođena "statistika" posjećenosti tog stanja. Svaki put kada je stanje posjećeno, njegova aktivnost postaje visoka, a zatim opada sve do nove posjete. Pravilo ažuriranja tragova aktivnosti, koje je označeno sa $e_t(s)$, je:



Slika 3. Tragovi aktivnosti.

$$e_t(s) = \begin{cases} \gamma\lambda_{t-1}(s) & ako\ \ je\ \ s \neq s_t; \\ \gamma\lambda_{t-1}(s) + 1 & ako\ \ je\ \ s = s_t, \end{cases} \tag{19}$$

gdje je $0 \leq \lambda \leq 1$ *parametar opadanja aktivnosti*. Uvođenje tragova aktivnosti u do sada navedene algoritme povećava efikasnost učenja.

Tragovi aktivnosti, koji se formiraju po pravilu (19), su *kumulativni tragovi aktivnosti*, za razliku od *jedinačnih tragova aktivnosti*, koji predstavljaju njihovu modifikaciju. Pravilo za jedinačne tragove aktivnosti je:

$$e_t(s) = \begin{cases} \gamma\lambda_{t-1}(s) & ako\ \ je\ \ s \neq s_t; \\ 1 & ako\ \ je\ \ s = s_t. \end{cases} \tag{20}$$

Ove dvije vrste tragova aktivnosti su prikazani na slici 3.

Tragovi aktivnosti se jednostavno implementiraju u osnovne TR metode, sa oznakom *TR(λ) metode*. TR(λ) metode su uopšteni oblik TR metoda, jer npr. za $\lambda = 0$ metode se svode na osnovne.

Slijedi prikaz modifikovanih pravila ažuriranja i modifikovanih algoritama.

Algoritam TR(λ) ima slijedeće pravilo ažuriranja:

$$V(s_t) \leftarrow V(s_t) + \alpha\delta e(s_t), \tag{21}$$

gdje izraz: $\delta = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ predstavlja TR grešku.

Zapis algoritma je:

INICIJALIZIRAJ *V(s) i e(s) = 0 za svako* $s \in S$
PONAVLJAJ *(*za svaku epizodu*)*
INICIJALIZIRAJ *s*
    PONAVLJAJ *(*za svaki korak u epizodi*)*
  $a \leftarrow$ *akcija određena politikom* π *za s*
  IZVEDI AKCIJU *a,* ***OSMOTRI NAGRADU*** *r i slijedeće stanje s'*
    $\delta \leftarrow r + \gamma V(s') - V(s)$
  $e(s) \leftarrow e(s) + 1$
  Za svako s:
    $V(s) \leftarrow V(s) + \alpha\delta e(s)$

e(s) ← $\gamma\lambda e(s)$
s ← *s'*
DO POSLJEDNJEG STANJA *s*
KRAJ

Algoritam Sarsa(λ*)* ima pravilo ažuriranja:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\delta e(s_t, a_t), \tag{22}$$

gdje izraz $\delta = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$ predstavlja TR grešku.

Zapis algoritma je:

INICIJALIZIRAJ *Q(s,a) i e(s, a) = 0 za svako s* ∈ *S, a*∈ *A(s)*
PONAVLJAJ *(*za svaku epizodu*)*
INICIJALIZIRAJ *s*
a ← *akcija određena politikom* π *za s*
PONAVLJAJ *(*za svaki korak u epizodi*)*
IZVEDI AKCIJU *a,* ***OSMOTRI NAGRADU*** *r i slijedeće stanje s'*
a'← *akcija određena politikom* π *za s'*
$\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
e(s, a) ← *e(s, a) + 1*
Za svako s, a:
Q(s, a) ← *Q(s, a) +* $\alpha\delta e(s, a)$
e(s, a) ← $\gamma\lambda e(s, a)$
s ← *s', a* ← *a'*
DO POSLJEDNJEG STANJA *s*
KRAJ

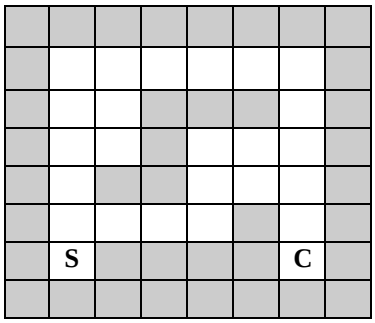
Q(λ*) algoritam* ima slijedeće pravilo ažuriranja:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\delta e(s_t, a_t), \tag{23}$$

gdje izraz $\delta = r_{t+1} + \gamma \max Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$ predstavlja TR grešku.

Zapis algoritma je:

INICIJALIZIRAJ *Q(s, a) i e(s, a) za svako s*∈ *S, a*∈ *A(s)*
PONAVLJAJ *(*za svaku epizodu*)*
INICIJALIZIRAJ *s, a*
PONAVLJAJ *(*za svaki korak u epizodi*)*
a ← *akcija određena politikom* π *za s*
IZVEDI AKCIJU *a,* ***OSMOTRI NAGRADU*** *r i slijedeće stanje s'*
a = arg*$_b$*max Q(s', b)*
$\delta \leftarrow r + \gamma Q(s', a^*) - Q(s, a)$
e(s, a) ← *e(s, a) + 1*



Slika 4. Lavirint sa startnom i ciljnom pozicijom agenta.

Za svako s, a:
Q(s, a)$\leftarrow$ *Q(s, a) +* $\alpha\delta e(s, a)$
ako je a' = a onda je e(s, a)*$\leftarrow$ $\gamma\lambda e(s, a)$
inače e(s, a) $\leftarrow$ *0*
$s \leftarrow s'$, $a \leftarrow a'$
DO POSLJEDNJEG STANJA *s*
KRAJ

Na osnovu dosadašnjeg izlaganja se može zaključiti da su procjene funkcija vrijednosti čuvane u obliku tabele, sa ulazima za svako stanje ili za svaki par *stanje – akcija*. Za slučajeve sa ograničenim i malim brojem stanja i akcija, ovaj način daje zadovoljavajuće rezultate. U suprotnom, javlja se problem ograničenosti memorijskog prostora. Čak, ukoliko bi na raspolaganju bio beskonačni memorijski prostor, problem sa tabelom ostaje zbog potrebnog vremena za nalaženje i pristup potrebnoj lokaciji.

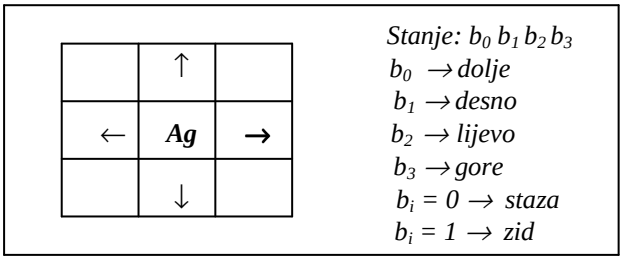
Rješenje navedenog problema je u *generalizaciji* prostora stanja i akcija. Ideja je da se mali podskup prostora stanja i prostora akcija, kroz iskustvo, generalizuje tako da predstavlja znatno veći podskup istog prostora. Mada je ideja generalizacije jednostavna, njena realizacija stvara velike probleme i predstavlja predmet daljnih istraživanja.

Kod sistema mašinskog učenja je najprihvatljivija generalizacija sa nazivom *aproksimacija funkcija*. Ova generalizacija uzima pojedine vrijednosti željene funkcije i pokušava da ih generalizuje tako da predstavljaju aproksimaciju te funkcije. Aproksimacija funkcija predstavlja instancu klase metoda učenja sa supervizorom, a može se primjeniti i kod vještačkih neuro mreža, stabala odlučivanja, prepoznavanja uzoraka, itd.

7. ANALIZA EKSPERIMENTALNIH REZULTATA JEDNOG ILUSTRATIVNOG PRIMJERA MAŠINSKOG UČENJA

Provedena teorijska razmatranja će biti ilustrovana praktičnim istraživanjima. Razmatrani primjer nalaženja puta u lavirintu je trivijalan, ali odražava suštinu analiziranih metoda.

Cilj zadatka je da agent sa startne pozicije, ne znajući izgled lavirinta, stigne na cilj, takođe ne znajući gdje se on nalazi. Da je stigao na cilj saznaje posredno preko signala na-



Slika 5. Načini kodiranja stanja i akcija.

grade. Da bi zadatak bio što ilustrativniji i vrijeme izvršavanja programa što kraće, lavirint, slika 4, je veoma jednostavan.

Nalaženje puta u lavirintu je epizodni zadatak, tj. nakon pronalaženja cilja agent se vraća na startnu poziciju. Sistem nagrađivanja za epizodne zadatke je dat izrazom (1). Međutim, pristup sa faktorom popusta dat izrazom (2) se češće koristi, jer za $\gamma = 1$ se svodi na prvi pristup.

Da bi se implementirao algoritam učenja, mora se odrediti kako će biti kodirani stanje i akcija i na koji način će biti memorisana funkcija vrijednosti i lavirint.

Lavirint se memoriše kao matrica (8×8) nula i jedinica, pri čemu nula označava stazu, a jedinica zid. Stanje se kodira sa četiri bita. Svakom bitu odgovara pozicija u ravni lavirinta i svaki bit označava da li je na toj poziciji prohodan ili neprohodan dio staze (slika 5). Na primjer, kada se agent nađe na poziciji 1000, on može produžiti lijevo, gore ili desno, dok mu se dolje nalazi zid. Ovaj način kodiranja omogućava da broj stanja ne zavisi od veličine lavirinta, nego od načina i broja mogućih akcija. Postoje samo četiri akcije: gore, lijevo, dolje i desno.

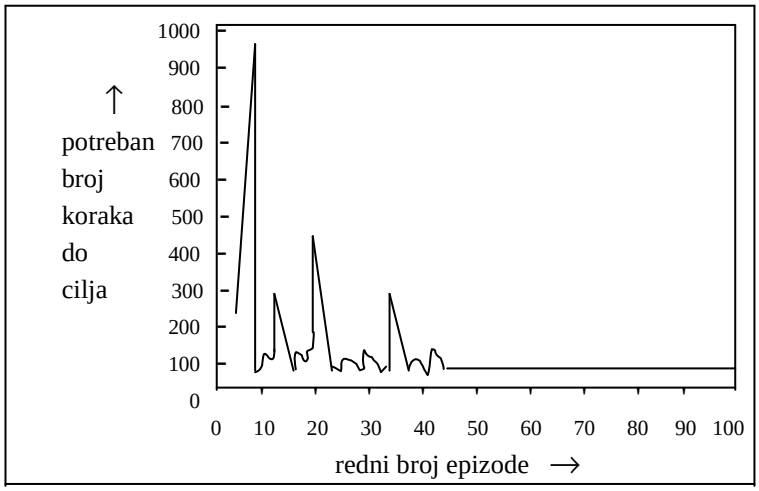
Funkcija vrijednosti je memorijski predstavljena kao tabela u kojoj je broj redova jednak broju stanja, a broj kolona jednak broju akcija.

Implementacija algoritma *Sarsa(λ)*, sa jediničnim tragovima aktivnosti i ε gramzivom politikom, je za razmatrani lavirint urađena pomoću programskog jezika *C*. Dobijeni program sadrži nešto više od sto programskih naredbi i sa tim programom su izvršena potrebna istraživanja. U program je ugrađena logička sekvenca za generisanje datoteke u koju se zapisuju brojevi koraka po epizodama potrebni da se nađe cilj. Rezultat je prikazan na slici 6, a predstavlja grafički prikaz zavisnosti potrebnog broja koraka za nalaženje ciljne pozicije od broja epizoda.

Grafikon prikazuje brzinu kojom se agent približava optimalnoj politici. Najinteresantniji dijelovi grafikona su "špicevi" koji se pojavljuju u određenim epizodama. Ovi "špicevi" su posljedica traženja optimalnog rješenja metodom probe i greške.

Dalja istraživanja potvrđuju pretpostavku da su performanse sistema učenja znatno bolje uvođenjem tragova aktivnosti i parametra opadanja aktivnosti λ. Već je rečeno da se algoritam *Sarsa (λ)* svodi na osnovnu varijantu algoritma za $\lambda = 0$. Na slici 7 su prikazani rezultati izvršavanja programa za tri vrijednosti ovog parametra.

Uočava se da se smanjenjem vrijednosti parametra λ performanse značajno degradiraju. Na osnovu toga se može donijeti zaključak da bi za još manju vrijednost parametra λ, algo-



Slika 6. Zavisnost broja koraka do cilja od broja epizoda.

ritmu bilo potrebno znatno duže vrijeme za konvergenciju. Već za $\lambda=0,65$ agent ne uspjeva da nađe optimalno rješenje u prvih 100 epizoda.

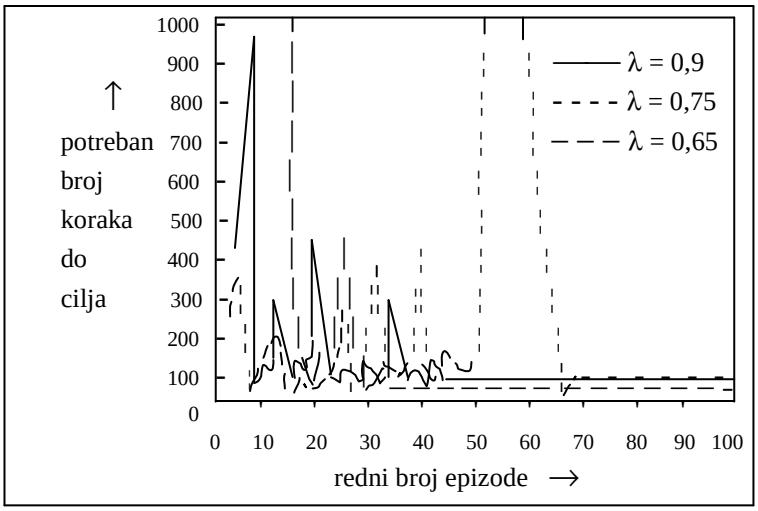
Cilj izvršenih istraživanja je da se učenje sa ojačanjem predstavi na jednostavan način. Prvi eksperiment daje praktični prikaz metoda učenja sa ojačanjem, dok drugi eksperiment prikazuje prednosti pristupa sa tragovima aktivnosti.

8. ZAKLJUČAK

Značajni predstavnik adaptivnih sistema mašinskog učenja je metoda mašinskog učenja sa ojačanjem. Ova metoda svoju popularnost u posljednjih desetak godina duguje razrađenoj teorijskoj osnovi, širokom području primjene i osobini da može da uči bez prethodno pripremljene baze znanja o problemu.

Nedostatak ove metode je relativno sporo učenje i ovaj nedostatak se pokušava otkloniti na nekoliko načina. Jedan od načina je da se sistem obuči na simulaciji problema, pa tek onda primjeni u praksi. Slijedeći načini su da se koriste modeli akcija, modeli planiranja, itd. Osim navedenog, otvoreni problem predstavlja kompromis između istraživanja i eksploatacije, kao i problem aproksimacije funkcija koje određuju obim zadataka na koji se ova metoda može optimalno primjeniti. Parcijalna rješenja navedenih problema već postoje, tako da je realno očekivati njihovo brzo prevazilaženje.

Iako navedeni problemi predstavljaju ograničavajuće faktore, ova se metoda pokazala veoma korisnom gdje god je moguće uvesti sistem nagrađivanja. Najznačajnije aplikacije se očekuju u području industrijske kontrole, autonomnih i mobilnih robota, rješavanju problema optimizacije i raspoređivanja resursa. Prihvatljiva je i ideja da se ova metoda upotrijebi u raznim aspektima ekonomskih i berzanskih predviđanja.



Slika 7. Efikasnost učenja u zavisnosti od parametra λ.

LITERATURA

[1] J.A.Boyan, M.L.Littman: "Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach", *Advances in Neural Information Processing Systems: Proceedings of the 994 Conference, San Francisco*, CA, USA, 1994.

[2] Doya, Kenji: "Reinforcement Learning in Continuous Time and Space", *Neural Computation*, Jan2000, Vol. 12 Issue 1, p219, 27p.

[3] L.P.Kaebling, M.L.Littman, A.W.Moore: "Reinforcement Learning: A Survey", *Journal of Artificial Intelligence*, vol. 4, 1996., pp.237-285.

[4] M.E.Lewis, M.L.Puterman: "A Probabilistic Analysis of Bias Optimality in Unichain Markov Decision Process", IEEE Transactions on Automatic Control, Vol.46 Issue 1, Jan.2001, p96, 5p.

[5] J.E.Poliščuk: "A contribution to methodology of development of Decision Support Systems and Expert Systems", *Doctors Thesis, Faculty of Organization and Informatics, University of Zagreb, Croatia*, 1991.

[6] E.T.Rolls, T.Milward, Wiskott, Laurenz: "A Model of Invoviant Object Recognition in the Visual System: Learning Rules, Activation Functions, Lateral Inhibition, and Information – Based Performance Measures", *Neural Computation*, Vol. 2 Issue 11, Nov.2000, p2547, 26p.

[7] R.S.Sutton, A.G.Barto: "Introduction to Reinforcement Learning", *MIT press – Bradford Books, Cambridge, MA*, 1998.

[8] C.Szepesvari, M.L.Littman: "A Unified Analysis of Value – Function – Based Reinforcement – Learning Algorithms", *Neural Computation*, Vol.11, Issue 8, 11/15/99, p2017, 44p.