Anđela Bogdanović i Jelena Cvetić

Optimizacija hodanja bipedalnog modela korišćenjem učenja podsticajem

Cilj ovog rada je bio da se izvrši optimizacija hoda bipedalnog modela pomoću Reinforcement learning-a, odnosno učenja podsticajem, jedne od vrsta mašinskog učenja. Hodanje bipedalnog modela simulira čovekovo kretanje projektovano u ravni. Model je projektovan uz pomoć Box2D biblioteke, i tokom simulacije može menjati ugao svojih zglobova i ugao pod kojim se nalazi trup. Za učenje hodanja, odnosno kontrolu položaja zglobova implementiran je Q-leaning algoritam. U implementaciji projekta korišćeni su modeli sa tri i pet akcija, odnosno toliko je iznosio broj uglova koji je neki zglob mogao zauzeti. Rezultati pokazuju da model sa većim brojem akcija ima veću stabilnost prilikom hodanja. Pored toga, na stabilnost modela prilikom hodanja dosta utiče odnos prethodnog znanja i istraživanja u trenutku kretanja.

Uvod

Humanoidni roboti mogu imati primenu u obavljanju teških i opasnih poslova umesto čoveka, zbog čega je veoma korisno ulagati u istraživanja i unapređenja njihove tehnologije. Jedan od osnovnih izazova robotike je pokret. Nemogućnost kretanja robota sa točkovima po različitim terenima, rešen je pojavom humanoidnih robota koji imaju iste mogućnosti kretanja kao i ljudi.

Bipedalni model robota, korišćen u ovom istraživanju, sastoji se od nogu i trupa, i kreće se upravljanjem uglova njegovih zglobova (slika 1). Uglovi pri kojima će model na optimalan način stići do cilja mogu se dobiti različitim optimizacionim algoritmima. Jedan od vidova optimizacije je da model sam nauči da dođe do cilja korišćenjem mašinskog učenja. U ovom radu optimizacija hodanja bipedalnog modela je izvršena korišćenjem tipa mašinskog učenja pod nazivom *učenje podsticajem* (reinforcement learning).

Učenje podsticajem

Mašinsko učenje je oblast analize podataka koja se bavi prepoznavanjem određenih obrazaca u podacima. Mašinsko učenje se koristi za različite vrste zadataka, kao što su na primer prepoznavanje slika, prepoznavanje govora i zvuka, automatsko prevođenje, takođe se koristi i kod autonomnih vozila. Nameće se pitanje šta je učenje i kako se uči. Taj proces možemo podeliti na nekoliko podprocesa:

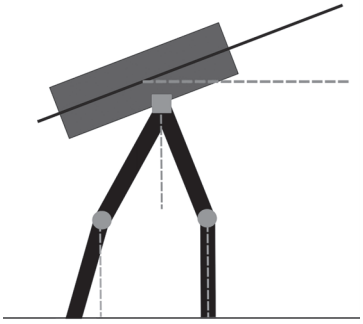
1. *Pokušaj* je deo u kom formiramo stukturu koju ćemo kasnije oceniti i potencijalno poboljšati, ukoliko ne zadovoljava standarde koje zahtevamo. Dakle, ovom delu učenja pripadaju svi načini za formiranje željene strukture.
2. *Evaluacija* je provera generisane strukture. Za ovaj deo potrebno je napraviti funkciju koja ocenjuje strukturu po određenim kriterijumima koje smo zadali.
3. *Evolucija* je proces unapređivanja strukture, koji za krajnji cilj ima kreiranje strukture traženih karakteristika. Ovaj proces može veoma dugo da traje.
4. *Kraj učenja* je trenutak kada je struktura dovoljno dobra da ispunjava sve zadate uslove.

Model mašinskog učenja predstavlja proizvod dobijen nakon procesa učenja koji sadrži

Anđela Bogdanović (2000), Vranovina, Novi Pazar, učenica 4. razreda Prve kragujevačke gimnazije

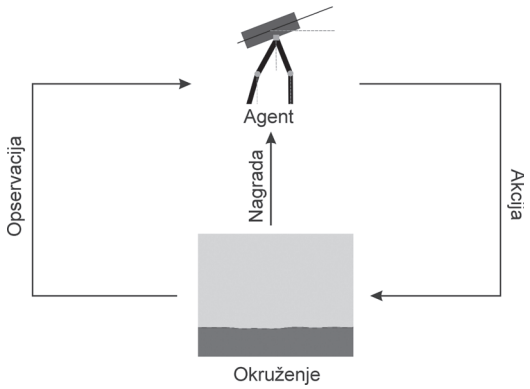
Jelena Cvetić (2001), Guncati, Knić, učenica 3. razreda Prve kragujevačke gimnazije

MENTOR: Andrej Lojdl, Continental Automotive Srbija d.o.o., softverski inženjer



Slika 1. Planarni bipedalni model

Figure 1. Planar bipedal model



Slika 2. Princip rada učenja podsticajem

Figure 2. Principle of Reinforcement learning

navedene elemente. Algoritmi mašinskog učenja dele se na nadgledano, nenadgledano učenje i učenje podsticajem. *Nadgledano učenje* predstavlja tip učenja kod kojeg su modelu date informacije (trening set) u vidu uređenih parova (X, Y), gde X predstavlja ulazni podatak, a Y izlazni podatak modela. Cilj učenja je nalaženje optimalne funkcije koja mapira ulazne podatke u izlazne, uz pomoć trening seta. *Nenadgledano učenje* za razliku od nadgledanog koristi samo ulazne podatke X, bez date željene izlazne vrednosti Y. Zadatak modela je da pronađe pravilnost u podacima.

Metode učenja podsticajem (reinforcement learning) nemaju pripremljene trening setove ili podatke uz pomoć kojih mogu učiti (nije ni nadgledan ni nenadgledan tip učenja), već same kreiraju svoja iskustva interakcijom sa okruženjem (Sutton i Barto 2017). Okruženje predstavlja sve sa čime agent može da interaguje. Agent je, dakle, model koji interaguje sa okruženjem (slika 2). Akcije su metode pomoću kojih agent interaguje sa okruženjem i menja ga, čime menja stanje. Stanje je svaki položaj u kom se agent može naći u okruženju. Da bi učenje bilo moguće, agent mora probati mnogo različitih akcija iz različitih stanja, za koje dobija povratne informacije u vidu nagrade. Nagrada predstavlja brojnu vrednost koju agent dobija od okruženja. Cilj je da se dobije maksimalna ukupna nagrada. Pored istraživanja okoline, agent takođe koristi informacije koje je već naučio kroz epizode, što se naziva eksploatacija. Epizoda predstavlja sekvencu stanja, akcija i nagrada, nakon koje se model vraća u „početno" stanje; epizoda za problem opisan u ovom radu bila bi jedan korak, nakon kojeg model ponovo pokušava da hoda. Agentu se ni u jednom trenutku ne saopštava akcija koju treba da izvrši, već istraživanjem sam otkriva koja akcija daje najveću nagradu. Nakon izvršenja svake akcije, agent dobija nagradu. Nagrade se mogu podeliti u tri vrste: pozitivna nagrada podstiče željeno ponašanje, negativna nagrada naglašava neželjeno ponašanje agenta, dok nagrada iznosi nula ukoliko agent ne uradi ni jednu od prethodne dve varijante. Učenje podsticajem se odvija sledeći politiku koja predstavlja strategiju agenta da maksimizira ukupnu nagradu. Strategija predstavlja način biranja uglova za svaki od zglobova bipedalnog modela, da bi se napravio jedan pokret.

Markovljev proces odlučivanja

Markovljev proces odlučivanja (Markov decision process, MDP) predstavlja matematički okvir za odlučivanje u situacijama kada je ishod relativno nasumičan, i zavisi od modela koji odlučuje.

Markovljevo svojstvo za pretpostavku uzima da svako stanje zavisi samo od prvog prethodnog stanja, preduzete akcije u tom stanju i dobijene nagrade, odnosno da za aproksimaciju novog

stanja nije potrebna istorija svih stanja i akcija (Rastogi 2017). Formula (1) prikazuje da je za aproksimaciju budućeg stanja S_{t+1} dovoljno samo trenutno stanje S_t, a ne cela istorija:

$$\Pr[S_{t+1}|S_t] = \Pr[S_{t+1}|S_1, \ldots, S_t] \qquad (1)$$

Za stanje S_t, koje je Markovljevo stanje, i naredno stanje S', verovatnoća prelaza iz jednog stanja u drugo definisana je sa:

$$P_{ss'} = \Pr[S_{t+1} = s' | S_t = s]$$

Matrica prelaza stanja P definše verovatnoću prelaza iz bilo kog stanja s u određeno stanje s', gde se iz stanja P_{11} prelazi u stanje P_{12}:

$$P = \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix}$$

Markovljev proces nagrađivanja predstavlja Markovljev proces odlučivanja sa vrednostima trenutne nagrade. Iz formule (2) se vidi da trenutna nagrada R_s predstavlja nagradu koju očekujemo u trenutnom stanju S_t:

$$R_s = E[R_{t+1}|S_t = S] \qquad (2)$$

Markovljev proces odlučivanja (MDP) je zasnovan na Markovljevom svojstvu. MDP predstavlja proces nasumičnog odabiranja stanja, korišćenjem Markovljevog svojstva. Određuje ga torka $\langle S, A, P, R, \gamma \rangle$, gde S predstavlja konačan broj stanja, A konačan broj akcija, P verovatnoću promene stanja, R nagradu, a γ faktor umanjenja (Rastogi 2017). Faktor umanjenja (γ) je konstanta vrednosti od 0 do 1. Predstavlja važnost budućih nagrada u odnosu na trenutnu: što je γ bliže 0, manje su bitne buduće nagrade, i obrnuto.

Q-learning

Q-learning je jedan od algoritma učenja podsticajem koji uz pomoć procesa nagrađivanja odlučuje o sledećim akcijama, i time ažurira Q-vrednost (Sutton i Barto 2017). Q-vrednost predstavlja uređeni par određene akcije i aproksimiranu vrednost valjanosti te akcije koju određuje na osnovu nagrade, odnosno nosi informacije o funkciji stanje-akcija. Q-tabela predstavlja n-dimenzionu matricu, popunjenu Q-vrednostima, koje su predstavljene kao verovatnoće koliko je optimalno iz nekog od mogućih stanja, preuzimanjem određene akcije, stići do željenog cilja, prateći povratnu informaciju o nagradi za svaki potez. Kao što je već rečeno, cilj Q-learning-a (uopšteno učenja podsticajem) jeste maksimiziranje ukupne nagrade. Početne vrednosti Q-tabele su nasumično odabrane vrednosti između 0 i 1, a svakom iteracijom te vrednosti se ažuriraju. *Politika* predstavlja način odabiranja Q-vrednosti iz Q-tablice. U ovom radu je korišćena *epsilon greedy politika* (Sutton i Barto 2017), koja utiče na odnos eksploatacije već stečenog znanja i istraživanja.

Kada se zna stanje simuliranog modela (u slučaju bipedalnog robota, stanje predstavlja uređenu listu položaja svakog zgloba u datom trenutku), vodeći se jednom od ponuđenih akcija, simulirani model prelazi u sledeće stanje, prilikom čega Q-vrednost, koja je predstavljena kao uređeni par stanja i uzete akcije, biva ažurirana. Nakon određenog broja iteracija model stiče izvesno „znanje" o valjanosti prelaska iz jednog stanja u drugo, ako sledi primenjenu akciju. To znanje model eksploatiše dalje. Kao što je napomenuto, ideja Q-learning-a je maksimiziranje ukupne nagrade tokom vremena, a model uz pomoć epsilon greedy politike ima šansu da istražuje, i time pronađe bolji put do uvećanja ukupne nagrade.

Vrednosti u Q-tabeli se ažuriraju na osnovu jednačine (3), iz koje se može videti da se Q-vrednost ažurira tako što se posmatra koliko je dobro biti u nekom stanju, što zavisi od nagrade koju model dobije, aproksimacije valjanosti sledećeg stanja i akcije u odnosu na trenutno stanje; α je learning rate, parametar koji određuje veličinu koraka pri svakoj iteraciji prilikom kretanja ka minimumu funkcije gubitka:

$$Q(S_t, A_t) \leftarrow$$

$$\leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_t, A'_t) - Q(S_t, A_t)] \qquad (3)$$

Metod

Simulacija je rađena u programskom jeziku Python. Bipedalni model je kreiran uz pomoć Box2D biblioteke, kao i dinamika celog modela (Šušteršič i Pantić 2015). Okruženje modela predstavlja podlogu po kojoj se model kreće, u

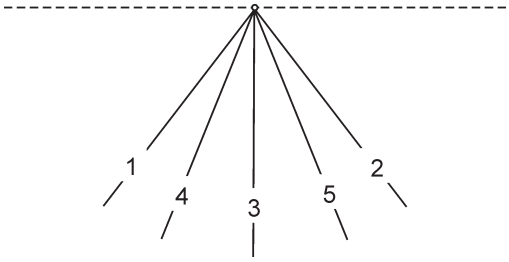
ovom slučaju izabrana je ravna podloga. Nakon kreiranja okruženja i modela, implementiran je Q-learning algoritam za učenje hodanja.

Model i okruženje

Simulirani bipedalni model, koji se sastoji iz dve noge i trupa, prikazan je na slici 1. Noga se sastoji iz potkolenice i butine. Butine su spojene sa trupom uz pomoć zgloba kuka, dok je butina spojena sa potkolenicom kolenom. Model u toku simulacije može kontrolisati uglove svakog zgloba i ugao trupa. Okruženje simulacije (slika 3), napravljeno je korišćenjem programske biblioteke Box2D, koja služi za simuliranje fizičkih modela i njihovu vizualizaciju. Model se kreće po ravnom terenu.

Slika 3. Okruženje simulacije

Figure 3. Environment of the simulation



Slika 4. Prikazane moguće pozicije zglobova

Figure 4. Discrete values of joint angles

Implementiran je model kod kojeg svaki od četiri zgloba ima pet mogućih pozicija koje može zauzeti u toku simulacije. U tabeli 1 su prikazane diskretne vrednosti uglova u radijanima koje svaki od zglobova može uzeti (Morimoto *et al*. 2004). U ovom radu implementiran je model koji za kontrolisanje zglobova i trupa koristi prve tri akcije, kao i model koji koristi svih pet akcija iz tabele 1. Pozicije uglova iz tabele vizualizovane su na slici 4.

Tabela 1. Opseg uglova zgloba [rad]

	Akcija				
Zglob	1	2	3	4	5
Kuk	-0.3	1.3	0.4	0.0	0.9
Koleno	-1.2	-0.5	-0.95	-1.1	-0.35

Algoritam

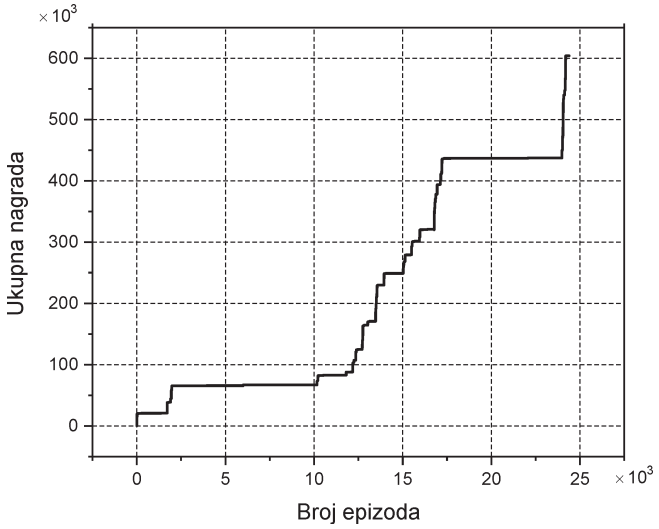
Rezultat optimizacije parametara je Q-tabela, koja nosi informacije o tome koliko je optimalno iz nekog od mogućih stanja, preuzimanjem određene akcije, stići do željenog cilja. Na samom početku se inicijalicuje nasumična Q-vrednost, za svaku epizodu se inicijalizuje početno stanje i iz tog stanja se biraju određene akcije, nakon kojih se dobija nagrada i prelazi u novo stanje. Tom prilikom se izračunava valjanost samog poteza, odnosno valjanost prelaska iz jednog stanja u drugo korišćenjem date akcije. Nakon ažuriranja Q-vrednosti model se nalazi u novom stanju i proces se nastavlja. Pseudokod algoritma je prikazan ispod (Rastogi 2017):

Inicijalizacija Q(*stanje*, *akcija*) tabele
for *epizoda* in range(*EPIZODE*):
    Inicijalizacija stanja
    for iteracija in range(broj iteracija po epizodi):
        odabir akcije *A*# na osnovu maksimalne
                Q-vrednosti za stanje i politike
        nagrada *R*# informacije iz okruženja
        novo stanje S_{t+1}
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) +$
        $+\alpha[r_{t+1} + \gamma \max_a Q(S_{t+1}, A) - Q(S_t, A_t)]$
    # ažuriranje Q-tabele

Rezultati

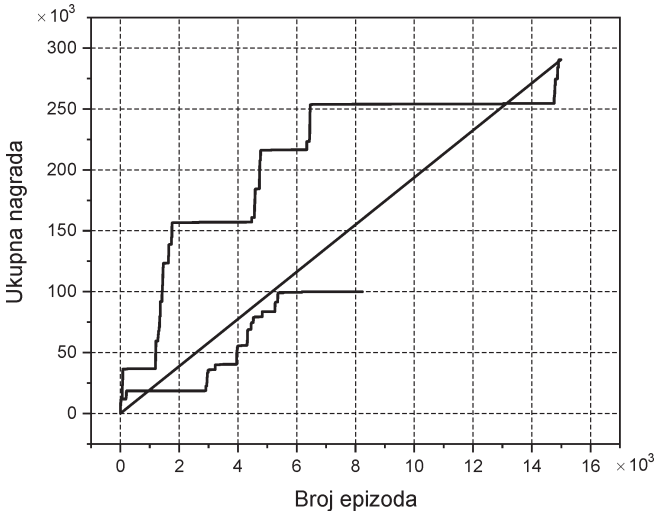
Prvi rezultati su dobijeni sa modelom koji koristi tri akcije, broj epizoda iznosi 25000, svaka epizoda ima po 1000 iteracija. Na slici 5 se vidi povećanje ukupne nagrade sa porastom broja epizoda. Porast nagrade pokazuje da model hoda, jer za svaki korak po x-osi dobija nagradu od 100, a za padanje –100, dok nagrada iznosi 0 kada stoji u mestu. Delovi grafika gde se ukupna nagrada ne povećava, predstavljaju situacije kada model pokušava da poveća nagradu pri čemu isprobava razne akcije. Kada bi se grafik zumirao, videlo bi se naizmenično uvećanje i smanjene nagrade, što je još jedan indikator njegovog učenja. Tokom prvog treninga model je prešao oko 330 cm.

Grafik na slici 6 predstavlja trening modela od pet akcija, broj epizoda je 15000, a broj iteracija po epizodi iznosi 100. Na samom grafiku se vidi da je robot hodao, tj. ukupna nagrada se povećavala do određenog trenutka nakon čega je model pao (naglo smanjenje nagrade). Nakon izvesnog vremena robot je ponovo krenuo da uvećava ukupnu nagradu, nastavio je sa kretanjem. Znamo da nije puzao po površini, pošto za



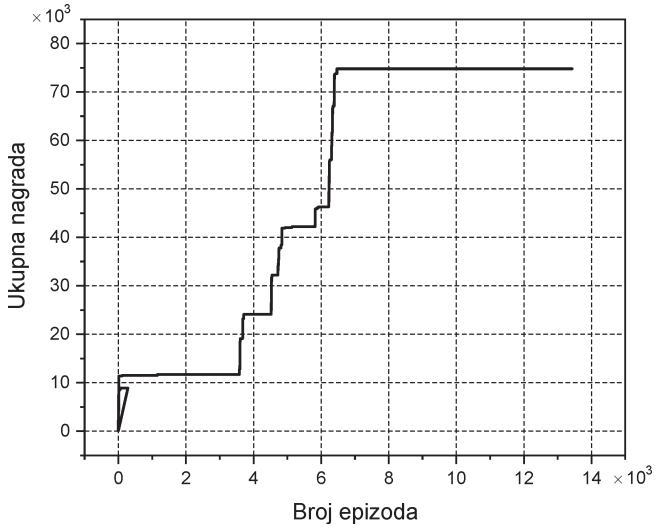
Slika 5. Zavisnost ukupne nagrade od broja epizoda modela sa tri akcije

Figure 5. Dependency of the total reward on the number of episodes with a model with three actions



Slika 6. Zavisnost ukupne nagrade od broja epizoda modela sa pet akcija

Figure 6. Dependency of the total reward on the number of episodes with a model with five actions



Slika 7. Zavisnost ukupne nagrade od broja epizoda modela sa pet akcija kada model koristi već definisanu Q tabelu na osnovu situacije prikazane na slici 6

Figure 7. Dependency of the total reward on the number of episodes with a model with five actions when the model uses the pre-defined Q table based on the situation presented in Figure 6

dobijanje nagrade, trup u određenom položaju ne može se nalaziti blizu površine po kojoj se robot kreće.

Grafik na slici 7 takođe je dobijen modelom koji koristi pet akcija, pri čemu je broj epizoda iznosio 15000, a broj iteracija po epizodi 100. Razlika između situacija predstavljenih na graficima 6 i 7 jeste u tome što u slučaju prikazanom na grafiku 7, model koristi već definisanu Q-tabelu na osnovu situacije sa grafika 6. Na taj način model unapred poseduje određenu predstavu o tome koje su akcije iz kog stanja povoljne, a koje ne. Na grafiku se vidi da je modelu na početku bilo potrebno više vremena za učenje, nagrada je potom rasla, ali je nakon određenog vremena model pao, i nagrada se konstantno smanjivala. Nakon toga, model povećava ukupnu nagradu. Tokom ovog treniranja model je prešao 37 cm.

Diskusija i zaključak

Kada se uporede grafik sa slike 5 i grafik sa slike 6, uočava se razlika u broju akcija koju svaki od zglobova može uzeti. Vidi se da je model sa pet akcija brže učio, odnosno manje vremena mu je trebalo za povećanje ukupne nagrade. Takođe se vidi da model sa pet akcija na početku sporije uči, potrebno mu je više vremena za istraživanje, ali kasnije mnogo brže napreduje od modela sa tri akcije. Uspešnost učenja je veća sa modelom od pet akcija, nezavisno od broja epizoda. Može se videti da u prvih 15000 epizoda model sa pet akcija bolje prolazi od modela sa tri akcije. Takođe, sa povećanjem broja mogućih akcija i povećanja broja iteracija, povećaće se i uspešnost treniranja. Pored toga što je sam model moguće trenirati od nule, moguće je i prethodno „treniranu" Q-tabelu implementirati u model, i na taj način dobiti bolje rezultate (grafik na slici 7).

Istraživanje je moguće proširiti promenom terena, pored jednoličnog terena po kojem se kretao model moguće je uvesti različite prepreke (stepenice) i videti kako će se model ponašati u drugačijem okruženju. U ovom radu je ispitivan model sa 3 i sa 5 akcija; to se takođe može promeniti, i uvesti još više diskretnih vrednosti za kontrolisanje samih zglobova. Pretpostavka je da bi sa povećanjem broja akcija i ukupna nagrada bila veća. Pokazalo se da se povećanjem broja iteracija model ima manju šansu za padanjem, jer ima više mogućnosti koje može da istraži pri jednoj epizodi. Takođe, Q-learning se može uporediti sa nekim drugim algoritmom učenja podsticajem ili genetskim algoritmom.

Zahvalnost. Velika zahvalnost Andreju Lojdlu na pomoći pri izradi projekta, kao i stručnoj saradnici Sofiji Petrović sa seminara primenjene fizike i elektronike za pomoć oko pisanja i korekcije izveštaja.

Literatura

Morimoto J., Cheng G., Atkeson C. G., Zeglin G. 2004. A simple reinforcement algorithm for biped walk. U *Proceedings of the 2004 IEEE International Conference on Robotics & Automation. IEEE*, str. 3030.

Rastogi D. 2017. Deep reinforcement learning for bipedal robots. Dostupno na https://repository.tudelft.nl/islandora/object/uuid%3A0fac495f-f87a-4a61-a80f-5f901323379a

Sutton R. S., Barto A. G. 2017. *Reinforcement Learning: An Introduction*. MIT Press

Šušteršič J., Pantić M., 2015. Primena genetskog algoritma za učenje hodanja dvodimenzionalnog bipedalnog modela. *Petničke sveske*, **74**: 95.

Anđela Bogdanović and Jelena Cvetić

Walking Optimization of Bipedal Walker Using Reinforcement Learning

The goal of this project was to optimize the walking of a bipedal walker using the Reinforcement learning machine learning method. The simulation was implemented in Python, physics and body dynamic were made using the Box2D Python library. The agent, in this case the bipedal walker, interacts with the environment taking actions and with them shifting his states. For every action taken, the agent gets a feedback signal from the environment, a reward. Interacting with the environment for a while, the model can learn which action to take in which state. In this paper the Q-learning algorithm was implemented on a model with three given actions and five given actions. Actions represent previously defined angles. Results show that the stability increases as the number of given actions is bigger. Besides the number of given actions, the stability of the model increases in cases where exploitation prevails over exploration.