Treniranje robotske ruke za pomeranje objekta uz pomoć pojačanog učenja

Darko Tica
Fakultet tehničkih nauka
Univerzitet u Novom Sadu
Trg Dositeja Obradovića 6
21000 Novi Sad
dtica17@gmail.com

***Apstrakt* – kreiranje robota i automatizovanje njihovih akcija predstavlja jedan od ključnih problema današnjice, i sastavni je deo primene robotizacije u svim ljudskim delatnostima. Ovaj rad se bavi podučavanjem robota u simuliranom okruženju u kome ima zadatak da pogura objekat u vidu cilindra do cilja, bez ljudske intervencije. Kao bazično okruženje iskorišćeno je *Pusher* okruženje iz *gymnasium* biblioteke, koje se zasniva na *MuJoCo* projektu, dok je agent treniran uz pomoć pojačanog učenja koristeći SAC (Soft Actor-Critic) algoritam. Rezultujući robot uspeva da reši problem, odnosno pogura cilindar do cilja u 95% slučajeva, pri čemu je testiran na nasumičnim pozicijama cilindra i ciljnog mesta. Glavni doprinos ovog rada jeste uvid u mogućnosti treniranja robota uz pomoć SAC algoritma, kao i značaj njegove parametrizacije, koja u ovom slučaju uključuje dodavanje šuma na akcije radi eksploracije i robusnosti, kao i korišćenje odgovarajuće funkcije stepena učenja (learning rate) u cilju olakšavanja učenja.**

Ključne reči – robot, simulacija, pojačano učenje, SAC, MuJoCo

I. UVOD

Robotika je jedna od najbrže rastućih grana tehnologije današnjeg vremena, i predstavlja osnovu automatizacije brojnih ljudskih delatnosti. Od pomeranja objekata u skladištima i gradnji auta u fabrikama, do korišćenja robota u hirurgiji u cilju veće preciznosti kod osetljivih zahvata, njihova primena se svakim danom sve više povećava, i očekuje se da će još značajnije da raste. Međutim, pored samih fizičkih izazova izgradnje takvih robota, postoje i problemi njihove kontrole i automatizacije. Roboti često treba da izvode kompleksne operacije koje je vrlo teško izmodelovati. U takvim situacijama, jedno od najkorišćenijih rešenja jeste kreiranje simuliranog okruženja u kome bi se robot uz pomoć veštačke inteligencije podučio kontrolama i pripremio za korišćenje u realnom svetu. Na taj način, robot može naučiti izuzetno kompleksne operacije, koje bi za ljude bilo izuzetno teško ili čak nemoguće izmodelovati i isprogramirati ga da radi (poput hodanja [1]). Jedno od često korišćenih simuliranih okruženja za takvo treniranje jeste MuJoCo [2], pri čemu se robot podučava uz pomoć pojačanog učenja, što je upravo pristup iskorišćen u ovom radu.

Kao osnova za samog robota u ovom projektu je upotrebljeno *Pusher* okruženje iz *gymnasium-farama* [3] projekta. U okviru tog projekta, zadatak robota je da pogura objekat (u ovom slučaju cilindar) do ciljne tačke, pri čemu se položaj objekta bira nasumično, dok je cilj fiksiran. U ovom radu, robot poseduje jednu ruku koja se nalazi na sredini samog robota (umesto sa desne strane robota kao u *Pusher*-u), dok je ruci dodata još jedna osa rotacije, čime se dobija veća sloboda kretanja. Takođe, pozicija i objekta i ciljne tačke se u ovom okruženju određuju nasumično, pri čemu su povećane i njihove regije mogućeg pojavljivanja (veći broj početnih pozicija), čime je dodatno povećana kompleksnost samog problema. Kao algoritam za treniranje je iskorišćen **SAC** (Soft actor-critic) [4], odnosno njegova implementacija u *stable-baseline3* biblioteci [5]. Sam robot je testiran tako što je postavljen pred pozicije cilindra i cilja koje nije video tokom treniranja (nasumično su određene pre treniranja), i od 60 pozicija uspeva da dogura objekat do cilja u 57 slučajeva (95%), u vremenskom roku ograničenom na 400 koraka u okviru okruženja.

U narednom poglavlju će biti detaljnije opisano okruženje u kojem je obučavan agent. U trećem poglavlju će biti opisan metod treniranja agenta, odnosno izazovi i iskorišćena rešenja. U četvrtom poglavlju će biti opisani rezultati, dok će poslednje poglavlje obuhvatati zaključak rada.

II. OKRUŽENJE ZA TRENIRANJE

Kao što je već napomenuto, osnova za okruženje za kreiranje i treniranje agenta jeste *Pusher* okruženje. Ono je zasnovano na *MuJoCo* biblioteci, koja u okviru sebe sadrži physics engine, koji služi za prikaz odnosno renderovanje okruženja, kao i za sva izračunavanja vezana za fiziku između objekata, njihovih pozicija i sila dejstvovanja između njih.

Iako se sile i trenje ne zanemaruju u ovom okruženju, masa objekata i trenje su relativno male, tako da agent ne mora da se bavi i učenjem kompleksnih fizičkih efekata, što nije moguće izbeći u realnim situacijama. Uprkos tome, agent i dalje može da nauči ključne aspekte potrebne za savladavanje rešenja,

poput toga da će guranjem objekta doći do njegovog pomeranja u smeru određenog uglom dejstva sile.

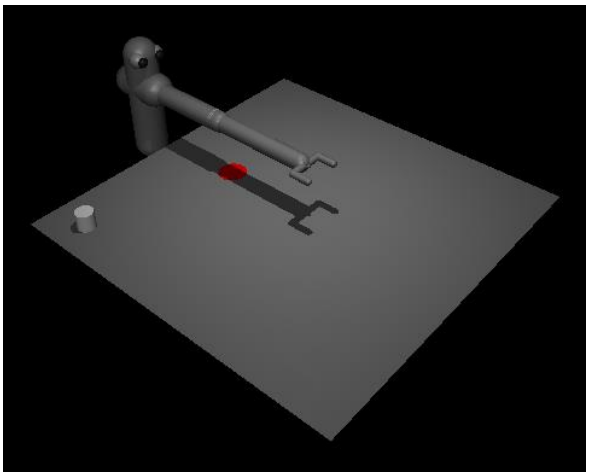
Što se tiče samog agenta, on sadrži tri zgloba koja mogu da se rotiraju – rame, lakat i šaku. Rame može da se rotira oko sve tri ose (*x, y* i *z*), dok lakat i šaka mogu oko dve (*x* i *y*). Svako rotiranje može biti u iznosu od -2 do +2 radijana (što je otprilike od -115° do +115° stepeni). Na taj način, simulira se pomeranje realnog robota, čija je sloboda kretanja ograničena. Dužina dela robota od ramena do lakta, kao i od lakta do šake je po 0.6 (jedinica dužine izražena u okruženju), što robotu daje prostor pokrivenosti od 1.2 u krugu oko njega (ne računajući vrh šake odnosno prste).

Kako bi robot bio u fizičkoj mogućnosti da gurne objekat do cilja sa svake date pozicije, njihove inicijalne pozicije pojavljivanja su ograničene. Njihova pozicija se tako kreće od -0.5 do +0.5 po *x* osi, odnosno -0.3 do +0.3 po *y* osi, pri čemu se objekat i cilj ne mogu pojaviti na udaljenosti manjoj od 0.3 (da problem ne bi postao trivijalan). Sa druge strane, rame robota se nalazi na poziciji *(0, -0.5, 0.325),* tj nalazi se na sredini *x* ose, izmaknuto od ose podloge po *y* osi (jer inače ne bi mogao da dohvati pozicije koje su odmah ispred njega), i izdignut u odnosu na ravan objekta i cilja po *z* osi (rame se nalazi iznad podloge, kao što bi bilo i u realnom svetu). Na taj način osigurano je da robot može da pokrije celu regiju pojavljivanja objekta i cilja. Primer početne pozicije okruženja dat je na *Slika 1*.

III. Treniranje agenta uz pomoć pojačanog učenja

A. Akcije, obzervacije i nagrade

Najosnovniji elementi okruženja koji moraju da se definišu za svaki proces pojačanog učenja su akcije, obzervacije i nagrade agenta. Akcije predstavljaju skup mogućih akcija koje agent može da izvrši. U slučaju ovog agenta, akcije su predstavljena uglom rotacije svih 7 zglobova (rame sadrži 3 zgloba, lakat i šaka po 2 – svaki zglob za jednu osu rotacije), odnosno sve akcije u jednom koraku koji agent vrši su predstavljene vektorom od 7 vrednosti. Kako bi se olakšalo treniranje, te vrednosti su normalizovane u rasponu od -1 do +1.



Slika 1 Primer početnog stanja robota (ciljno mesto je predstavljeno crvenom bojom)

Obzervacije agenta predstavljaju informacije o okruženju agenta koje on vidi. U ovom slučaju, one predstavljaju vektor od 23 vrednosti:

- po dve vrednosti za svaki od 7 zglobova (trenutni ugao zgloba kao i njegova ugaona brzina)
- (x, y, z) koordinate od cilindra
- (x, y, z) koordinate od ciljne tačke
- (x, y, z) koordinate od vrha prsta robota

Nagrade agenta predstavljaju glavno merilo uspešnosti agenta, i služe kao osnova za računanje funkcije greške. Njihov zadatak je da navode agenta ka pozitivnim akcijama, i da kazne loše akcije tj. akcije koje ne vode ka cilju, čime direktno utiču na učenje. Stoga, dobro definisane nagrade mogu da značajno ubrzaju i olakšaju treniranje agenta, ali isto tako da ga otežaju ukoliko je njihova definicija loša. U ovom slučaju, konačna nagrada se sastoji iz tri aspekta:

- bliskost prsta od objekta
- bliskost objekta od cilja
- jednostavnost pojedinačne akcije

Prve dve nagrade su poprilično intuitivne: ukoliko je vrh prsta koji agent pomera bliži objekat, to nagrada treba da bude veća, što isto važi za udaljenost objekta od cilja. Sa druge strane, kompleksnost pojedinačne akcije se računa kao zbir intenziteta svih zglobova (njihovih rotacija). Agent je nagrađen ukoliko su te akcije jednostavne - ukoliko bi svi zglobovi bili aktivirani istovremeno to bi dovelo do poprilično nasumičnih poteza koji verovatno ne bi vodili do konvergencije. Nakon eksperimentisanja, utvrđeno je da je odnos vrednosti nagrada prikazan u *Jednačina 1* optimalan, odnosno konačna nagrada se računa kao:

$$reward = -dist_{go} - 0.5 * dist_{fo} - 0.1 * sqrt(sum(a^2))$$

Jednačina 1 Računanje konačne nagrade

Pri čemu je $dist_{go}$ udaljenost od cilja do objekta, $dist_{fo}$ udaljenost od prsta do objekta, a a vektor akcije. Ono što je bitno primetiti jeste da su nagrade poprilično oskudne, što u ovom kontekstu znači da dva stanja okruženja mogu imati sličnu nagradu (slična udaljenost objekta od cilja, kao i prsta do objekta), ali različitu verovatnoću za rešavanje problema – jedno može potencijalno da obuhvati rešavanje jednim pokretom ruke, dok drugo može da zahteva kompleksne akcije, koje je možda čak i nemoguće izvršiti u ograničenom vremenu okruženja. Time se proces razlikovanja takvih pozicija znatno oslanja na eksperimentisanje prilikom treniranja agenta, što čak i u malom prostoru poput ovog obuhvata veliki broj mogućih stanja koja treba istražiti. Ovo generalno predstavlja značajan problem za treniranje agenata, pogotovo u robotizovanim simulacijama i drugim trodimenzionalnim okruženjima sa velikim prostorom istraživanja.

B. SAC algoritam

Kao glavni algoritam za pojačano učenje korišćen u ovom radu jeste SAC algoritam, koji se zasniva na *actor-cirtic* modelu pojačanog učenja. Međutim, za razliku od ostalih

pristupa poput popularnog **DDPG** (Deep Deterministic Policy Gradient) [6], ovaj model pored klasičnog pristupa u vidu težnje da poveća nagradu, takođe teži da poveća entropiju sistema. Na ovaj način, značajno se podstiče istraživanje novih, neotkrivenih stanja. Agent može da ispita i otkrije optimalne akcije čak iako one na početku treniranja deluju neobećavajuće, te bi u suprotnom slučaju bile odbačene i neistražene. Sa druge strane, tokom trajanja treniranja stepen željene entropije tj. istraživanja se automatski podešava u zavisnosti od uspešnosti agenta i trajanja treniranja, čime se sprečava divergiranje od naučenih optimalnih i istraženih akcija. Ovakav pristup je idealan za robotske kontinualne sisteme sa velikim brojem stanja poput ovoga.

U ovom radu, iskorišćena je SAC implementacija iz *stablebaseline3* biblioteke, čija je dodatna prednost što ima integraciju sa *Openai Gym* [7] okruženjima poput ovog. Međutim, uprkos tome što je algoritam već implementiran, izbor njegovih hiperparametara je veoma bitan i često može predstavljati ključan faktor u podučavanju agenta. Sa druge strane, usled kompleksnosti problema i zahteva za resursima (fizičkim i vremenskim), nemoguće je ispitati sve kombinacije parametara (metodom poput *grid search* npr.). Stoga, u ovom radu su objašnjeni samo parametri koji su istraživanjem i testiranjem smatrani za najbitnije:

- ***learning rate*** – stepen učenja za optimizator
- ***action noise*** – dodavanje šuma na svaku akciju, radi boljeg istraživanja
- ***policy network architecture*** – arhitektura neuronskih mreža
- ***episode length*** – dužina epizode, maksimalan broj koraka koje agent može da uradi u toku epizode
- ***rewards*** – nagrade koje dobija agent, odnos između parametara nagrade

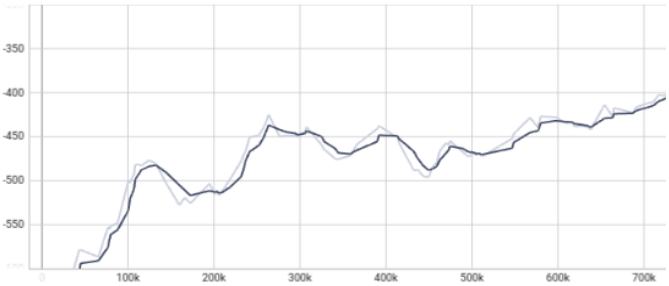
Kao funkcija za određivanje *learning rate*-a (*scheduler* funkcija) u datom momentu korišćena je kosinusna funkcija sa periodičnim restartovanjima [8]. Za razliku od klasnične kosinusne funkcije, koja je pri treniranju isključivo opadajuća, ova funkcija se nakon određenog broja koraka treniranja restartuje (u ovom slučaju learning rate se postavlja na 90% maksimalne vrednosti iz prethodnog restartovanja). Na taj način, learning rate se povećava i smanjuje u ravnomernim intervalima, čime se mreža podstiče da izađe iz lokalnih optimuma u kojima bi se potencijalno nalazila, i na taj način dodatno podržalo eksperimentisanje.

Jedna od stvari koje su takođe doprinele većem eksperimentisanju agenta jeste dodavanje *action noise*-a (šuma akcija). Korišćenjem njega, svakoj akciji agenta se dodavao šum generisan iz normalne raspodele. Taj pristup je doprineo tome da se agent učini robusnijim na greške, odnosno neprecizne poteze koji bi ga potencijalno zbunili i sprečili dalje akcije. Takođe, time se dodatno podstiče eksperimentisanje, s obzirom da šumovi mogu dovesti agenta do novih, neistraženih pozicija [9]. Takođe, bitno je i definisati šumove koji neće biti preveliki, i na taj način degradirati agenta i sprečiti ga da uči. S obzirom da su akcije normalizovane u rasponu od -1 do +1, iskorišćena je normalna raspodela sa *m=0, std=0.1*, koja neće proizvoditi značajno velike šumove za pomenut raspon akcija. Na *Slika 2* i *Slika 3* prikazane su razlike u prosečnim nagradama agenta tokom treniranja bez i sa šumovima akcija – iako se obe vrednosti nagrada povećavaju sa brojem koraka, treniranje sa šumovima dovodi do bržeg i konstantnijeg rasta nagrade, što je uslovljeno većim brojem istraženih koraka koji bi inače bili odbačeni, što je veoma bitno pogotovo na početku treniranja.

Ostali parametri nisu u tolikoj meri uticali na unapređenje rada agenta koliko su bili presudni u tome da li je dolazilo do konvergencije ili ne. U slučaju arhitekture neuronskih mreža, koje predstavljaju osnovu actor i critic modula, korišćenje većih mreža je dovelo do overfittinga i divergencije prilikom treniranja, tako da je korišćen jednostavan *feed-forward* model sa dva sloja od po 512 čvorova, i *ReLU* aktivacionim funkcijama. Što se tiče dužine epizode, dužina od 100 koraka koja se nalazi u originalnom gymnasium rešenju se pokazala kao nedovoljna, te je njen broj povećan na 400, što je takođe omogućilo agentu da ispita veći broj pozicija u okviru jedne epizode. Nagrade i njihov odnos je već opisan u prethodnom delu poglavlja. Nakon eksperimentisanja, utvrđeno je da je originalan odnos prikazan u *Jednačina 1* dao najbolje rezultate, što predstavlja identičan način računanja nagrade kao i u gymnasium okruženju.

IV. Rezultati

Korišćenjem prethodno pomenutih metoda, i treniranjem robota u trajanju od 4000000 koraka (što je otprilike 10000 epizoda), dobijen je finalni agent. On je testiran u istom okruženju u kom je i treniran, uz bitnu napomenu da su testne pozicije tj. pozicije cilja i objekta koji se gura, definisane pre samog treniranja. Prilikom treniranja, novogenerisane trening pozicije su proveravane sa bazom testnih, i ukoliko bi se poklapala sa nekom od njih morale bi biti generisane nove,



Slika 2 Prikaz prosečne nagrade tokom treniranja agenta bez korišćenja action noise

Slika 3 Prikaz prosečne nagrade tokom treniranja agenta sa korišćenjem action noise

čime se obezbeđuje to da testne pozicije zaista nisu bile viđene od samog agenta.

Pored toga, same testne pozicije su organizovane tako da se u svakoj od njih cilj nalazi na različitoj poziciji. Da bi se postigla ravnomernost, a i testirala agentova sposobnost da gurne ciljani objekat na bilo koji deo prostora, sam prostor je izdeljen na mrežu od 60 ćelija, pri čemu se startna pozicija cilja u svakom testu nalazi u različitoj ćeliji. Podela je izvršena na 60 delova s obzirom da je prostor generisanja objekta i cilja tokom treniranja bio ograničen na vrednosti -5 do +5 za *x* osu, odnosno -3 do +3 za *y* osu, te je određeno da svaka ćelija bude stranice dužine 0.1 (što i predstavlja okvirnu veličinu samog cilja). Sa druge strane, početne pozicije objekta za guranje su se nasumično generisale kao i u toku treniranja (sa istim ograničenjima), sa time što je njihov minimalan raspon do ciljne tačke za testne slučajeve povećan sa 0.3 na 0.4, da bi se dodatno ispitala robusnost agenta. Prilikom pokretanja testa, agent je uspeo da reši tj. gurne objekat do cilja u 57 datih slučajeva, što predstavlja 95% uspešnosti.

V. Zaključak

U ovom radu prikazano je rešenje za treniranje robota u simuliranom okruženju uz pomoć pojačanog učenja. Iako je korišćenje ovakvog rešenja praktično nemoguće u realnom okruženju, ono ipak daje uvid u mogućnosti SAC algoritma u robotskim zadacima. Takođe, ovo rešenje predstavlja bitnu osnovu za dalje treniranje robota u još komleksnijim okruženjima sa većim brojem akcija, poput većeg prostora za guranje, prepreka koja treba izbegavati prilikom guranja ili većim stepenom slobode pokretanja robota.

Analizom rezultata i vizuelnim nadgledanjem robota može se utvrditi da čak i u relativno malom broju iteracija (svega 10000 epizoda), robot može da nauči izuzetno kompleksne poteze ruke. Bez ikakvog ljudskog uputsva, robot je naučio da sinhronizuje rotaciju ramena, lakta i šake, u sve tri ose, i pritom postigao tačnost od 95%. Međutim, za neka kompleksnija okruženja, najverovatnije je potrebno iskoristiti neke naprednije metode pojačanog učenja. Jedno od potencijalnih poboljšanja jeste korišćenje **HER** (Hindsight Experience Replay buffer) [10], čime bi se robot dodatno podučavao čak i u slučajevima kad ne uspe da dođe do krajnjeg cilja. Takođe, jedna od ideja za unapređenje bi takođe bila upotreba drugačijeg algoritma pojačanog učenja (poput **TD3 – Twin Delayed Deep Deterministic**) ili čak korišćenja skroz drugačije paradigme pojačanog učenja - korišćenje *model-based* umesto predstavljenog *model-free* rešenja.

VI. Literatura

[1] Boston Dynamics - Starting on the Right Foot with Reinforcement Learning: https://bostondynamics.com/blog/starting-on-the-right-foot-with-reinforcement-learning/

[2] MuJoCo - https://mujoco.org/

[3] Gymnasium farama : https://gymnasium.farama.org/index.html

[4] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen. Soft Actor-Critic Algorithms and Applications, 2019.

[5] Stable-baseline3: https://stable-baselines3.readthedocs.io/en/master/

[6] Timothy P. Lillicrap , Jonathan J. Hunt. Continuous Control with Deep Reinforcement Learning, 2016.

[7] Openai Gym: https://github.com/openai/gym

[8] Ilya Loshchilov, Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts, 2017.

[9] Jakob Hollenstein, Sayantan Auddy, Mattero Saveriano, Erwan Renaudo, Justus Piater. Action Noise in Off-Policy Deep Reinforcement Learning: Impact on Exploration and Performance, 2022.

[10] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, Wojciech Zaremba. Hindsight Experience Replay, 2018.

[11] Scott Fujimoto, Herke van Hoof, David Meger. Addressing Function Approximation Errors in Actor-Critic Methods, 2018.