Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.31
DOI: https://doi.org/10.24867/15BE28Pantic

IZBOR HIPERPARAMETARA ALGORITAMA DUBOKOG UČENJA SA POTKREPLJENJEM PRIMENOM GENETSKOG ALGORITMA

SELECTION OF HYPERPARAMETERS OF DEEP REINFORCEMENT LEARNING ALGORITHMS USING A GENETIC ALGORITHM

Vasilije Pantić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Ovaj rad rešava problem hoda robota u prostoru pomoću algoritama dubokog učenja sa potkrepljenjem koji se optimizuju pomoću genetskog algoritma.*

Ključne reči: *Duboko učenje sa potkrepljenjem, genetski algoritam*

Abstract – *This paper solves the problem of robots walking in space using deep reinforcement learning algorithms that are optimized using a genetic algorithm.*

Keywords: *Deep Reinforcement Learning, Genetic Algorithm*

1. UVOD

Veštačka inteligencija (eng. *artificial intelligence - AI*) ima za cilj pravljenje programa koji bi imitirao ljudska ponašanja i rezonovanje. *AI* na osnovu prethodnog iskustva (podataka kojim ima pristup) pokušava da nađe šablon po kome su mapirani podaci tako da dolaskom novih podataka može da donosi odluku ili rezultat sličan tim podacima. *AI* zahteva ogromnu količinu podataka u odnosu na ljudski mozak, kome je potrebno nekoliko puta da ugleda dva različita objekta da zna da ih razlikuje, dok to sa *AI*-em nije slučaj.

Pored traženja šablona u podacima, postoji i način učenja donošenja odluka načinom kazni i nagrada. Kao što čovek teži u životu nagradama i pozitivnim stvarima, tako je i nad ovim pristupom cilj da se loše odluke penalizuju negativnom nagradom (kaznom), dok bi odluke koje vode ka željenim ciljevima davalo pozitivne nagrade.

Rad rešava problem kretanja robota (koji predstavlja repliku čoveka) sa ciljem da hodajući u prostoru održava ravnotežu i uspešno korača napred metodama nagrađivanja i kažnjavanja, izraženim kroz mašinsko učenje i pronaći optimalna rešenja u pogledu onih koja su najbolja i time postići maksimalne vrednosti i na najbolji način obučiti robota da hoda.

2. IMPLEMENTACIJA REŠENJA

2.1. Uvod u problem i tehnologije

Problem koji ovaj rad rešava pomoću algoritama dubokog učenja sa potkrepljenjem i genetskim algoritmom se tiče učenja robota hodu (eng. *humanoid robot walk*).

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Milan Rapaić, red. prof.

Okruženje koje se koristi za ovaj problem je predefinisan (nije pravljen od strane autora rada) i u pitanju je *OpenAI Gym* [4] i *PyBullet Gym* [5], koji daju gotovo okruženje u kome se robot nalazi, samog robota sa svojim akcijama koje može da izvršava, kao i sistem nagrađivanja i kažnjavanja.

Programski jezik iskorišten za implementaciju rešenja jeste *Python* sa radnim okvirom za duboko učenja *PyTorch* [6]. Na slici 1 dat je izgled robota i okruženja koji se rešava u ovom radu.

Slika 1. *Primer robota i okruženja problema [5]*

Cilj robota prilikom hoda jeste da se uspešno kreće u prostoru (prostor nema prepreke). Stanje prostora u kome se robot nalazi je predstavljeno kao niz veličine 44 popunjen vrednostima od -1 do 1 gde ti brojevi prezentuju ugao i poziciju zglobova robota. Kao akcije koje robot vrši u ovom prostoru, prezentovane su kao niz veličine 17 popunjen vrednostima od -1 do 1 gde ti brojevi predstavljaju momente sile primenjene na zglobove. Po pitanju nagrađivanja robota za uspešne akcije, robot ostvaruje nagradu onda kada se pomera unapred i kada protivi teži gravitacije.

Formalnije izraženo, cilj je pronaći politiku π kojom će agent postizati maksimalnu kumulativnu nagradu nad trajektorijom, tačnije da će povrat jedne epizode biti maksimalan. Robot počinje iz početnog stanja s_0 te je cilj izvršavati akcije koje će omogućiti robotu da ne završi u neželjenom terminalnom stanju, koje zapravo predstavlja njegov pad.

Da bi za ovo rešenje postojalo željeno terminalno stanje, određen je maskimalan broj akcija koje agent može izvršiti u jednoj epizodi, tako da se sada problem odnosno kriterijum definiše kao postizanje maskimalnog broja akcija koje agent može izvršiti.

Algoritmi koji se koriste za rešavanje ovog problema su *PPO* [3] i *TRPO* [1], dok se genetski algoritam koristi za optimizaciju hiperparametara nad algoritmima tako da se ostvari bolji i brži rezultat. Sav kod implementacije rešenja problema se nalazi na [7] i [8].

2.2. Implementacija neuronskih mreža

Oba algoritma dubokog učenja sa potkrepljenjem (*TRPO* i *PPO*) koriste neuronske mreže da aproksimiraju politiku i funkciju vrednosti stanja. Neuronska mreža koja aproksimira politiku naziva se glumac (eng. *actor*), a neuronska mreža koja aproksimira funkciju vrednosti stanja se naziva kritičar (eng. critic).

Ideja pristupa sa glumcem i kritičarem jeste da kritičar aproksimira funkciju vrednosti za celi prostor na osnovu nagrada koje agent ostvaruje vršeći trajektorije u datom prostoru odnosno na osnovu dosadašnjeg iskustva, dok je cilj glumca da ažurira parametre politike i aproksimira politiku agenta u pravcu sugerisanom od strane kritičara.

Obe arhitekture neuronskih mreža imaju isti ulaz u mrežu - niz veličine 44 koji prezentuje stanje u kome se agent trenutno nalazi.

Broj skrivenih slojeva za obe arhitekture je 2, sa brojem neurona po sloju 128 i 64 redom. Jedina razlika između ove dve arhitekture jeste u dimenziji izlaznog sloja, koji za glumca ima 17 neurona u izlaznom sloju da predstavi akciju koju treba agent da izvrši u datom stanju, dok kritičar ima samo jedan neuron u izlaznom sloju da predstavi funkciju vrednost tog stanja. Obe arhitekture kao aktivacionu funkciju (osim nad izlaznim slojem) koriste *ReLU* aktivacionu funkciju.

Kako je u pitanju kontinualan prostor akcija koje agent i izvršava i kako se očekuje da vrednosti vektora koji se prosleđuje kao akcija ima vrednosti u intervalu $[-1,1]$, tako se za izlaz glumca koji treba da proizvede sledeću akciju koristi *Gaussova MultivariateNormal* distribucija [9], koja se zapravo brine da izlaz iz neuronske mreže baš daje takve vrednosti, postavljajući srednju vrednost distribucije na izlaze neuronske mreže glumca.

2.3. Implementacija algoritama dubokog učenja sa potkrepljenjem

Hiperparametri koji su od interesa konkretno za *TRPO* algoritam [1] i koji su bitni za adekvatno treniranje samog algoritma na zadatom problemu su sledeći:

- *delta*: ograničenje KL-divergencije
- *gamma*: faktor obezvređivanja nagrada
- *cg_delta*: ograničenje za *conjugate gradient* algoritam [2]
- *cg_iterations*: broj maksimalnih iteracija za conjugate gradient algoritam [2]
- *alpha*: koeficijent za računanje pravilnog gradijenta da zadovolji *delta* koeficijent
- *backtrack_steps_num*: maskimalan broj koraka kojim se ide unazad za računanje pravilnog gradijenta da zadovolji *delta* koeficijent
- *critic_epoch_num*: broj epoha za treniranje kritičara za jednu epohu *TRPO* algoritma

Implementacija TRPO algoritma data je na [7]. Kod TRPO, neuronska mreža kritičar se ažurira gradijentnim spustom sa ciljem da estimira funkciju vrednosti stanja, dok neuronska mreža glumac računa manuelno gradijent kao maksimalni korak koji unapređuje politiku, a zadovoljava ograničenje dozvoljene razlike stare i nove politike, tačnije $delta$.

Hiperparametri koji su od interesa konkretno za *PPO* algoritam [3] i koji su bitni za adekvatno treniranje samog algoritma na zadatom problemu su sledeći:

- *number_of_network_updates_per_iteration*: broj epoha za treniranje kritičara i glumca za jednu epohu *PPO* algoritma
- *discounting_factor*: faktor obezvređivanja nagrada
- *clip_range*: specifično ograničenje koje ograničava koliko stara i nova politika mogu da se razlikuju

Implementacija PPO algoritma data je na [7]. Kod PPO, neuronska mreža kritičar se ažurira gradijentnim spustom sa ciljem da estimira funkciju vrednosti stanja, dok neuronska mreža glumac se ažurira gradijentnim usponom sa ciljem da estimira politiku agenta.

2.4. Implementacija genetskog algoritma

Ideja genetskog algoritma za problem jeste da nađe optimalne parametre za TRPO algoritam [1] (s obzirom da je to algoritam sa više parametara, te zahteva više eksperimentisanja i podešavanja), te je cilj pronaći kandidata koji postiže najbolje performanse u prvih nekoliko desetina epoha nad problemom.

Hiperparametri koji se tiču konkretno genetskog algoritma su sledeći:

- *number_of_agents*: broj kandidata po generaciji
- *number_of_generations*: maksimalan broj generacija koje će genetski algoritam da produkuje
- *number_of_iterations*: broj epoha TRPO algoritma koje će jedan kandidat da izvrši u generaciji
- *top_limit_agents*: broj elitnih kandidata za prenos u sledeću generaciju
- *mutation_chance*: šansa da će detetovi genomi biti mutirani

Kao fitnes funkcija svakog kandidata se gleda *rewards-to-go* koje je ostvario kroz *number_of_iterations* broja epoha, sa dodatkom da se prvo stanje (početno) postavlja na nulu jer je nasumično i ne igra ulogu.

Za inicijalizaciju populacije, ideja je nasumično dodeliti vrednosti parametara koje želimo da optimizujemo.

Za selekciju roditelja, ideja je da dva roditelja uvek kreiraju dva deteta, koja će imati drugačiji način ukrštanja, sa ciljem razlikovanja među decom. Svaki roditelj ima šansu, ali bolju šansu dobijaju roditelji sa boljom vrednošću fitnes funkcije. Prvo se na osnovu vrednosti fitnes funkcije poredaju u rang (od 1 koji je najbolji do *number_of_agents* koji je sa najmanjom fitnes vrednošću), te se svaki od njih množi sa nasumičnim brojem iz uniformne distribucije od 0 do 1. Za kraj se nad ovim brojevima vrši softmax [10]. Onda se na osnovu šanse tih brojeva biraju roditelji (moraju biti različita).

Za mutaciju genotipa dece, na osnovu *mutation_chance* se odlučuje da li će se detetovi genotipi mutirati ili ne (predstavlja broj između 0 i 1). Dete se mutira tako što se uzme nasumičan broj iz uniformne distribucije između 0 i 1, koji ima za cilj da promeni vrednosti kandidatskih parametara u rasponu -10% do +10% u zavisnosti od nasumičnog broja.

Za ukrštanje genotipa roditelja, ideja je takva da se nađe nasumičan broj iz uniformne distribucije između 0 i 1 i primene sledeće formule za ukrštanje dva deteta koje dva roditelja stvaraju za genotipe dece

$$newParam1 = param1 * c + param2 * (1 - c)$$
$$newParam2 = param2 * c + param1 * (1 - c)$$

Za elitne kandidate, biraju se direktno oni koji imaju najveći fitnes rezultat, broj elitnih kandidata definisan je prema *top_limit_agents* parametru. Elitni kandidati prelaze u novu generaciju netaknuti, odnosno ne prolaze kroz proces mutiranja.
Za kreiranje nove generacije se koriste tehnike kreiranja novih kandidata standardnim putem ukrštanja i mutiranja preko roditelja, a ostatak se popuni elitnim kandidatima, tako da veličina populacije iz generacije u generaciju ostane ista.

3. EKSPERIMENTI I REZULTATI

Za adekvatno treniranje algoritama dubokog učenja sa potkrepljenjem potrebni su i adekvatni hiperparametri koji će algoritam da dovedu do zadovoljavajućeg rešenja.
Za treniranje *TRPO* algoritma [1], korišteni su hiperparametri dati u tabeli ispod. Naime, ovi parametri birani su eksperimentisanjem i isprobavanjem, kao i upotrebom preporučenih vrednosti koji za većinu problema radi zadovoljavajuće.

TRPO hiperparametri	
Ime parametra	Vrednost parametra
learning_rate	$2.5*10^{-4}$
delta	$[3*10^{-1}, 3*10^{-2}, 3*10^{-3}, 3*10^{-4}]$
cg_delta	$1*10^{-2}$
cg_iterations	10
alpha	0.99
backtrack_steps_num	100
critic_epoch_num	20
num_of_timesteps	4800
max_timesteps_per_episode	1600
gamma	0.99
num_of_epochs	40900
mean_episode_reward	~526
time_for_training (days)	10

Prilikom treniranja *TRPO* algoritma, pojavljuju se dva glavna problema:

- potreba da se $delta$ hiperparametar menja u toku treniranja, jer na početku treniranja odgovara veća vrednost (ako zadržimo manju, učenje ide sporo do nivoa da se i ne trenira), dok posle nekog vremena zahteva manju vrednost jer se zahteva manji region poverenja radi optimalnijeg treniranja (ako zadržimo veliku vrednost, očekivani povrat epizoda po epohama ili stagnira ili opada)
- dugotrajan proces obučavanja, koji ukazuje na to da hiperparametri mogu bolje biti optimizovani tako da je proces manji

Grafici treniranja i prosečnog povrata po epohama TRPO algoritma dati su na [7].
Za treniranje *PPO* algoritma [3], korišteni su hiperparametri dati u tabeli ispod. Naime, ovi parametri birani su eksperimentisanjem i isprobavanjem, kao i upotrebom preporučenih vrednosti koji za većinu problema radi solidno.

PPO hiperparametri	
Ime parametra	Vrednost parametra
learning_rate	$2.5*10^{-4}$
num_of_timesteps	4800
max_timesteps_per_episode	1600
critic_epoch_num	20
gamma	0.99
clip_range	0.2
num_of_epochs	9000
mean_episode_reward	~558
time_for_training (days)	2

PPO kao algoritam za dati problem se umnogome pokazao kao bolji izbor. U slučaju ovog algoritma, nema potrebe za manuelnim menjanjem parametara, niti za dugim obučavanjem. Grafici treniranja i prosečnog povrata po epohama PPO algoritma dati su na [7].
S obzirom na napomenute probleme koji se javljaju kod *TRPO* algoritma, cilj je sada inkorporirati pristup genetskog algoritma tako da potencijalno reši nedostatke koji se javljaju prilikom obučavanja *TRPO* algoritma na ovom problemu. Cilj genetskog biće da pronađe optimalne hiperparametre koji će dovesti do efikasnijeg treniranja i rešiti navedene probleme.
Kod *TRPO* algoritma, jedini hiperparametar koji zahteva da se manuelno menja tokom treniranja jeste $delta$, i to na opadajući način. Ideja je predstaviti $delta$ kao kvadratnu funkciju koja opada u zavisnosti od trenutnog broja epohe na način:

$$delta = \frac{1}{deltaa2 * n^2 + deltaa1 * n + deltaa0}$$

gde je n broj trenutne epohe, a parametri $deltaa2, deltaa1, deltaa0$ ciljani parametri za optimizaciju nad genetskim algoritmom.

Za treniranje genetskog algoritma, korišteni su hiperparametri dati u tabeli ispod. Naime, ovi parametri birani su eksperimentisanjem i isprobavanjem, kao i upotrebom preporučenih vrednosti koji za većinu problema radi solidno. Konkretan parametar $number\ of\ iterations$ je odabran na datu vrednost kao najbolji odnos vremenske zahtevnosti obučavanja jednog kandidata i broja iteracija (epoha) koliko će jedan kandidat da se trenira.

GA hiperparametri	
Ime parametra	Vrednost parametra
number_of_agents	15
number_of_iterations	70
top_limit_agents	1
mutation_chance	0.1
number_of_generations	17
delta_a2	0.01487870062184785
delta_a1	0.0005061652202640241
delta_a0	8.163872665158063
time_for_training (days)	4.5

Rezultati koji su postignuti na optimizovanim hiperparametrima nisu bolji, čak su i znatno lošiji sa daljim epohama od onih koji se dobijaju sa menjanjem manuelno hiperparametra *delta*, a glavni razlog tome je što vrednost $delta$ postaje premali posle nekog broj epohe (zato što je funkcija predstavljena pomoću nje), te se treniranje izvršava sporo, skoro pa nikako, što ne doprinosi rešenju efikasnosti i optimalnosti, dok je rešenje da se ne menjaju parametri tokom treniranja zadovoljeno. Ovakvim pristupom rešenje nikada ne bi konvergiralo ka krajnjem rešenju jer je vrednost $delta$ premalo i sa sve većim brojem epoha bi težilo nuli odnosno nepostojećem regionu poverenja, tako da učenja nema.
Kao rešenje koje će dovesti do konvergencije, uzimajući u obzir prvobitni eksperiment nad *TRPO* algoritmom jeste promena funkcije koja opisuje deltu na sledeći način:

$$delta = max(\frac{1}{deltaa2*n^2 + deltaa1*n + deltaa0}, deltamin)$$

gde je sada $deltamin$ jednak minimalnoj pozitivnoj vrednosti kojoj može parametar $delta$ da bude jednak, u cilju prevencije premalih vrednosti.
Gledajući novi predlog $delta$ funkcije, kao $deltamin$ se može odabrati najmanja vrednost iz prvog eksperimenta nad *TRPO* algoritmom, a za preostale vrednosti koristeći preostale hiperparametre dobijene genetskim algoritmom. Takav *TRPO* algoritam uspešno konvergira ka dobrim rezultatima i rešava problem potrebe za menjanjem parametara u toku treniranja. Dužina trajanja treniranja je i dalje duga (9 dana), što je u poređenju sa *PPO* algoritmom neefikasno rešenje. Grafici treniranja i prosečnog povrata po epohama optimizovanog TRPO algoritma dati su na [8].

4. ZAKLJUČAK

Kao što se može videti u radu, algoritmi dubokog učenja sa potkrepljenjem mogu biti efektivni za rešavanje problema učenja robota da hoda, te oba primenjena uspešno rešavaju postavljeni problem.
Kao ideja za rešavanja efikasnosti *TRPO* algoritma uz pomoć genetskog algoritma, definitivno bi potreba bila koristiti više iteracija po svakom kandidatu (reda par stotina pa možda čak i do hiljada), jer tek tada bi genetski mogao da pronađe rešenje koje će da efektivno trenira i na početnim epohama i posle nekoliko hiljada epoha i time se svrsta u rang efikasnosti kao što je to postigao *PPO*. Ovo bi zahtevalo velike vremenske i računarske resurse da bi se izvršilo uspešno do kraja. Svakako, *PPO* kao algoritam je generalno bolji u odnosu na *TRPO*, što je takođe i pokazano kroz ovaj rad koliko je efikasniji u treniranju.

5. LITERATURA

[1] Schulman, John, et al. "Trust region policy optimization." *International conference on machine learning*. PMLR, 2015.
[2] Fletcher, Roger. "Conjugate gradient methods for indefinite systems." *Numerical analysis*. Springer, Berlin, Heidelberg, 1976. 73-89.
[3] Schulman, John, et al. "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347* (2017).
[4] https://gym.openai.com/
[5] https://pybullet.org/wordpress/
[6] https://pytorch.org/
[7] https://github.com/reinai/HumanoidRobotWalk
[8]https://github.com/sovaso/GeneticAlgorithmForHumanoidRobotWalk
[9] Reynolds, Douglas A. "Gaussian mixture models." *Encyclopedia of biometrics* 741 (2009): 659-663.
[10] Gao, Bolin, and Lacra Pavel. "On the properties of the softmax function with application in game theory and reinforcement learning." *arXiv preprint arXiv:1704.00805* (2017).

Kratka biografija:

Vasilije Pantić rođen je u Novom Sadu 1997. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Softversko inženjerstvo odbranio je 2021.god.
kontakt: vpantic10@gmail.com