VILNIAUS UNIVERSITETAS
INFORMATIKOS INSTITUTAS
PROGRAMŲ SISTEMOS

Skatinamojo mokymosi taikymas žaidimo agento valdymo programos kūrimui

Application of reinforcement learning to the software development for game agent management

Bakalauro baigiamasis darbas

Atliko:	Jokūbas Rusakevičius	(parašas)
Darbo vadovas:	vyresn. m.d. Virginijus Marcinkevičius	(parašas)
Darbo recenzentas:	j. asist. Linas Petkevičius	(parašas)

Vilnius – 2020

Dėkoju šeimai ir draugams, už meilę ir nuolatinį palaikymą bei darbo vadovui, vyresn. m.d. Virginijui Marcinkevičiui, už visapusišką pagalbą geriausio sprendimo beieškant.

Santrauka

TODO: Santrauka

Raktiniai žodžiai: Skatinamasis mokymas, Sokoban žaidimas, aktorius-kritikas based metodai, raktinis žodis 4, raktinis žodis 5

Summary

TODO: summary

Keywords: Reinforcement learning, Sokoban game, actor-critic based methods, keyword 4, keyword 5

TURINYS

Įvadas

Problematika

Kompiuterių pajėgumui ir atliekamų operacijų per sekundę skaičiui nuolatos didėjant – didėja ir lūkesčiai bei sprendžiamų uždavinių sudėtingumas. Dar reliatyviai neseniai sudėtingiausios programos ir kompiuterių sprendžiami uždaviniai susidėjo iš skaičiuotuvo operacijų ar žinučių perdavimo. Tačiau technologijoms tobulėjant, kiekvienam žmogui kišenėje besinešiojant pirmųjų kompiuterių kaip „ENIAC“ [oCom] dydį pajuokiančius kompiuterinius įrenginius, natūraliai didėja ir jiems keliami iššūkiai.

Šiais laikais kompiuteriai gali simuliuoti atominius sprogimus, nuspėti orus ir atlikti kitas didžiulių skaičiavimo išteklių reikalaujančias užduotis [Ker]. Tačiau užduoties sudėtingumą gali lemti ne tik milžiniškų išteklių skaičiaus reikalavimas. 2016 metais matėme, kaip „Google's AlphaGo“ nugalėjo pasaulio aukščiausio lygio „Go“ žaidėją ir čempioną Ke Jie [Moz17]. Autonominiai gatvėmis važinėjantys automobiliai neišvengiamai artėja, o „Boston Dynamics“ robotai stebina savo galimybėmis [Dyn20].

Šie uždaviniai nėra trivialiai aprašomi ar išsprendžiami, jiems gali net neegzistuoti sprendimas. Tokiems uždaviniams spręsti yra naudojami mašininio mokymosi metodai (pvz. neuroniniai tinklai). Viena šių metodų šaka yra „skatinamasis mokymas“ – agento atliekami veiksmai yra reguliariai vertinami ir atitinkamai agentas yra apdovanojamas arba baudžiamas.

..Kažką apie sokoban...

Darbo tikslas

Šio darbo **tikslas** – išanalizavus populiariausius skatinamojo mokymosi algoritmus, pritaikyti kelis labiausiai tinkamus Sokoban žaidimo aplinkai.....

Darbo uždaviniai

Darbui iškelti **uždainiai**:

1. Paruošti eksperimentinę aplinką ir agentą.
2.

1. Teorija

1.1. Skatinamasis mokymasis

Skatinamasis mokymasis (RL), kartu su prižiūrimuoju ir neprižiūrimuoju mokymu, yra viena iš pagrindinių mašinų mokymosi (ML) paradigmų. Klasikinis RL modelis susideda iš agento (*angl. agent*), kuris priima sprendimus ir atlieka veiksmus (*angl. actions*) pagal strategiją (*angl. policy*), paremtus aplinkos būsena (*angl. state*), ir siekiantis pasiekti didžiausią įmanomą atlygį (*angl. reward*).

Skirtingai nei kitos ML paradigmos, RL sprendžia ne regresijos, klasifikacijos, ar grupavimo, bet atlygiu paremtas (*angl. reward-based*) problemas, ir tai daro bandymų ir klaidų (*ang. trial and error*) principu. Dėmesys yra nukreiptas ne į sužymėtas („angl. labeled“) įvesties ir išvesties duomenų poras, bet į balanso tarp tyrinėjimo (*angl. exploration*) ir išnaudojimo (*angl. exploitation*) ieškojimą [KLM96].

RL algoritmai yra skirstomi pagal skirtingus rodiklius į kelias skirtingas kategorijas. Prieš aiškinantis kuo skiriasi kiekviena kategorija, svarbu suprasti, kad RL algoritmai susideda iš dviejų fazių:

1. **Mokymosi fazė** – (*angl. learning phase*) tai yra algoritmo apmokymo dalis, kai kiekvienas agento priimtas sprendimas ir atliktas veiksmas aplinkoje bei gautas atlygis ir nauja aplinkos būsena yra panaudojama tolimesniam modelio optimizavimui ir gerinimui.
2. **Taikymo fazė** – (*angl. interface phase*) tai yra algoritmo dalis, kai, nepriklausomai nuo atlikto veiksmo optimalumo, modelis nebėra keičiamas ir yra naudojamos iki šiol išmoktos reikšmės.

Kaip jau minėta anksčiau, RL algoritmai yra skirstomi į skirtingas kategorijas. Pagal modelio struktūrą:

1. **Pagrįsti modeliu** – (*angl. model-based*) tai yra algoritmai, kurie optimalios strategijos apskaičiavimui naudojasi transakcijų funkcija (ir atlygio funkcija) (daugiau poskyryje 1.1.3). Pagrįsti modeliu algoritmai gali nuspėti galimus aplinkos pakitimus, kadangi naudojasi apskaičiuota transakcijų funkcija. Tačiau, modelio turima funkcija gali būti tik apytikslė „tikrajai“ funkcijai, tad modelis gali niekada nepasiekti optimalaus sprendimo.
2. **Nepagrįsti modeliu** – (*angl. model-free*) tai yra algoritmai, kurie apskaičiuodami optimalią strategija nesinaudoja ir nebando apskaičiuoti aplinkos dinamikų (transakcijų ir perėjimų funkcijos nenaudojamos). Nepagrįsti modeliu algoritmai bando apskaičiuoti vertės arba strategijos funkciją tiesiai iš patirties (interakcijų su aplinka).

Pagal strategijos pritaikymą:

1. **Besiremiantys optimalia strategija** – (*angl. on-policy*) algoritmai, kurie, mokymosi metu rinkdamiesi veiksmą, remiasi strategija išvesta iš tuo metu apskaičiuotos optimaliausios strategijos bei atlieka atnaujinimus remdamiesi ta pačia strategija.
2. **Nesiremiantys optimalia strategija** – (*angl. off-policy*) algoritmai, kurie mokymosi metu remiasi skirtinga strategija nei tuo metu apskaičiuota optimaliausia strategija. Atnaujinimai atliekami remiantis geresnį rezultatą gražinančia strategija.

Pagal ... asasdasd 1.1.3:

1. **Pagrįsti verte** – (*angl. value-based*) tai yra algoritmai pagrįsti *laiko skirtumų mokymusi* (*angl. temporal difference learning*), kur yra mokomasi funkcija V^π arba V^* (daugiau poskyryje 1.1.1).
2. **Pagrįsti strategija** – (*angl. policy-based*) tai algoritmai, kurie tiesiogiai mokosi optimalios strategijos π^* arba bando apytiksliai surasti optimalią strategiją.

   Toliau šiame poskyryje bus giliau nagrinėjamas RL ir jį sudarantys elementai.

1.1.1. Skatinamojo mokymosi teorija

Šiame darbe bus remiamasi standartine RL aplinka, kur agentas yra aplinkoje $\mathcal{E}$ diskretų kiekį laiko vienetų arba žingsnių (*angl. time steps*). Kiekvieną žingsnį t agentas gauna informaciją apie aplinkos būseną s_t ir iš visų įmanomų veiksmų rinkinio $\mathcal{A}$ pasirenka atitinkamą veiksmą a_t pagal strategiją π, kur π yra sužymėjimas kokį veiksmą a_t rinktis situacijoje s_t. Aplinka agentui grąžina informaciją apie sekančią aplinkos būseną s_{t+1} ir skaliarinę atlygio reikšmę r_t. Toks procesas yra tęsiamas, kol aplinka pasiekia galinę būseną (*angl. terminal state*). Kai galinė būsena yra pasiekiama – procesas yra pradedamas iš naujo. Rezultatas $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ yra bendras žingsnyje t surinktas atlygis su nuolaidos koeficientu (*angl. discount factor*) $\gamma \in (0, 1]$. Agento tikslas yra gauti didžiausia įmanoma tikėtina rezultatą iš visų aplinkos būsenų s_t.

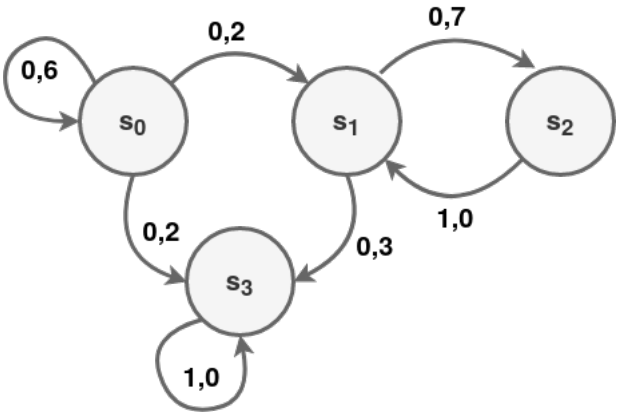
Veiksmo vertė (*angl. value*) apskaičiuojama: $Q^\pi(s, a) = \mathbb{E}[R_t | s_t = s, a]$, kur tikėtinas rezultatas $\mathbb{E}$ gaunamas pagal strategiją π pasirinkus veiksmą a, esant situacijoje s. Optimali vertės funkcija $Q^*(s, a) = \max_\pi Q^\pi(s, a)$, grąžina didžiausią strategijos π pasiektą veiksmo vertę aplinkos būsenai s ir veiksmui a. Panašiai, aplinkos būsenos s vertė remiantis strategija π yra apibrėžiama taip: $V^\pi(s) = \mathbb{E}[R_t | s_t = s]$, ir yra tikėtinas rezultatas būsenai s pagal strategiją π.

1.1.2. Gilusis mokymas

1.1.3. Markovo procesas

Pirmieji aprašyti Markovo procesai (MDP) [BEL57] priminė Markovo grandines (*angl. Markov chains*).

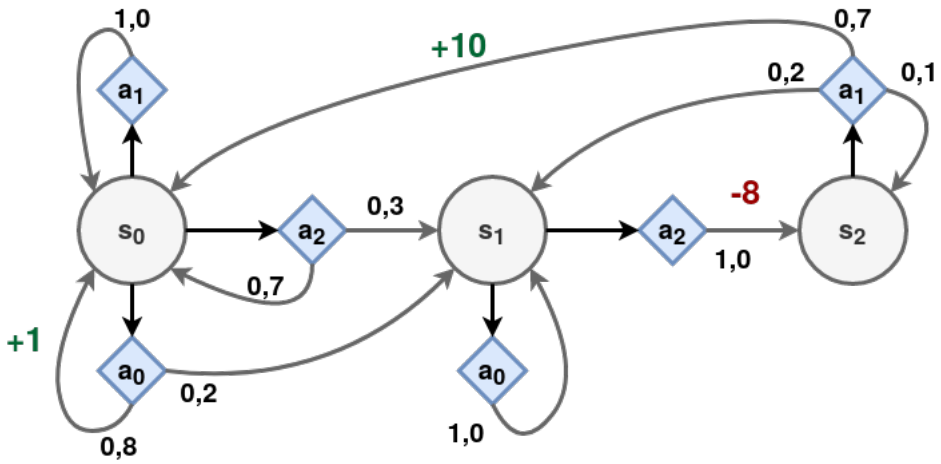
Markovo grandinė – tai kiekviename žingsnyje atsitiktinai judantis iš vienos į kitą būseną (*angl. state*) procesas su fiksuotu skaičiumi būsenų, kur tikimybė pereiti iš būsenos s į būseną s' yra taip pat fiksuota ir priklauso tik nuo poros (s, s'), ne nuo jau praėjusių būsenų. Markovo grandinės neturi atminties [Gér17].



1 pav. Markovo grandinės pavyzdys

Paveikslėlyje 1 pavaizduotas Markovo grandinės pavyzdys su keturiomis būsenomis. Jeigu laikome būseną s_0 pradine, tai yra 60% tikimybė, kad procesas pasiliks šioje būsenoje ir kitą žingsnį. Po tam tikro kiekio žingsnių, procesas galiausiai paliks s_0 ir niekada nebegrįš į šią būseną, nes jokia kita rodyklė nerodo į s_0. Jei procesas pereis į būseną s_1, tai yra labiausiai tikėtina (60% tikimybė), jog kita būsena bus s_2) ir tada iškart atgal į s_1 (100% tikimybė). Procesas gali pereiti per s_1 s_2 kelis kartus, prieš galiausiai patenkant į galinę būseną s_3, kur procesas ir pasiliks.

MDP nuo Markovo grandinės skiriasi tuo, kad MDP perėjimai iš vienos būsenos į kitą, gali turėti jiems priskirtus atlygius (gali būti teigiami ir neigiami). RL problemos labai dažnai yra formuluojamos kaip MDP, kur agento tikslas yra surasti strategija, kuri privestų prie didžiausio bendro atlygio per trumpiausia laiko tarpą [Gér17].



2 pav. Markovo proceso pavyzdys

Paveikslėlyje 2 pavaizduotas MDP pavyzdys su trimis būsenomis ir iki trijų diskrečių veiksmų per būseną. Jeigu laikome, kad agentas pradeda būsenoje s_0, tai pirmame žingsnyje agentas gali pasirinkti vieną iš trijų galimų: a_0, a_1, a_2 veiksmų. Jeigu agentas pasirinktų atlikti veiksmą a_1 – jis garantuotai liktų būsenoje s_0. Tačiau, jeigu agentas pasirinktų veiksmą a_0 – yra 80% tikimybė, gauti atlygį $+1$ ir likti toje pačioje būsenoje s_0 arba 20% tikimybė be atlygio patekti į būseną s_1. Galiausiai arba a_0, arba a_2 veiksmu agentas pateiks į būseną s_1. Šioje būsenoje agentas gali pasirinkti tik vieną iš dviejų veiksmų: a_0 arba a_2. Nors yra du galimi veiksmai, tik veiksmas

a_2 veda į kitą būseną. Tačiau pasirinktus šį veiksmą agentas taip pat garantuotai gauna atlygį (bausmę) -8. Pasiekus būseną s_3 yra galimas tik vienas veiksmas a_1, bet galimi trys skirtingi rezultatai: likti toje pačioje būsenoje s_3 (10% tikimybė), pereiti į būseną s_1 (20% tikimybė) arba grįžti į pradinę būseną s_0 (70% tikimybė) ir gauti atlygį $+10$.

Optimaliai būsenos vertei bet kuriai būsenai s, žymimai $V^*(s)$, nustatyti, galima naudoti *Belmano Optimalumo Lygtį*. Ši rekursyvi lygtis 1 parodo, kad jei agentas atlieka veiksmus optimaliai, tada optimali dabartinės būsenos reikšmė yra lygi vidutiniškai agento gaunamam atlygiui atlikus vieną optimalų veiksmą, plius visų įmanomų toliau einančių būsenų tikėtina optimali vertė.

$$V^* = \max_a \Sigma_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')] \text{ su visais } s \tag{1}$$

- **Transakcijų funkcija** (*angl. transaction function*) $T(s,a,s')$ yra perėjimo iš būsenos s į būseną s' tikimybė, agentui pasirinkus veiksmą a.
- **Atlygio funkcija** (*angl. reward function*) $R(s,a,s')$ yra agento gaunamas atlygis, kai jis pereina iš būsenos s į būseną s', agentui pasirinkus veiksmą a.
- γ yra nuolaidos koeficientas.

  MDP taip pat gali būti užrašyta taip: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$.
- $\mathcal{S}$: visų būsenų rinkinys.
- $\mathcal{A}$: visų veiksmų rinkinys.
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$: transakcijų tikimybės pasiskirstymas $P(s'|s,a)$.
- $R : \mathcal{S} \to \mathbb{R}$: atlygio funkcija, $R(s)$ yra atlygis būsenai s.
- γ: nuolaidos koeficientas.

1.1.4. Sustiprinto mokymosi strategijos ieškojimas

1.1.4.1. MlpPolicy

1.1.4.2. CnnPolicy

1.1.4.3. LSTM

1.1.5. Aktoriaus-kritiko principas

1.2. Neuroniniai tinklai

1.2.0.1. Gilieji neuroniniai tinklai

1.2.0.2. Konvoliuciniai neuroniniai tinklai

2. Metodologija

2.1. Sokoban žaidimas

2.1.1. OpenAI Gym

2.2. Skatinamojo mokymosi bibliotekos parinkimas

2.2.1. Stable Baselines architektūra

2.2.1.1. A2C aprašymas

2.2.1.2. ACER aprašymas

2.2.1.3. POP2 aprašymas

3. Eksperimentai

Šiame skyriuje aprašomi bakalauro darbo metu atlikti eksperimentai bei jiems paruošta eksperimentinė aplinka.

3.1. Sokoban aplinkos paruošimas

Šiame poskyryje yra aprašomas metodas eksperimento metu tiriamos Sokoban aplinkos paruošimui.

3.1.1.

3.2. Eksperimentinė aplinka

Eksperimentai atlikti naudojantis realia mašina su „Ubuntu“ OS. Minėtoje mašinoje įdiegta „Anaconda“ paketų valdymo ir dislokavimo sistema, naudojama aplinkų atskyrimui. Didžioji programinė dalis eksperimento atliekama „Jupyter Notebook“ programavimo aplinkoje naudojantis „Python“ kalba.

3.2.1. Eksperimentinės aplinkos specifikacijos

Eksperimentas atliekamas naudojantis realią „Ubuntu“ mašiną.

1. Kompiuterio techninė specifikacija:
   (a) Procesorius – „**Intel Core i5-9600K**“ (6 branduoliai, bazinis greitis 3.70 GHz).
   (b) Grafinė vaizdo plokštė – „**Nvidia GeForce RTX 2070 Super**“.
   (c) Operatyvioji atmintis – „**HyperX Predator Black**“ (**32GB**, 3200MHz, DDR4, CL16).
   (d) Pastovioji atmintis – „**Western Digital**“ (**1TB**).
2. Kompiuterio programinė įranga:
   (a) Operacinė sistema – „**Ubuntu 18.04 LTS**“ (versija: **18.04.4 LTS**).
   (b) Paketų ir aplinkų valdymo sistema - „**Anaconda**“ (versija: **2020.02**).
   (c) Programavimo kalba – „**Python**“ (versija: **3.7.6**).
   (d) Atviro kodo programa kintančio kodo, matematinių funkcijų, teksto bei duomenų vizualizavimui – „**Jupyter Notebook**“ (versija: **6.0.3**).
   (e) ... – „**Tensorflow**“ (versija: **1.14.0**).
   (f) ... – „**OpenAI Gym**“ (versija: **0.17.1**).
   (g) ... – „**Stable Baselines**“ (versija: **2.10.1a0**).
   (h) Išskirstyta VSC sistema pakeitimų sekimui kode – „**Git**“ (versija: **2.23.0**)

3.2.2. Ekseperimentinės aplinkos paruošimas

3.3. Eksperimento planas

Darbo metu atliktas eksperimentas susideda iš trijų dalių. Šiame skyriuje yra aprašomi šių trijų eksperimentų planai: kaip bus atliekamas eksperimentas, kokia bus naudojama aplinka, kokių rezultatų yra tikimasi ir pan.

3.3.0.1. Pirmo eksperimento planas: Geriausios strategijos ieškojimas

3.3.1. Eksperimentas

Literatūra

[BEL57] RICHARD BELLMAN. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957. ISSN: 00959057, 19435274. URL: http://www.jstor.org/stable/24900506.

[Dyn20] Boston Dynamics. Boston dynamics. 2020. URL: https://www.bostondynamics.com/about (tikrinta 2020-03-19).

[Gér17] A. Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2017. ISBN: 9781491962244. URL: https://books.google.lt/books?id=bRpYDgAAQBAJ.

[Ker] Kate Kershner. What are supercomputers currently used for? URL: https://computer.howstuffworks.com/supercomputers-used-for1.htm (tikrinta 2020-03-19).

[KLM96] L. P. Kaelbling, M. L. Littman ir A. W. Moore. Reinforcement learning: a survey, 1996. URL: https://doi.org/10.1613/jair.301.

[Moz17] Paul Mozur. Google's alphago defeats chinese go master in win for a.i. 2017. URL: https://www.nytimes.com/2017/05/23/business/google-deepmind-alphago-go-champion-defeat.html (tikrinta 2020-03-19).

[oCom] History of Computers. History of computers. URL: https://homepage.cs.uri.edu/faculty/wolfe/book/Readings/Reading03.htm (tikrinta 2020-03-19).

Santrumpos

Darbe naudojamų santrumpų paaiškinimai:

A2C – (*angl. Advantage Actor-Critic*) ...

ACER – (*angl. Actor-Critic with Experience Replay*) ...

MDP – (*angl. Markov Decision Processes*) Markovo procesas.

ML – (*angl. Machine Learning*) mašinų (arba kompiuterių) mokymasis.

PPO2 – (*angl. Proximal Policy Optimization*) ...

RL – (*angl. Reinforcement Learning*) skatinamasis mokymas.

VSC – (*angl. Version-Control System*) versijų tvarkymo sistema.

4. a