Python simuliacija
Gynimo ataskaita

Milita Songailaite

2020 m. gegužės 28 d.

1 Ką pavyko įgyvendinti

Šio projekto tikslas buvo sukurti simuliaciją, kurioje agentai, mokinami neuroninio tinklo, išmoktų susirasti maisto ir išgyventi tam tikrą laikotarpį. Projektą sudaro šios dalys:

- Sukurta grafinė sąsaja su simuliacijos aplinka ir agentais;
- Simuliacijoje agentai gali vaikščioti po visą žemėlapį bei susidūrę su maistu, jį suvalgyti;
- Agentų smegenys, t.y. mąstymo mehcanizmas yra neuroninis tinklas, kurs veikia pagal Advantage Actor Critic (A2C) algoritmą.

Genetiniai algoritmai

Pradžioje semestro buvo planuojama programoje panaudoti ir genetinius algoritmus. Jie turėjo būtų naudojami agentų organizmams besidauginant - genetiniais algortimais būtų nusprendžiami palikuonių turimi požymiai bei turimos žinios aplinkoje. Ši dalis dar nebuvo pridėta, nes tam reikėjo skirtingų smegenų kiekvienam agentui.
Taip pat, buvo mąstyta genetinius algoritmus naudoti greitesniam tinkamų tinklo svorių suradimui, bet ši dalis taip pat dėl laiko stokos buvo palikta ateičiai.

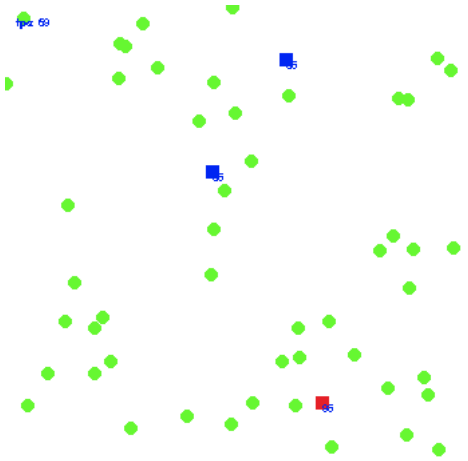
2 Simuliacijos aplinka

Simuliacijos aplinką sudaro tam tikras kiekis mėlyna spalva pažymėtų agentų (po pirmo susidūrimo su maistu, agentas pakeičia savo spalvą į raudoną). Kiekvieną kartą pradedant naują epochą, agentas atsiranda atsitiktinėje žemėlapio vietoje. Agentai gali judėti visame žemėlapyje, susidūrę su maistu (žemėlapyje pažymėtas žalia spalva), jie jį suvalgo. Po susidūrimo su agentu, maistas iš savo vietos yra pašalinamas ir iškart atsitiktinėje žemėlapio vietoje pridedamas naujas maistas.
Kol programoje dar nebuvo neuronino tinklo, nusakančio į kurią pusę eiti, agentas paprasčiausiai eidavo iki artimiausio jam esančio maisto. Atstumams paskaičiuoti buvo naudojamas Euklido atstumas:

$$d(x, y) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Simuliacijos stadijos aprašymas

Vėliau, pridėjus neuroninį tinklą, visas aplinkos žemėlapis buvo paverstas į 10 kartų mažesnį tinklelį, t.y. jei žemėlapio išmatavimai yra $980x740$, tai jis bus verčiamas į $98x74$ langelių tinklelį. Taip buvo padaryta dėl to, kad būtent tas tinklas bus perduodamas neuroniniam tinklui, kaip esama stadija. Jei tinklui duotume kiekvieną žemėlapio pikselį, jo skaičiavimai užtruktų per ilgai ir simuliacija negalėtų reikiamai veikti.



1 pav.: Simuliacijos aplinka

Agentų duomenys

Simuliacijoje visi agentai juda vienodu greičiu, bei turi vienodą alkio rodiklį - 100. Kuo agento alkio rodiklis didesnis, tuo jis yra geresnėje situacijoje. Agentų alkio rodiklis kiekvieną milisekundę krenta po 0.1, o suvalgius maistą, jis padidėja 10. Kai agento alkio rodiklis pasiekia 0, agentas miršta. Bent vienam agentui mirus, simuliacijos epocha yra nutraukiama ir iškart pradedama nauja epocha. Taip pat, jei visi agentai sugeba išgyventi, epocha yra baigiama praėjus tam tikram nustatytam laikui.

3 Algoritmai

Kuriant simuliaciją, buvo išbandyti keli algoritmai, tačiau finale nuspręsta pasilikti prie A2C algoritmo. Žemiau nurodytos priežastys, kodėl išbandyti algoritmai simuliacijai tiko, ir kodėl netiko.

3.1 Q-learning

Vienas pimrųjų pasirinkimų buvo Q-learning algoritmai. Jų veikimas yra paremtas funkcija, kuri sako agentui, kaip reikėtų elgtis tam tikrose situacijose, taigi, tai buvo paprasčiausias pasirinkimas, nuo ko pradėti. Pradžioje algoritmas buvo išbandytas ant nedidelio žemėlapio (1x5), kuriame pirmame ir paskutiniame langelyje buvo maistas - pirmame langelyje vertas 2 tašku, paskutiniame - 10 taškų.

| 2 | _ | ∗ | _ | 10 |

2 pav.: Q-learning algoritmo aplinka, kur agentas pažymėtas “*”

Kadangi pradinis projekto tikslas buvo naudajantis neuroniniu tinklu išmokinti agentus surasti maistą, toliau buvo nuspręsta pereiti prie deep Q-learning algortimo. Tačiau šis algoritmas nepilnai tiko pasirinktam uždaviniui, nes jis apmokina agentą atlikti diskrečius veiksmus. Kadangi simuliacijoje veiksmai turėtų būti tolydūs (t.y. vektoriai vedantys agentą link artimiausio maisto), šis algoritmas nebuvo pasirinktas.

3.2 Neuroninis tinklas

Kitas bandymų etapas buvo pasirašyti neuroninį tinklą, kuris išmoktų rasti geriausias koordinates (x ir y reikšmes), į kurias agentui reikėtų eiti. Tam, kad būtų galima paprasčiau keisti visus parametrus, tinklas

buvo rašytas nenaudojant papildomų neuroninių tinklų bibliotekų. Naudojant paprastą neuroninį tinklą buvo gana sunku nustatyti, ar jo parinktos koordinatės yra geresnės, ar blogesnės, todėl buvo pereita prie paskutinio algoritmo - A2C.

3.3 A2C algoritmas

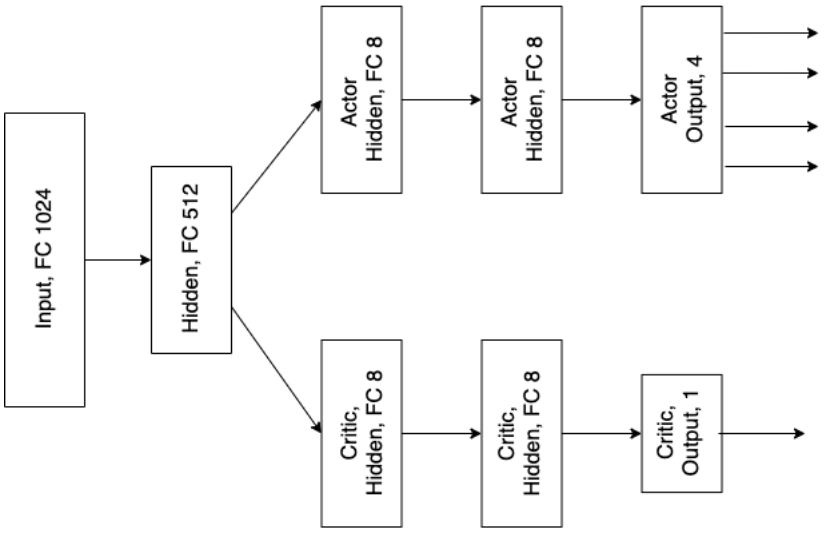
Šis algoritmas veikia panašiai, kaip ir deep Q-learning algoritmai. Kiekvieną iteraciją, jis vertina dabartinę situaciją ir būsimą situaciją. Palygindamas abejas situacijas, jis gali įvertinti dabartinio veiksmo gerumą. Dar vienas pranašumas yra tas, kad algoritmas gali dirbti su tolydžiais inputais ir outputais - tai buvo vienas svarbiausių veiksnių, dėl ko simuliacijai buvo išrinktas šis algoritmas.

4 Pasirinkto algoritmo ypatumai

4.1 Tinklo inputai/outputai

A2C algortimtas kaip tinklo inputus pasiima esamas agentų būsenas žemėlapyje. Kadangi perduodant kiekvieną langelį net ir iš sumažinto žemėlapio, tinklas skaičiuotų svorius pakankamai ilgai, tai buvo nuspręsta sumažinti agentų matomumą. Tai yra, dabar kiekvienas agentas mato ne visą žemėlapį, o tik jo dalį - 32x32 langelių. Taigi, vieno agento inputai į tinklą bus 32x32 dydžio matrica, kurioje 1 yra pažymetas maistas, o 0 - tušti langeliai.
Apsimokęs tinklas gražina 4 reikšmes - jos parodo keturias kryptis (Š, P, V, R), į kurias agentas turėtų judėti, norint pasiekti artimiausią maistą. Pagal šias kryptis yra sudaromi judėjimo vektoriai ir perduodami agentui, kad jis galėtų judėti reikiama kryptimi.

4.2 Tinklo struktūra

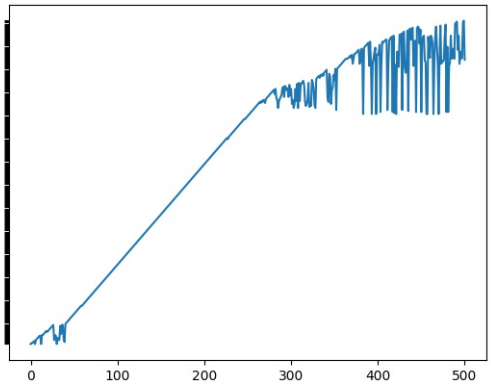


3 pav.: Neuroninio tinklo struktūra

5 Rezultatai

Simuliacija su 100x100 žemėlapiu

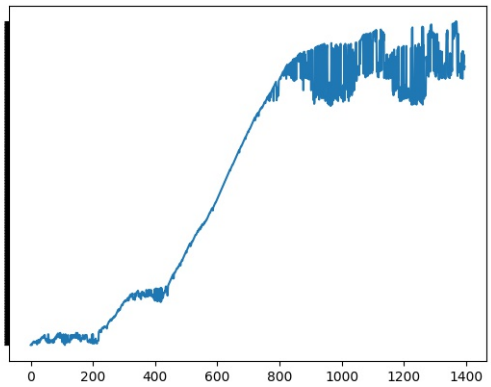
Pradžioje simuliacija vyko tik su vienu agentu mažame 100x100 žemėlapyje. Agentui buvo leista matyti beveik visą žemėlapį - jo matymo laukas buvo 160x160 pikselių. Kadangi agentas yra savo matymo lauko viduryje, tai tuo atveju, kai jis stovi pačiame kampe, jis mato ne visą žemėlapį. Agento greitis buvo pasirinktas salyginai didelis (500).
Simuliacijoje su šiais parametrais, agentas sugebėjo greitai išmokti rasti artimiausią maistą, greitai nustojo eiti i lokalius minimumus, jo vidutiniškai savalgomas maisto kiekis buvo 20 maistų per epizodą.



4 pav.: Simuliacijos su mažu žemėlapiu vidutinis suvalgyto maisto per epizodą grafikas

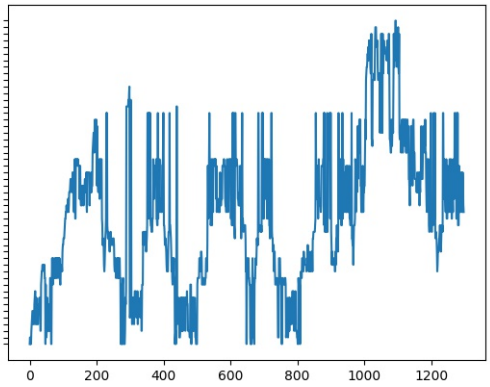
Simuliacija su 200x200 ir 300x300 žemėlapiais

Toliau simuliacija vyko su šiek tiek padidintu žemėlapiu - 200x200. Taip buvo stebima žemėlapio dydžio įtaka vieno agento mokymuisi. Kai agentas mokėsi 200x200 žemėlapyje, jo išmokimas rasti maistą užtruko kiek ilgiau, tačiau jis vis tiek sugebėjo išmokti rasti artimiausią maistą ir iki jo nueiti. Tačiau sumažinus



5 pav.: Simuliacijos su 200x200 žemėlapiu vidutinis suvalgyto maisto per epizodą grafikas

agento greitį (iki 200) ir palikus to paties dydžio žemėlapį, agentas išmokti, kur yra maistas, nebesugebėjo.

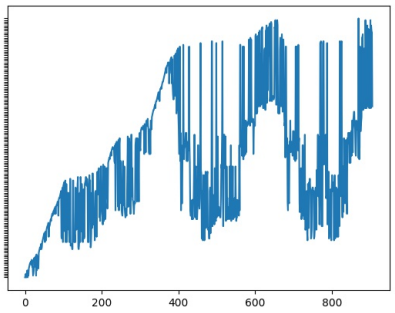


6 pav.: Simuliacijos su 200x200 žemėlapiu vidutinis suvalgyto maisto per epizodą grafikas, kai agento greitis yra sumažintas

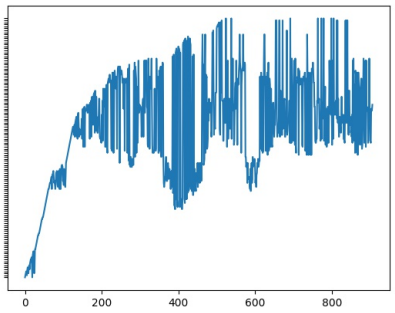
Panašūs rezultatai buvo gaunami ir padidinus žemėlapį iki 300x300 pikselių. Agentas mokėsi labai lėtai, o jo suvalgomo maisto per epizodą vidurkis kito nuo 0.6 iki 2. Taigi, galima matyti, kad tinklas sugeba apmokinti vieną agentą mažoje erdvėję, tačiau su didesnėmis edvėmis jis dirba ne taip gerai.

Simuliacija su 15 agentų, dideliame žemėlapyje

Kadangi projekto tikslas buvo sukurti simuliaciją, kurioje daug agentų išmoktų susirasti masito, tai sukurta simuliacija buvo išbandyta ir su didesniu agentų kiekiu. Šioje simuliacijoje yra 15 agentų, žemėlapio dydis pasirinktas 1200x800 pikselių bei žemėlapyje yra 200 masito objektų. Kadangi kiekvienas iš 15 agentų turėjo savo atskiras smegenis, todėl viena epocha vykdavo ilgiau, nei simuliacijose su mažiau agentų. Ši simuliacija buvo palikta veikti per naktį (t.y. veikė 6h). Po 6 valandų mokymosi, agentai išmoko vidutiniškai per epizodą suvalgyti 3 maistus.



(a) label 1



(b) label 2

7 pav.: Simuliacijos su 1200x800 žemėlapiu dviejų agentų vidutinis suvalgyto maisto per epizodą grafikas

6 Problemos, su kuriomis teko susidurti

Aplinkos ir tinklo sutaikymas

Didžioji dalis nesklandumų vyko todėl, kad sukurtą aplinką buvo sunku pritaikyti neuroniniam tinklui. Pirmiausia, reikėjo nuspręsti, kaip tinklas supras kiekvieno agento aplinką. Tam buvo pasirinktą žemėlapį skaidyti į 10 kartų mažesnį tinklą. Tokiu būdu buvo prarandama šiek tiek informacijos (pvz.: nebuvo galima nustatyti ar langelyje yra vienas maistas, ar keli), tačiau pats tinklas buvo mažiau apkraunamas ir galėjo veikti kartu su simuliacija.
Dar viena problema buvo žaidimo laikrodžio ir neuroninio tinklo laikrodžio suderinimas. Dėl jų nesuderinamumo, agentas kartais eidavo labai per mažus žingsnius ir neprieidavo iki maisto, o kartais jį prašokdavo. Kadangi tai buvo trukdžiai agento mokymuisi, buvo paaukotas tolydesnis simuliacijos piešimas.

Tinklo output'ų vertimas į nuorodas, kur eiti agentui

Pradžioje buvo norėtą, kad tinklas gražintų tolydžias reikšmes, todėl tinklo struktūrą buvo paruošta taip, kad ji gražintų dvi reikšmes - x ir y - nusakančias, kur turėtų keliauti agentas. Šis sprendimas sudarė daug problemų ir tinklas nerasdavo tinkamų svorių agentui. Todėl tinklo struktūra buvo pakeista - dabar tinklas gražina 4 reikšmes su kryptimis, kur agentui reikia eiti. Taip tinklo gražinama informacija tapo diskreti, tačiau tai vis tik programos veikimui nepakenkė.

Agento matymo laukas

Agento matomos erdvės apribojimas buvo tik dar vienas sprendimas, kaip tinklui geriau perduoti apdorotas agento būsenas. Šis apribojimas buvo įdėtas su mąstymu, kad jei agentas matys mažiau maisto (tik tą maistą, kuris yra aplink jį), tai jis greičiau išmoks eiti link vieno pasirinkto maisto, o ne blaškysis po visą žemėlapį. Taip pat, kadangi dabar kiekvienas agentas turi savo skritingą matymo lauką, tai visas judėjimas buvo nusakomas pagal jo esamą poziciją žemėlapyje, o ne žemėlapio centrą. Kitaip sakant, atstumai nuo agento iki maisto nepasikeitė, tačiau pasikeitė agento žemėlapio suvokimas, kadangi jis visada įsivaizduoja, kad jis yra žemėlapio centre.