Emil Keränen

Vertaileva tutkimus koneoppimisen hyödyntämisestä videopelien reitinhaussa

Tietotekniikan pro gradu -tutkielma

11. huhtikuuta 2022

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Emil Keränen

Yhteystiedot: emil.a.keranen@student.jyu.fi

Ohjaaja: Tommi Kärkkäinen

Työn nimi: Vertaileva tutkimus koneoppimisen hyödyntämisestä videopelien reitinhaussa

Title in English: Comparative study of utilizing machine learning in video games' pathfinding

Työ: Pro gradu -tutkielma

Opintosuunta: Tietotekniikka

Sivumäärä: 16+0

Tiivistelmä: TODO: tiivistelmä suomeksi

Avainsanat: koneoppiminen, videopeli, reitinhaku, syvä vahvistusoppiminen

Abstract: TODO: In english

Keywords: machine learning, video game, pathfinding, deep reinforcement learning, Soft Actor Critic, Machine Learning Agents, Unity

Sisällys

1 Johdanto

- Tutkimuksen kohteena koneoppimisen tehostama reitinhaku videopeleissä ja sen vertaaminen heuristiseen A*-algoritmiin.

- Yleisesti reitinhaulla tarkoitetaan alku- ja loppupisteen välisen reitin selvittämistä. Useimmiten tarkoituksena on löytää lyhin reitti väistellen samalla matkan varrella olevia esteitä.

- Reitinhakua tarvitaan videopelien lisäksi myös mm. robotiikassa.

- Videopeleissä reitinhaku ilmenee pääasiassa tekoälyagenttien suorittamana toimintana, joten tässä tutkimuksessa keskitytään agentteihin. Näitä agentteja kutsutaan myös ei-pelaaja-hahmoiksi (engl. non-player-character, NPC).

- Ei-pelaaja-hahmot ja itseasiassa videopelien reitinhaku vertautuvat hyvin robotiikkaan ja robottien reitinhakuun.

- Sekä robotiikassa että videopeleissä toiminta-alue voi muuttua hyvinkin paljon reaaliajassa, jolloin reitinhaun täytyy sopeutua muutoksiin nopeasti. Käytetyt ratkaisut sen sijaan voivat vaihdella näiden kahden osa-alueen välillä: robotiikassa tarkkuus ja turvallisuus nousevat tärkeimmiksi ominaisuuksiksi ja vastaavasti videopeleissä nopeus määrittää reitinhaun "hyvyyden".

- Reitinhaku on aina ollut vaativa ongelma videopeleissä, mutta nykyään reitinhaun ongelmallisuus voidaan useimmissa tapauksissa sivuuttaa laatimalla heuristinen ratkaisu A*-algoritmin avulla.

- Koneoppiminen mahdollistaa aiemman kokemuksen hyödyntämisen myöhemmässä toiminnassa. Agentteja voidaan kouluttaa harjoitteludatan avulla, jolloin ne oppivat toimimaan tuntemattomissa tilanteissa. Koneoppimisen ansiosta reitinhaku-agentti voidaan opettaa toimimaan vaativissa ja dynaamisissa pelialueissa, joissa muuttuvat esteet ja alueen labyrinttimäisyys heikentävät A*-algoritmin toimintaa.

- Tutkimuksen ideana on käyttää Unity-pelimoottorille luotuja koneoppimisagentteja (engl. Unity Machine Learning Agents) ja opettaa niitä erilaisten pelialueiden avulla. Opettamisen

jälkeen agentteja testataan oikeilla pelialueilla ja verrataan tuloksia A*-algoritmilla saatuihin tuloksiin.

- ML-agents perustuu PyTorch-kirjastoon ja mahdollistaa vahvistusoppimisen hyödyntämisen.

- Tensorboard-lisäosan avulla voidaan visualisoida palkkioiden keskiarvot ja opetuksen edistyminen opetuksen aikana.

- Pelialueiden on tarkoitus olla monimutkaisia ja dynaamisia, koska A*-algoritmi suoriutuu yksinkertaisista reitinhakutehtävistä moitteettomasti.

2 Reitinhaku videopeleissä

3 Unity

Unity on Unity Technologiesin kehittämä pelinkehitysalusta, joka sisältää oman renderöinti- ja fysiikkamoottorin sekä Unity Editor -nimisen graafisen käyttöliittymän (Juliani ym. 2018). Unityllä on mahdollista kehittää perinteisten 3D- ja 2D-pelien lisäksi myös esimerkiksi VR-pelejä tietokoneille, mobiililaitteille ja pelikonsoleille. Unitystä onkin vuosien mittaan tullut yksi tunnetuimmista pelinkehitysalustoista, jonka parissa työskentelee kuukausittain jopa 1.5 miljoonaa aktiivista käyttäjää ("Unity" 2022).

Viime vuosina Unityä on käytetty simulointialustana tekoälytutkimuksen parissa. Unity mahdollistaa lähes mielivaltaisten tilanteiden ja ympäristöjen simuloinnin 2D ruudukkokartoista monimutkaisiin pulmanratkaisutehtäviin, joka on sen suurimpia vahvuuksia simulointialustana. Kehitystyö ja prototypointi ovat Unityllä myös erityisen nopeaa. (Juliani ym. 2018).

3.1 Machine Learning Agents

Machine Learning Agents on Unitylle kehitetty

4 Koneoppiminen

Soft Actor-Critic on Haarnojan ym. kehittämä syvä vahvistusoppimis algoritmi (Haarnoja ym. 2018).

5 Tutkimusstrategia/metodi ja sen valintaperusteet

- Empiirinen vertaileva tutkimus?

- Luodaan pelialueita, toteutetaan heuristinen A*-algoritmi, opetetaan koneoppimisagentit syvän vahvistetun oppimisen avulla (Soft Actor Critic -algoritmi) ja sijoitetaan koneoppimisagentit pelialueille. Tämän jälkeen ratkaistaan reitinhakutehtävä erikseen molemmilla menetelmillä ja verrataan saatuja tuloksia keskenään (mm. lasketaan tehtävään kulunut aika, suoriutuiko tehtävästä tietyssä ajassa vai ei).

6 Aineiston keruun suunnittelu

- Agentit koneopetetaan käyttäen Unity ML-agents -pakettia. Opetusprosessista saatu mallitiedosto (model, .onnx-tiedosto) liitetään agentin komponentiksi, jolloin agentti voi toimia pelialueella itsenäisesti.

- Tensorboardin avulla oppimisprosessia voidaan visualisoida esimerkiksi palkkioiden suhteen.

- Aineisto kerätään peliagenteilta jokaisen reitinhakuongelman aikana. Vähintään aika ja tieto siitä onnistuiko agentti tai A*-algoritmi otetaan talteen. Myös Tensorboardin muodostamia kuvaajia voidaan käyttää apuna koneoppimista arvioidessa (mm. palkkioiden keskiarvo opetuksessa).

- Ajan laskemiseen käytetään Unityn valmiita kirjastoja.

7 Aineiston keruu

- Reitinhakuongelmia on jokaista erilaista pelialuetta kohden yksi. Pelialueita luodaan ainakin muutama kymmen.

- Malli-tiedostoja (model, .onnx-tiedosto) luodaan muutamia eri parametreilla, jolloin voidaan verrata parametrien vaikutusta agentin suoriutumiseen.

- Ajan laskeminen alkaa, kun agentti käsketään siirtymään pisteestä A pisteeseen B. Ajan laskeminen loppuu, kun agentti saapuu pisteeseen B. Jos agentti ei syystä tai toisesta pysty ratkaisemaan ongelmaa eli ei saavuta pistettä B (esim. aikamaksimi saavutetaan, alueesta riippuen 20+ sekuntia), merkitään ratkaisun tilaksi "epätosi"ja jätetään aika-arvo tyhjäksi. Jos ratkaisu löytyy, merkitään ratkaisun tilaksi "tosi"ja asetetaan aika-arvoksi mitattu aika. Sama prosessi toistetaan A*-algoritmilla.

- Jos koneoppimisagentteja opetetaan eri määrällä dataa, ilmoitetaan myös opetusdatan määrä.

- Kaikki data kerätään yhteen tiedostoon ja käsitellään jälkikäteen.

8 Aineiston analyysi

- Kuvaajien avulla visualisoidaan kuluneita aikoja ja vertaillaan saatujen aikojen erotuksia.

- TODO: lisätietoja analyysista

9 Tulokset

- TODO

10 Johtopäätökset

- TODO

Lähteet

Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel ja Sergey Levine. 2018. “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”. Teoksessa *International conference on machine learning,* 1861–1870. PMLR.

Juliani, Arthur, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar ym. 2018. “Unity: A general platform for intelligent agents”. *arXiv preprint arXiv:1809.02627.*

“Unity”. 2022. Viitattu 4. huhtikuuta 2022. https://unity.com/.

- Panov, A., Yakovlev, K. ja Suvorov, R., Grid Path Planning with Deep Reinforcement Learning: Preliminary Results, Procedia Computer Science Volume 123, Pages 347-353, 2018, https://doi.org/10.1016/j.procs.2018.01.054

- X. Lei, Z. Zhang ja P. Dong, Dynamic Path Planning of Unknown Environment Based on Deep Reinforcement Learning, 2018, https://doi.org/10.1155/2018/5781591

- TODO: oikea merkintätapa ja muita lähteitä.

11 Liitteet

- Kuvat tai mallinnokset pelialueesta.

- Tensorboardin kuvaajat mm. agentin palkkioiden kehityksestä.