VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

PROGRAMŲ SISTEMOS

**Skaitmeninis intelektas ir sprendimų priėmimas**

**Žaidimas „Kryžiukai–nuliukai“**

**Darbą atliko:**

Vadim Čeremisinov

Pijus Zlatkus

Vilnius

2024

Turinys

[1. Užduoties tikslas 3](#_Toc168070309)

[2. Žaidimo taisyklės ir modelio sprendžiamas uždavinys 3](#_Toc168070310)

[3. Žaidimo neuroninio tinklo realizacija 3](#_Toc168070311)

[3.1. Naudojami programiniai įrankiai ir resursai 3](#_Toc168070312)

[3.2. Neuroninio tinklo architektūra 3](#_Toc168070313)

[3.3. Apmokymo strategija 5](#_Toc168070314)

[3.3.1. Q-learning algoritmas 5](#_Toc168070315)

[3.3.2. Duomenų generavimas 5](#_Toc168070316)

[3.3.3. Aplinka 6](#_Toc168070317)

[3.3.4. Modelio mokymas 6](#_Toc168070318)

[3.3.5. Modelio vertinimas 8](#_Toc168070319)

[4. Gauti rezultatai 9](#_Toc168070320)

[5. Išvados 11](#_Toc168070321)

# Užduoties tikslas

Užduoties tikslas – sukurti programą, įgyvendinančią žaidimą „Kryžiukai-nuliukai“, taikant daugiasluoksnį tiesioginio sklidimo neuroninį tinklą.

# Žaidimo taisyklės ir modelio sprendžiamas uždavinys

Tai yra paprastas, nesunkus ir dviem žaidėjams skirtas žaidimas. Žaidimo laukas: 9 langelių (3×3) lentelė. Vienas žaidėjas lentelės langelius žymi X (kryžiukais), antrasis – 0 (nuliukais). Žaidėjai eina paeiliui. Laimi tas, kuris pirmas užpildo pilną eilutę, stulpelį ar įstrižainę savo ženklais.

Išmokyti neuroninį tinklą žaisti šį žaidimą yra sprendžiamas **klasifikavimo uždavinys**. Neuroninis tinklas turi nuspręsti, koks veiksmas (t.y., kur padėti nuliuką ar kryžiuką) yra geriausias esamoje žaidimo būsenoje. Tai galima laikyti klasifikavimo problema, nes iš visų galimų veiksmų (paprastai nuo 1 iki 9, priklausomai nuo tuščių langelių skaičiaus) reikia pasirinkti vieną.

# Žaidimo neuroninio tinklo realizacija

## Naudojami programiniai įrankiai ir resursai

Neuroninio tinklo realizacijai buvo pasirinkta naudoti Python programavimo kalbą su neuroniniams tinklams skirta biblioteka TensorFlow. Ši biblioteka ne tik suteikia reikalingą funkcionalumą kurti ir mokinti neuroninius tinklus, bet ir prisitaikyti prie techninės įrangos ir skaičiavimo resursų. Mokinti neuroninio tinklo modeliui reikia nemažai skaičiavimo resursų, todėl naudoti kompiuterio procesorių šiam tikslui nėra efektyvu ir tai gali užtrukti daug laiko. Geriausia tai atlikti pasitelkiant kompiuterio grafinį procesorių (GPU). Tai leido vykdyti visą mokymo procesą asmeniniuose kompiuteriuose.

## Neuroninio tinklo architektūra

Šiam uždaviniui atlikti buvo naudojamas daugiasluoksnis tiesioginio sklidimo neuroninis tinklas. Kadangi žaidimas yra paprastas, tam nereikėjo turėti sudėtingą architektūrą. Geriausią veikimą parodė architektūra parodyta kode esančiame **1 pav.**.

A computer code on a black background

Description automatically generated

**1 pav.** Neuroninio tinklo modelio architektūra

Pateiktas daugiasluoksnis tiesioginio sklidimo neuroninio tinklo modelis buvo sukurtas naudojant *Sequential* klasę iš Keras bibliotekos. Modelis susideda iš trijų sluoksnių: pirmasis sluoksnis yra *Flatten* sluoksnis, kuris paverčia 3x3 įėjimo matricą į 9 elementų vektorių, antrasis sluoksnis yra pilnai jungus (angl. fully-connected layer) sluoksnis su 36 neuronais ir *ReLU* (angl. Rectified Linear Unit) aktyvacijos funkcija, o trečiasis sluoksnis taip pat yra pilnai jungus sluoksnis su 36 neuronais ir ReLU aktyvacijos funkcija. Paskutinis sluoksnis yra išvesties sluoksnis su 9 neuronais ir tiesine aktyvacijos funkcija.

Modeliui buvo pritaikytas *Adam* (angl. Adaptive Moment Estimation) optimizavimo metodas ir *MSE* (angl. Mean Square Error) nuostolių funkcija, su nustatytu mokymosi greičiu lygiu *0,1*. Šis modelis parodė geriausius rezultatus paprastam žaidimui, kuriam nereikėjo sudėtingos architektūros, ir buvo naudojamas tolimesniems tyrimams bei eksperimentams.

## Apmokymo strategija

Apmokymo strategija žaidimo kryžiukų-nuliukų neuroniniam tinklui buvo paremta savarankiško mokymosi metodu, kai modelis treniruojasi žaisdamas prieš save patį, naudojant agentą, kuris prižiūri mokymo procesą. Šiam atvejui buvo būtent panaudotas Q-learning algoritmas.

### Q-learning algoritmas

Q-learning yra skatinamojo mokymosi (angl. reinforcement learning) algoritmas, naudojamas mokytis siekiant pasiekti didžiausią atlygį.

**Algoritmo veikimas:**

1. **Q-Vertės:** Q-learning algoritmas palaiko Q-vertės (kokybės vertės) funkciją , kuri atspindi tikėtiną sukauptą atlygį atliekant veiksmą būsenoje .
2. **Veiksmų Pasirinkimas:** Agentas renkasi veiksmus remdamasis Q-vertėmis. Dažnai naudojama -godumo strategiją, kuri parinka vieną iš šių veiksmų:
   1. Su tikimybe agentas pasirinks atsitiktinį veiksmą (tyrinėjimas).
   2. Su tikimybe agentas pasirinks veiksmą, turintį aukščiausią Q-vertę.
3. **Q-Funkcijos Atnaujinimas:** Po kiekvieno veiksmo agentas atnaujina Q-vertes pagal gautą atlygį ir būsimos būsenos vertę. Atnaujinimas vykdomas pagal formulę:

### Duomenų generavimas

Kadangi kryžiukų-nuliukų žaidimas yra paprastas ir lengvai simuliuojamas, mokymo duomenys buvo generuojami dinamiškai, modeliams žaidžiant tarpusavyje. Tai leido išvengti poreikio turėti iš anksto paruoštą didelį duomenų rinkinį.

### Aplinka

Aplinka buvo sukurta naudojant *TicTacToe* klasę, rodomą **2 pav.**, kuri simuliuoja žaidimo lentą ir taisykles. Ši klasė leidžia atlikti veiksmus, tikrina jų validumą ir nustato, kada žaidimas baigiasi, grąžindama atitinkamus atlygius.

A computer screen shot of a program code

Description automatically generated

**2 pav.** TicTacToe klasės struktūra

### Modelio mokymas

Modelio treniravimo procesas buvo įgyvendintas naudojant savarankiško žaidimo, **3 pav.** esančią funkciją *self\_play*, kur modelis treniruojasi žaisdamas prieš save. Treniravimo metu modelis naudojo Q-learning algoritmą, kurio metu agentas (modelis) mokosi optimalios strategijos, atnaujindamas savo žinias po kiekvieno žingsnio.

A computer screen shot of a program code

Description automatically generated

**3 pav.** Žaidimo prieš save funkcijos realizacija

### Modelio vertinimas

Po treniravimo modelis buvo vertinamas žaidžiant prieš atsitiktinį agentą. Vertinimo metu buvo stebimos pergalės, pralaimėjimai ir lygiosios, siekiant nustatyti modelio efektyvumą.

A computer screen shot of a program code

Description automatically generated

**4 pav.** Modelio vertinimo funkcijos realizacija

**4 pav.** esanti funkcija įvertino rezultatus, kad modelis galėtų išmokti optimalias žaidimo strategijas ir efektyviai žaisti kryžiukų-nuliukų žaidimą.

# Gauti rezultatai

Rasti geriausią modelio variantą, buvo atlikti modelio modelių apmokymai, kiekvienam modeliui skiriant sužaisti 500 žaidimų prieš save. Kadangi mokymas išnaudoja daug laiko, buvo pasirinktas nedidelis sužaisti skirtų žaidimų skaičius. Gauti rezultatus buvo skirta jau anksčiau aprašyta ir **4 pav.** pavaizduota funkcija, kuri įvertina santykį tarp laimėtų, pralaimėtų ir baigtų lygiosiomis žaidimų. Naudojant skatinamąjį mokymąsi sunku yra įvertinti jo tikslumą ir reikšmę, todėl buvo pasirinkta vertinti būtent šiuos santykius.

**1 lentelė.** Mokymo metu sužaistų žaidimų santykis

|  |  |
| --- | --- |
| Laimėjimai: | 55% |
| Pralaimėjimai: | 45% |
| Lygiosios: | 0% |

**1 lentelė** gauti rezultatai parodo, kad po 500 sužaistų su savimi žaidimų geriausias rastas modelis jau pasiekė 55% laimėjimų santykį. Tai rodo, kad toliau mokinant modelį galima pasiekti žymiai geresnius rezultatus. Tačiau vien šios lentelės neužtenka, kad galima būtų lyginti, kiek vienas modelis geresnis už kitą, nes galiausiai jie visi gali pasiekti tą patį laimėjimų santykį sužaidžiant panašų žaidimų skaičių.

Mokymosi greičio palyginimui ir santykių nuo žaidimų skaičiaus stebėjimui buvo sukurta papildoma klasė renkanti mokymo metrikas (**5 pav.**).

A screen shot of a computer program

Description automatically generated

**5 pav.** Metrikų stebėjimo klasės realizacija

Naudojant šią klasę yra sugeneruojamas grafikas, kuris būtent ir parodo rezultatus.

A graph of a number of people

Description automatically generated with medium confidence

**6 pav.** Santykių priklausomybė nuo sužaistų žaidimų

Pagal **6 pav.** ir **1 lentelė** gautus rezultatus galima pastebėti, kad nei vienas iš sužaistų žaidimų nebuvo baigtas lygiosiomis. Šis rezultatas gavosi, būtent tik mokinant šį geriausią gautą modelį, bet ne su kitais modeliais. Taip pat galima pastebėti, kad nors mokymosi progresas yra ne toks greitas, modelis vis tiek toliau mokosi ir daro vis mažiau klaidų.

# Išvados

Šiame projekte buvo sėkmingai sukurta kryžiukų-nuliukų žaidimo programa, naudojant daugiasluoksnį tiesioginio sklidimo neuroninį tinklą. Remiantis gautais rezultatais, galima padaryti keletą svarbių išvadų:

1. **Neuroninio tinklo veikimas:** Sukurtas neuroninio tinklo modelis, susidedantis iš trijų sluoksnių, parodė gerus rezultatus žaidžiant kryžiukų-nuliukų žaidimą. Naudojant paprastą architektūrą ir aktyvacijos funkcijas, modelis sugebėjo mokytis ir priimti teisingus sprendimus žaidimo metu.
2. **Apmokymo strategija:** Naudota savarankiško mokymosi strategija su Q-learning algoritmu leido modeliams mokytis žaisdami prieš save. Tai efektyvus būdas generuoti mokymo duomenis ir užtikrinti, kad modelis nuolat tobulėtų.
3. **Mokymosi progresas:** Nepaisant to, kad mokymosi progresas nebuvo greitas, modelis parodė gebėjimą mokytis ir daryti vis mažiau klaidų ilgalaikėje perspektyvoje. Tai rodo, kad toliau mokinant modelį galima pasiekti dar geresnių rezultatų, tačiau tam reikia daugiau laiko.

Nors gauti rezultatai yra perspektyvūs, yra galimybių toliau tobulinti modelį. Galima eksperimentuoti su sudėtingesnėmis architektūromis, kitomis mokymosi strategijomis ar hiperparametrų optimizavimu, siekiant dar didesnio modelio efektyvumo. Apskritai, šis projektas parodė, kad naudojant daugiasluoksnius neuroninius tinklus ir Q-learning algoritmą, galima efektyviai ir nesunkiai išmokyti modelius žaisti kryžiukų-nuliukų žaidimą.