Vysoké učení technické v Brně
Fakulta informačních technologií

Projekt do předmětu SUI

Umělá inteligence pro hru Dice Wars

Aleš Ondráček (xondra51)
Pavel Nováček (xnovac16)
Tomáš Willaschek (xwilla00)

27. prosince 2020

1 Úvod

Naše umělá inteligence využívá neuronovou síť pro lokální predikce a navíc je doplněna o prohledávání stavového prostoru, které má za úkol maximalizovat skóre hráče pomocí propojování menších regionů s regionem největším.

Umělá inteligence při vyhodnocení tahu postupuje následovně:

1. Umělá inteligence se pokusí propojit největší region s ostatními regiony a tím co nejvíce zvětšit hráčovo skóre.
2. Pokud není nalezen takový útok, který by vedl na sérii útoků, jež by dané regiony propojila, pak neuronová síť ohodnotí všechny možné útoky a nejlépe ohodnocený útok dosahující námi zvolené hranice je vykonán.
3. Pokud žádný z proveditelných útoků nedosahuje této hranice, pak se předá tah dalšímu hráči. Jinak se pokračuje bodem 1.

2 Řešení

Pro vytvoření umělé inteligence (dále jen UI) bylo potřeba provést několik úkonů. Začali jsme získáváním trénovacích dat a jejich předzpracováním pro budoucí neuronovou síť. Jakmile byla data připravená, pokračovali jsme sestrojením, natrénováním a vyladěním neuronové sítě. Nakonec jsme umělou inteligenci doplnili o algoritmus pro prohledávání stavového prostoru za účelem propojení největšího regionu hráče s jeho ostatními regiony.

2.1 Prohledávání stavového prostoru

Pro část, která má za úkol propojování regionů, jsme zvolili jednoduché prohledávání stavového prostoru. Nejprve získáme všechny možné útoky, ve kterých má naše UI vyrovnaný nebo vyšší počet kostek jak napadené území. Poté zjistíme, jak by se pro každý z těchto útoků změnilo skóre hráče. Podle nárůstu skóre pak vybere ten nejoptimálnější útok. Toto vyhledávání můžeme provádět do námi zadané hloubky.

Po několika experimentech jsme zjistili, že nejlepší výsledky naše UI podává v případě, kdy tento stavový prostor prohledáváme pouze do hloubky 1.

2.2 Neuronová síť (NN)

Neuronová síť byla implementována pomocí knihovny `Keras`, která vytváří jednoduché API pro definice a učení NN. Ze začátku jsme zvolili topologii NN náhodně, protože jsme nevěděli, jak by měla daná topologie vypadat. Postupem času, když už jsme měli dostupná kvalitní trénovací data, tak jsme topologii postupně upravovali podle následujících parametrů:

- **Rychlost učení** – větší síť se rychleji učí, ale na druhou stranu počítá víc operací a čas je v tomto projektu klíčový.
- **Velikost vstupního vektoru** – sledováním délky herního času bylo zjištěno, že vektor 11 vstupů zabírá podstatně více času než menší vektory.
- **Aktivační funkce** – byla zvolena nejprve jako kombinace `relu` a `sigmoid`, poté však došlo k nepochopení jednoho z informačních zdrojů, tudíž všechny `relu` byly nahrazeny funkcí `sigmoid`. Tato změna ovšek vrací velmi přesvědčivé výsledky, proto již zůstala jako finální.

Výsledná NN vypadá následovně:

- **1. vrstva** – 30 neuronů, aktivační funkce sigmoid;
- **2. vrstva** – 20 neuronů, aktivační funkce sigmoid;
- **3. vrstva** – 10 neuronů, aktivační funkce sigmoid;
- **4. vrstva** – 1 neuron, aktivační funkce sigmoid.

Za optimalizátor jsme zvolili `Adam`a. Ten byl zvolen z toho důvodu, že se síť naučila poměrně rychle a dosahovala dobrých výsledků. K tomuto optimalizátoru a následné metrice přesnosti byla vhodně zvolena velikost dávky, kde nám nejlépe vycházela velikost `150`.

Metrika pro přesnost výsledků sítě nám ze začátku dělala problémy, protože síť byla schopna vyhodnotit výsledky s přesností v řádů tisícin, někdy ještě hůř. Tento problém byl způsoben tím, že trénovací data mají poměrně velký počet desetinných míst. Při vytvoření vlastní metriky, která umožňovala odchylku výsledku `0.05` však přesnost NN vyskočila nad 90 %, čímž byl problém vyřešen.

2.2.1 Trénovací data

Trénovací data jsme získali tak, že jsme si zahráli několik desítek her a z každé hry jsme ukládali vstupní vektor, který je definovaný dále. Tento vektor se počítal pro tah, který zvolil člověk, ale také pro ostatní možné tahy. Dále jsme vytvořili evaluační funkci, která ohodnotila jednotlivé položky vektoru a následně ještě vylepšila hodnotu pro takové tahy, které člověk opravdu vybral. Tuto funkci jsme následně ladili tak, ať NN naučená těmito daty vyhrává co nejvíce her. Ze začátku jsme

vyhrávali jen zlomek, ale při ukončování optimalizace této funkce jsme postupně poráželi všechny dostupné umělé inteligence, což značilo, že tato optimalizace opravdu funguje.

Vznik množiny trénovacích dat byl z počátku takový tip, které hodnoty by se nám vlastně mohly hodit. Následně jsme zkoušely některé z nich vynechat a zahrát si několik her. Po několika experimentech jsme zjistili, že nejoptimálnější vstupní vektor NN je následující:

- pravděpodobnost úspěchu útoku
  (`successful_attack_p`)
- pravdivostní hodnota vyjadřující jestli vedeme útok z největšího regionu
  (`attacker_max_regio_flag`)
- pravdivostní hodnota vyjadřující jestli vedeme útok na největší region daného hráče
  (`defender_max_regio_flag`)
- procento zaplnění regionu kostkami, z nějž je útok veden
  (`attacker_region_occupancy`)
- procento zaplnění regionu kostkami, jehož pole je napadeno
  (`defender_region_occupancy`)
- procento kostek vlastněných útočníkem
  (`attacker_dice_proportion`)
- procento kostek vlastněných obráncem
  (`defender_dice_proportion`)
- procento polí vlastněných útočníkem
  (`attacker_area_proportion`)
- procento polí vlastněných obráncem
  (`defender_area_proportion`)
- velikost skóre obránce po provedení útoku v procentech vůči původnímu skóre
  (`enemy_score`)
- počet nepřátelských polí, které sousedí s polem, z něhož je útok veden
  (`count_of_enemy_neighbours`)

Původně jsme chtěli do vstupního vektoru zahrnout i velikost rezervy kostek. Ta se však nakonec jevila jako nedůležitá a neměla valný vliv na výsledky NN. Kvůli úspoře času při vyhodnocování NN byl tento parametr ze vstupního vektoru odebrán.

2.2.2 Zasazení do kontextu umělé inteligence

Pokud při prohledávání stavového prostoru není nalezen žádný další útok vedoucí na propojení regionů, pak dochází k vyhodnocení dalších tahů pomocí NN.

Všechny možné útoky jsou nejdříve ohodnoceny NN přičemž, jestliže se při ohodnocování narazí na útok, jehož ohodnocení je větší než 75 %, pak je tento útok vykonán okamžitě, protože při této hranici je natolik významný, že je účelně upřednostněn.

Po ohodnocení všech útoků je vybrán ten, jenž má největší ohodnocení. Takto vybraný útok musí mít ohodnocení alespoň 51 %, aby se provedl. Tato hranice byla určena experimentálně a pro různé hry je často více vyhovující její o trochu zvýšená hodnota, proto hodnota 51 % je výchozí a pokaždé, když se prohrají dva útoky za sebou, tak je tato hranice zvýšena o 1 %, ale maximálně do hodnoty 57 %. UI se tedy snaží o přesnější aproximaci této číselné hranice během hry. Navíc, v případě hry dvou hráčů je lepší větší agresivita UI a tím pádem více vyhovuje nižší počáteční hodnota této hranice, a proto byla opět experimentálně zvolena hodnota 46 % a i pro tuto hodnotu platí inkrementace v případě, kdy dojde k prohrání dvou útoků v řadě.

Pokud se v aktuálním kole provedou méně než 3 tahy a přitom již další možné útoky nesplňují ani jednu z uvedených číselných hranic, pak se přistupuje k vyhodnocování dalších podmínek, které řeší některé hraniční případy. Naše NN vrací lepší ohodnocení pro útoky, které protihráči pokud možno nejvíce uškodí, tj. seberou mu nejvíce kostek, a proto se upřednostňují tahy, kdy se útočí osmi kostkami na sedm kostek apod. Proto tahy, ve kterých se útočí například osmi kostkami na jednu kostku, nebo naopak tahy, ve kterých se útočí osmi kostkami na osm kostek, dostanou příliš malé ohodnocení, a proto by nebyly nikdy vykonány, což by mohlo vést na prohru z důvodu dlouhého čekání a neútočení hráčem. Tyto dodatečné podmínky pak řeší právě tyto případy, a proto se nedívají na ohodnocení NN, ale pouze na počet kostek v poli útočníka a počet kostek v poli obránce.

Nakonec, jestliže se provede více než 5 útoků s pomocí NN, pak všechny další útoky jsou provedeny pouze pokud dosáhly ohodnocení alespoň 65 %. Při sledování her bylo provedeno více než 5 útoků výjimečně, proto při překročení této hranice jsou provedeny pouze útoky, jejichž úspěšnost je víceméně zaručená.

3 Zhodnocení výsledků

Na následujících obrazcích jsou výsledky experimentů s prohledáváním stavového prostoru, pomocí kterého naše UI propojuje regiony. Dle očekávání přidáním tohoto prohledávání dosahuje naše UI podstatně lepších výsledků. Zajímavé ovšem je, že pokud prohledáváme stavový prostor do hloubky větší než 1, začne *winrate* opět klesat. Z toho důvodu jsme zvolili jako hloubku prohledávání 1.

V průběhu vytváření NN jsme narazili na stav, kdy byla síť přeučená. Tento stav se projevoval snížením počtu výher např. z 50 % na 40 %. Řešením tohoto problému bylo snížit počet epoch. Při zkoušení různého počtu epoch jsme došli k závěru, že nejvíce výher proti dostupným UI má naše NN při 80 % přesnosti.

```
(env-sui) osboxes@osboxes:~/Documents/sui3/SUI/dicewars-master$ python3 ./scripts/dicewars-tournament.py -r -g 4 -n 100 -b 101 -s 1337 -l ./logs --debug
.         .     % winrate [  .  / .   ] dt.rand  dt.sdc   dt.ste   dt.wpm_c xwilla00 xlogin00
xwilla00 42.69 % winrate [ 111 / 260 ] 42.1/164 35.8/148 40.5/148 46.8/156 42.7/260 47.6/164
dt.ste   42.42 % winrate [ 112 / 264 ] 47.0/164 45.0/160 42.4/264 40.0/160 41.2/148 38.8/160
dt.wpm_c 26.14 % winrate [ 69  / 264 ] 28.1/160 29.3/164 24.4/160 26.1/264 21.8/156 27.0/152
dt.sdc   22.79 % winrate [ 62  / 272 ] 26.6/184 22.8/272 20.0/160 18.9/164 20.9/148 26.9/160
xlogin00 9.09  % winrate [ 24  / 264 ] 9.0/156  10.0/160 8.8/160  10.5/152 7.3/164  9.1/264
dt.rand  7.97  % winrate [ 22  / 276 ] 8.0/276  11.4/184 4.3/164  6.9/160  6.7/164  10.3/156
```

Obrázek 1: Výsledek turnaje bez prohledávání stavového prostoru.

```
.         .     % winrate [  .  / .   ] dt.rand dt.sdc   dt.ste   dt.wpm_c xwilla00 xlogin00
xwilla00 45.59 % winrate [ 62 / 136 ] 51.2/80 51.2/80  34.5/84  43.4/76  45.6/136 47.7/88
dt.ste   36.76 % winrate [ 50 / 136 ] 38.8/80 34.1/88  36.8/136 35.0/80  36.9/84  39.5/76
dt.wpm_c 26.56 % winrate [ 34 / 128 ] 26.3/76 27.6/76  26.2/80  26.6/128 26.3/76  26.3/76
dt.sdc   20.59 % winrate [ 28 / 136 ] 23.8/84 20.6/136 22.7/88  19.7/76  12.5/80  23.8/80
xlogin00 12.12 % winrate [ 16 / 132 ] 13.2/76 10.0/80  15.8/76  13.2/76  9.1/88   12.1/132
dt.rand  7.58  % winrate [ 10 / 132 ] 7.6/132 9.5/84   5.0/80   10.5/76  6.2/80   6.6/76
```

Obrázek 2: Výsledek turnaje při prohledávání stavového prostoru do hloubky 1.

```
(env-sui) osboxes@osboxes:~/Documents/sui4/SUI/dicewars-master$ python3 ./scripts/dicewars-tournament.py -r -g 4 -n 50 -b 101 -s 1337 -l ./logs --debug
.         .     % winrate [  .  / .   ] dt.rand dt.sdc   dt.ste   dt.wpm_c xwilla00 xlogin00
xwilla00 42.42 % winrate [ 56 / 132 ] 42.5/80 41.2/68  41.2/80  44.0/84  42.4/132 42.9/84
dt.ste   38.64 % winrate [ 51 / 132 ] 47.5/80 38.8/80  38.6/132 36.8/76  35.0/80  35.0/80
dt.wpm_c 29.41 % winrate [ 40 / 136 ] 25.0/76 30.7/88  28.9/76  29.4/136 28.6/84  33.3/84
dt.sdc   21.97 % winrate [ 29 / 132 ] 25.0/80 22.0/132 17.5/80  21.6/88  17.6/68  27.5/80
dt.rand  9.09  % winrate [ 12 / 132 ] 9.1/132 12.5/80  6.2/80   7.9/76   6.2/80   12.5/80
xlogin00 8.82  % winrate [ 12 / 136 ] 11.2/80 8.8/80   8.8/80   7.1/84   8.3/84   8.8/136
```

Obrázek 3: Výsledek turnaje při prohledávání stavového prostoru do hloubky 2.

```
(env-sui) osboxes@osboxes:~/Documents/sui/SUI/dicewars-master$ python3 ./scripts/dicewars-tournament.py -r -g 4 -n 50 -b 101 -s 1337 -l ./logs
.         .     % winrate [  .  / .   ] dt.rand dt.sdc   dt.ste   dt.wpm_c xwilla00 xlogin00
xwilla00 50.00 % winrate [ 66 / 132 ] 48.8/80 44.1/68  47.5/80  57.1/84  50.0/132 51.2/84
dt.ste   40.91 % winrate [ 54 / 132 ] 48.8/80 43.8/80  40.9/132 34.2/76  38.8/80  38.8/80
dt.wpm_c 25.00 % winrate [ 34 / 136 ] 25.0/76 28.4/88  22.4/76  25.0/136 21.4/84  27.4/84
dt.sdc   18.94 % winrate [ 25 / 132 ] 21.2/80 18.9/132 16.2/80  19.3/88  11.8/68  25.0/80
dt.rand  9.09  % winrate [ 12 / 132 ] 9.1/132 12.5/80  6.2/80   7.9/76   6.2/80   12.5/80
xlogin00 6.62  % winrate [ 9  / 136 ] 7.5/80  8.8/80   6.2/80   6.0/84   4.8/84   6.6/136
```

Obrázek 4: Výsledky turnaje s finální umělou inteligencí.