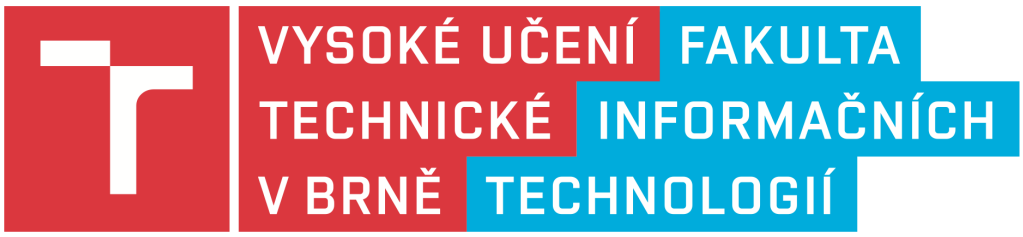
Vysoké učení technické v Brně
Fakulta informačních technologií



Projekt do předmětu SUI

Umělá inteligence pro hru kostek

Autoři:
Daniel Bubeníček (xbuben05)
Peter Dragúň (xdragu01)
Jan Beran (xberan43)

30. prosince 2020

Obsah

1 Úvod a zadání projektu

Zadáním projektu bylo navrhnout umělou inteligenci pro hru Dicewars[1]. Kromě samotného agenta bylo cílem zadání implementovat i prohledávání stavového prostoru.
Dicewars je strategická hra, ve které se hráči postupně střídají ve svých tazích. Princip hry je takový, že se hráč snaží útočit ze svých políček na políčka sousední. Každé políčko přitom obsahuje určitý počet kostek. Při útoku si útočník i obránce házejí všemi kostkami, které jsou v tu chvíli dostupné na daných dvou políčkách. Vyhrává hráč s vyšším hodem, v případě remízy vyhrává obránce.
Po úspěšném útoku jsou všechny obráncovy kostky odstraněny a na jejich místo se přesunou všechny kostky z útočníkova políčka až na jednu. Pokud útok úspěšný nebyl, obráncovy kostky zůstanou nedotčeny, útočník ovšem přijde o všechny kostky kromě jedné.
Cílem hry je poté dobýt celou hrací plochu, která je unikátní pro každou hru.

2 Průběh vývoje umělé inteligence

Nejprve byl vyvinut jednoduchý pravděpodobnostní agent (bez prvků umělé inteligence), který pouze počítal pravděpodobnost, s jakou bude schopen vyhrát útok. Tento agent porážel všechny „náhodné“ umělé inteligence a v turnajích se umisťoval zhruba v polovině. Tento agent posléze sloužil jako baseline.
V následujícím kroku již byl vytvořen skutečný agent, založený na diskriminativním klasifikátoru z domácího úkolu. Pro jeho trénování bylo využito celkem 11 vlastností, které popisují stav hry:

- Počet regionů daného hráče
- Velikost největšího regionu daného hráče
- Počet kostek daného hráče
- Průměrný počet kostek ostatních hráčů
- Počet hráčů „naživu“
- Délka hranic největšího regionu ve hře daného hráče
- Celková délka hranic daného hráče
- Celková pravděpodobnost udržení území na všech hranicích
- Pravděpodobnost udržení hranic největšího území
- Průměrný počet kostek na hranicích

2.1 Způsob získávání dat a trénování klasifikátoru

Data byla získávána z agentů `dt.ste, dt.stei, dt.sdc, dt.wpm_c` pomocí skriptu `data_collector.py`. Agenti přitom byli upraveni, aby logovali data v námi určeném formátu (o správné logování se starají třídy `Log` a `Helper` v souboru `log.py`).
Bylo odehráno cca 10000 her mezi výše vypsanými umělými inteligencemi a během těchto her byly logovány všechny vlastnosti popsané výše. Pokud daný agent prohrál, jeho stavy byl označeny jako negativní, v opačném případě jako pozitivní.

[1] `https://github.com/ibenes/dicewars`

Následně došlo k úpravě dat na trénovací a validační pomocí skriptu `splitData.sh`. Na těchto datech již mohlo započít trénování, o které se staral skript `trainer.py`. Model nakonec byl uložen jako fea11.pt. Z tohoto souboru je také zpětně načítán. Tento natrénovaný klasifikátor poté dosahoval zhruba 85% úspěšnosti odhadnutí, zda daný tah vedl k výhře či nikoli. Při evaluaci mezi agenty `dt.ste`, `dt.wpm_c`, `dt.sdc`, `dt.rand`, `xlogin00` a naším agentem `xberan43` dosahoval náš agent 53% úspěšnosti při 10000 her.
Jelikož pro získávání dat byly upraveny všechny umělé inteligence, není postup jednoduše reprodukovatelný, ale pro natrénování modelu na vzorku dat lze použít následující postup:

- Připravit data pomocí `./splitData.sh` ve stejné složce jako data. Tím vzniknou soubory positives.trn, positives.val, negatives.trn a negatives.val.
- Příkazem `python3 trainer.py` spustit trénování modelu. Ten bude následně automaticky uložen jako fea11.pt.

2.2 Konečná podoba agenta

Jádrem agenta je diskriminativní klasifikátor, který klasifikuje stav hry na základě 11 vlastností, které byly popsány výše. Úspěšnost tohoto klasifikátoru se pohybovala okolo 85 %. Klasifikátor je využit na ohodnocení stavů hry po hypotetickém provedení daného tahu.
Během testování se ovšem ukázalo, že agent založený pouze na klasifikátoru si v některých situacích nepočínal ideálně. Níže budou popsány tyto situace a způsoby, jakým byl agent upraven.
První situací je stav, kdy se klasifikátor dostal do lokálního maxima (ohodnocení současného stavu bylo natolik dobré, že všechny tahy by vedly k horšímu ohodnocení). Tato situace je řešena několika způsoby:

- Snížením ohodnocení aktuálního stavu. Ohodnocení současného stavu je násobeno hodnotou 0.7[2] , čímž je docíleno, že se agent snaží přejít i do nepatrně horšího stavu. Tento způsob se aktivuje častěji uprostřed hry.
- Pokud agent nehraje po dobu 3 a více kol [3], vynuceně se provede tah s největší pravděpodobností úspěchu. Původně se experimentovalo i s verzí, kdy by při určitém počtu kol, kdy nebyl proveden žádný tah, aktivovala „lite verze“, zde ovšem vzácně docházelo k problémům, kdy se hra dostala do „statu quo“ a ani jeden z agentů nebyl schopen vyhrát. Nakonec by hra samozřejmě skončila, ovšem v situaci, kdy jedna hra trvala i více než 5 minut, se nám zdálo vhodné se tomuto chování vyhnout, i když teoreticky vede k horším výsledkům. Měření ovšem neukázalo významný pokles úspěšnosti. Tento způsob je častěji aktivován v patových situacích (zbývají dva vyrovnaní agenti, agent je zahnaný do rohu atp.).

Při nedostatku času je poté opět aktivována `lite` verze agenta. Hraniční čas byl zvolen na 0.3 s.
Agent v této podobě poráží všechny dodané agenty a jeho celkový winrate je cca 53 %. Oproti baseline tedy došlo k výraznému zlepšení.

2.3 Prohledávání stavového prostoru

Prohledávání stavového prostoru v plné míře implementováno. Stavový prostor se prohledává pouze na úrovni možných útoků námi vytvořeného agenta a hypotetického stavu hry po tomto útoku.
Důvodem je, že stavový prostor tak, jak byl námi definován, nebylo možné rozumně prohledávat. Agent spotřebovával všechen dostupný čas hned v prvním kole a v dalších kolech tedy hrál pouze pomocí „baseline“ algoritmu jakožto nouzového prostředku v případě nedostatku času.

[2] Tato hodnota byla zjištěna experimentálně, vizte tabulku, sloupec „Koeficient agresivity“
[3] Opět experimentálně zjištěná hodnota, v tabulce ve sloupci „Počet kol, po který se nehrálo“

Koeficient agresivity	Počet kol, po který se nehrálo	Úspěšnost v %
0.7	2	52.7
0.7	3	52.7
0.7	4	52.3
0.7	8	51.2
0.8	2	52.3
0.8	3	52.3
0.8	4	49.7
0.8	8	49.7
0.9	2	48.0
0.9	3	48.0
0.9	4	47.1
0.9	8	46.9

Tabulka 1: Tabulka obsahuje vyzkoušené hodnoty koeficientů agresivity a počtu kol, po který se nehrálo. Jako nejlepší kombinace se jeví koeficient 0.7 a 3 kola.

Prohledávání stavového prostoru by pravděpodobně bylo možné zefektivnit pomocí vhodné heuristiky, ale tento postup nebyl z důvodu časové tísně použit a prohledávání stavového prostoru tedy není použito.

3 Součásti odevzdání

Odevzdáván je jeden archiv se jménem `xberan43.zip`. Uvnitř se nachází soubory a složky popsané níže:

- Dokumentace jménem `xberan43.pdf`, stručně popisující tento projekt.
- Složka `xberan43`, obsahující:
  - Soubor `__init__.py`
  - Samotného agenta v souboru `xberan43.py`
  - Soubor `fea11.pt` obsahující data s natrénovanou umělou inteligencí
  - Soubor `trainer.py`, obsahující skript určený pro trénování umělé inteligence.
  - Soubor `log.py` obsahující pomocné třídy pro logování a získávání dat.
- Skripty `splitData.sh` a `data_collector.py` pro získání a přípravu dat pro trénování umělé inteligence (pro plnou funkcionalitu vyžaduje skript data_collector upravené všechny agenty, odevzdávám je tedy pouze pro úplnost).
- Vzorek dat v souborech `positives` a `negatives`. Kompletní data byla příliš rozsáhlá a svou velikostí neumožňovala odevzdání.

4 Závěr

Zadání bylo částečně splněno. Byl vytvořen agent, který se pravidelně v turnaji umisťuje na prvním místě v turnaji umělých inteligencí při hře 4 hráčů. Prohledávání stavového prostoru nebylo implementováno.