VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Projekt do předmětu SUI

AI pro hru Dicewars

25. prosince 2020

Kateřina Fořtová (xforto00)
Jakub Kolb (xkolbj00)
Matej Kolesár (xkoles07)
Maroš Orsák (xorsak02)

1 Úvod a motivace

Tato dokumentace se zabývá průběhem implementace a výsledky AI pro projekt do předmětu SUI (Umělá inteligence a strojové učení). Úkolem bylo implementovat umělou inteligenci, která je založená na prohledávání stavového prostoru a strojovém učení pro hru Dicewars pro 2 až 4 hráče.

2 Obsah archivu

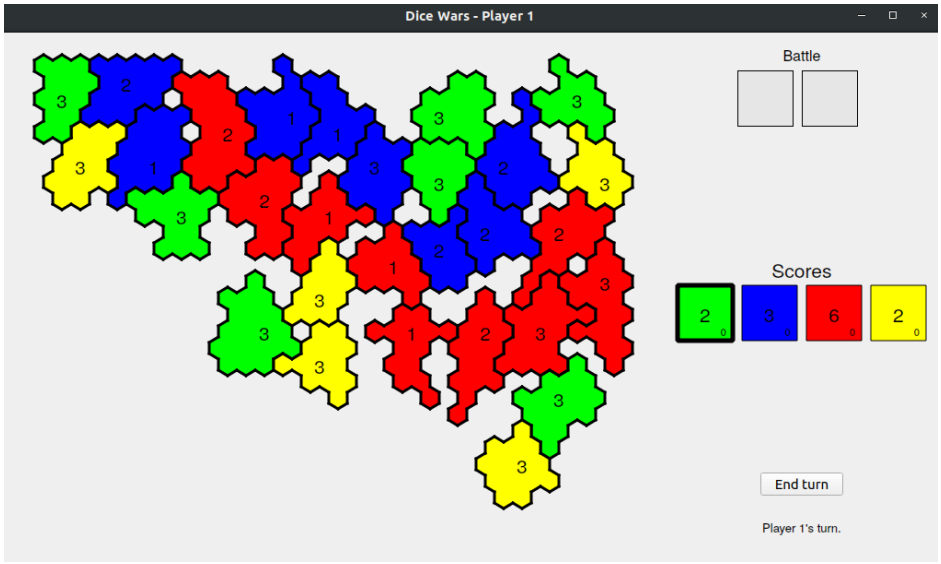
Archiv `xforto00.zip` obsahuje:

- Finálně odevzdávanou AI pojmenovanou loginem `xforto00`, koncipovanou do balíčku. V balíčku se mimo implementace finální AI `xforto00` a logistické regrese nachází například složky s trénovacími a validačními daty, které využívá klasifikátor logistické regrese, příklad grafu přesnosti a loss natrénovaného modelu nebo tabulka s vyhodnocením turnajů.
- Pomocnou AI `trainAI`, koncipovanou do balíčku. Běh této AI umožňuje reprodukci a generování nových trénovacích nebo validačních vektorů a příslušných tříd, které využívá při logistické regresi `xforto00` AI využívající nejen prohledávání stavového prostoru, ale i strojové učení.
- Dokumentaci popisující řešení projektu.

3 Pravidla hry

Dicewars je hrou pro více hráčů. Naše implementace počítala se hrou 2 až 4 hráčů, přičemž finální turnajové vyhodnocování bude probíhat s variantou pro 4 hráče. Herní plocha je složená z polí, kde na každém z nich může ležet jedna až osm kostek. Hrající hráči se střídají v tazích. Hráč může útočit a nebo ukončit tah. Útok může být veden pouze z pole, se kterým dané území sousedí. Cílem hry je ovládnout a zabrat celou herní plochu.

Hráč může útočit z vlastního pole, které má aspoň dvě kostky. Hráč i jeho nepřítel, který dané území, na které se útočí, zatím vlastní, hodí všemi svými kostkami, které jsou umístěné na daném poli a je pak následně porovnán jejich součet. Pokud je součet hráče vyšší a útok je tedy úspěšným, tak získá od nepřítele dané pole a i veškeré kostky na zabraném území. Hráč na nově dobyté území pak dál přesouvá až na jednu všechny své kostky z území, ze kterého útočil (kostky, se kterými před zabráním území házel). Pokud hráč neuspěje, tak území dále patří nepříteli a hráč, co chtěl území získat, ztrácí všechny kostky, až na jednu. Pokud je tah ukončen, tak se nalezne největší souvislá oblast hráče a ten obdrží počet kostek, který je roven počtu polí v jeho největším souvislém regionu. Tyto kostky se sečtou s kostkami v rezervě hráče (rezerva je na začátku každé hry u všech hráčů prázdná) a rozdělí náhodně mezi vlastněná pole. Pokud však mají všechna hráčova vlastněná území už maximální počet osmi kostek, tak jsou zbývající kostky uloženy do rezervy.



Obrázek 1: Ukázka hry Dicewars pro 4 hráče

4 Prohledávání stavového prostoru

Pro prohledávání stavového prostoru byla využitá modifikovaná verze WPM (Win Probability Maximization), která využívá různých zajímavých vlastností, které lze zjistit o hrajících hráčích. Využívá se tedy charakteristik aktuálního stavu hry a odhadu aktuální pravděpodobnosti výhry. Úkolem je maximalizovat pravděpodobnost vyhrání hry.

Součástí vektoru jsou dané vlastnosti pro hráče (`_player_`) a možného oponenta, který vlastní území, na které chceme zaútočit (`_oponent_`):

- aktuální celkové skóre hráče (`score_player_value`, `score_oponent_value`),
- aktuální celkový počet kostek hráče (`dice_player_value`, `dice_oponent_value`),
- aktuální počet vlastněných území hráčem (`owned_fields_player`, `owned_fields_oponent`),
- počet snadných území k útoku (`effortless_target_areas_sum_player`, `effortless_target_areas_sum_oponent`) – jedná se o sousední území, která mají menší počet kostek než náš hráč (je tedy velmi výhodné zaútočit),
- velikost největšího regionu hráče (`largest_region_player`, `largest_region_oponent`).

Prvně je iterováno přes všechny hrající hráče (`_player_`) a je provedena suma těchto pěti vlastností, uložení do listu `features` a vynásobení tohoto celého listu s náhodnými vahami (získáme hodnotu `win_probability`).

Dále je iterováno přes všechny možné útoky a jsou filtrovány ta území, která mají pravděpodobnost úspěšného útoku aspoň 50 procent a zároveň území, ze kterého hráč chce zaútočit, je součástí největšího regionu a nebo leží na daném území, ze kterého útočíme, osm kostek. Opět je iterováno přes všechny hrající hráče a pole `new_features` je modifikováno na základě pole `features`, možného území v největším regionu a také na základě spočítání pěti charakteristik i pro nepřítele (`_oponent_`), který území zatím vlastní. Opět spočítáme sumu daných vlastností. Modifikovaný list `new_features` je pak opět vynásoben náhodnými vahami (získáme hodnotu `new_win_probability`).

Útoky jsou poté seřazeny podle nejvyššího zlepšení, které bylo využito pro výběr území pro zaútočení pro trénovací AI (v odevzdávaném archivu balíček `trainAI`), která měla za úkol získat trénovací a validační data. Trénovací AI `trainAI` zaútočila na území s nejvyšším zlepšením z daných vyfiltrovaných útoků, AI `xforto00` využívající strojové učení nezaútočí nutně vždy na místo s nejlepším zlepšením, ale rozhodne

o nejlepším útoku z vyfiltrovaných na základě klasifikátoru logistické regrese, co využívá data získaná na základě běhu `trainAI`. Toto zlepšení je spočítáno následovně:

$$calculated\_improvement = new\_win\_probability - win\_probability \quad (1)$$

Pro uložení do vektoru byla hodnota `calculated_improvement` vynásobena konstantou 1000 díky své malé velikosti ve srovnání s ostatními deseti hodnotami obsaženými ve vektoru příznaků. Tedy výsledná hodnota uloženého zlepšení ve vektoru je:

$$calculated\_improvement\_mul = calculated\_improvement * 1000 \quad (2)$$

5 Získání trénovacích a validačních dat

Byla získána sada dat pro trénování i validaci. U trénovací AI `trainAI` bylo vždy zaútočeno na území, které mělo nejvyšší hodnotu `calculated_improvement`. Po provedení dalšího kola tahů hráčů bylo zjištěno, zda dané území stále patří našemu příslušnému hráči a nebo bylo nějakým oponentem odebráno v průběhu předchozího kola. Zjišťujeme tedy, zda se nám útok na dané území vyplatil a stále nám i po uplynutí kola území patří (třída 1) a nebo nepatří a získal ho jiný hráč (třída 0).

Vzniknou tedy následující soubory, které jsou klíčem k druhé části zadání – využití strojového učení:

- `trainingFeaturesWithImprovement.csv` – uložení trénovacího vektoru sestávající se z daných deseti vlastností a vypočítané hodnoty `calculated_improvement_mul`,
- `trainingClassesWithImprovement.csv` – uložení dané třídy na základě toho, zda nám území zůstalo do dalšího kola a nebo ho vlastní nepřítel,
- `validationFeaturesWithImprovement.csv` – uložení dat určených pro validaci v každé z epoch klasifikátoru provádějící strojové učení, tedy obdobně deset příslušných vlastností a hodnota `calculated_improvement`,
- `validationClassesWithImprovement.csv` – uložení příslušných tříd pro validační vektory.

Jelikož je balíček `trainAI` dostupný v odevzdávaném archivu, tak je možné tuto AI spustit a vygenerovat další data pro trénování nebo validaci (ukládáno do `./dicewars/ai/xforto00/valFiles/` nebo `./dicewars/ai/xforto00/trainFiles/`).

Trénovací data byla získána při hře 4 hráčů, kdy vždy trénovací AI `trainAI` hrála turnaj s těmito dalšími AI: `dt.rand`, `dt.sdc`, `dt.ste`, `dt.wpm_c` a `xlogin00`. Bylo získáno zhruba 2500 vektorů a AI hrála po 16 (počet desek) * 4 (počet hráčů) her. Validační data byla získána obdobným způsobem, avšak těchto vektorů je pouze okolo 500 a bylo hrány 4 * 4 hry.

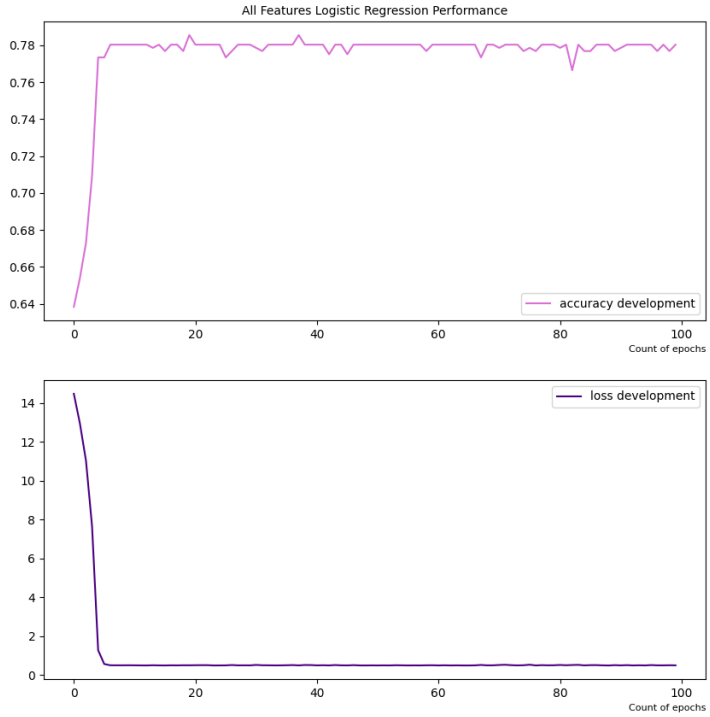
6 Strojové učení

Pro část zadání, kdy je nutno využít nějaký způsob strojového učení, byla využita logistická regrese. Pro implementaci byla využita knihovna PyTorch samozřejmě bez rozšíření CUDA a verzí pouze pro CPU. Trénovací data jsou rozdělena do tzv. minibatches. Zde je nutno klást důraz na skutečnost, že naše AI se musí před začátkem každé hry zkonstruovat do deseti sekund. Původně bylo počítáno s minibatches velikostí 32, ale následně bylo využito až vyšší číslo 128, protože často rychlost natrénování záležela na daném zařízení, na kterém byla hra spuštěna. Velikost minibatches lze snadno ve volání funkce pro natrénování modelu případně upravit.

Logistická regrese je modelem pro prediktivní analýzu. Můžeme využít sigmoid funkce, která je často využívanou ve strojovém učení a její výstup získává hodnoty mezi 0 a 1. Daná logistická regrese tedy počítá s celým vektorem příznaků, který obsahuje 11 vlastností (všechny hodnoty uvedeny ve výčtu ve třetí kapitole a vypočítanou hodnotu `calculated_improvement_mul`). Délka natrénování činí 100 epoch. Jako optimalizátor je využit Adam s koeficientem učení 0,01.

V každé z epoch probíhá vyhodnocení na validačních datech a pro finální testování je uložena kopie modelu s největší přesností. Je přitom nutné provést prahování výsledné predikované hodnoty na hodnotu 0 nebo 1 a následně porovnat s reálnou třídou. Po natrénování dosahuje přesnost modelu obvykle skoro 80 procent.

AI sloužící pro testování `xforto00` funguje jako umělá inteligence, která tedy po vyfiltrování zajímavých tahů provede predikci hodnoty třídy (0 nebo 1). Ze seznamu možných vyfiltrovaných útoků je pak tedy vybrán index tahu, který má hodnotu predikce nejblíže k hodnotě 1 (nejvyšší pravděpodobnost třídy 1) a na toto území námi implementovaná AI finálně zaútočí.



Obrázek 2: Ukázka vývoje přesnosti a loss při trénování logistické regrese

7 Testování implementované AI

Model procházel iteracemi vývoje, kdy v každé z nich byl testován na turnaji proti AI, které budou uvažovány při opravování projektu (`dt.rand`, `dt.sdc`, `dt.ste`, `dt.wpm_c` a `xlogin00`). Byly testovány pouze hry 2 až 4 hráčů. Výsledky šampionátů jsou uvedeny v tabulkách níže. Tyto turnaje byly hrány na počítači s operačním systémem Ubuntu 18.04.5 LTS a procesorem Intel® Core™ i5-8250U CPU @ 1.60GHz × 8. AI byla implementována v jazyce Python 3.6.9, který je dostupný implicitně v užívané verzi operačního systému Ubuntu. Otestována však byla i při spuštění verze Python 3.8, na kterém se projekt bude vyhodnocovat. Na tomto stroji rychlost natrénování modelu logistické regrese trvá zhruba tři sekundy, testování na jiných CPU dalších členů týmu také splní časový limit, tedy je splněn požadavek maximálně 10 sekund na konstrukci AI před každou hrou. Jedním z možných vyhodnocovacích procesorů je například Intel Xeon E5-2670 @ 2.60GHz, který je obecně výkonnějším[1], tudíž můžeme pro natrénování ponechat velikost minibatches 128. Pokud by však i tak byl problém s časovým limitem pro natrénování na daném CPU, tak velikost minibatches lze snadno upravit

[1] Viz https://www.cpubenchmark.net/compare/Intel-i5-8250U-vs-Intel-Xeon-E5-2670/3042vs1220.

ve volání funkce `train_all_fea_llr(nb_epochs, lr, batch_size, inputs, targets)` v konstruktoru AI `xforto00`. Při každém novém natrénování před hrou můžeme využít zobrazení grafů přesnosti a loss modelu logistické regrese.

Můžeme vidět, že naše AI se chová lépe při hrách většího počtu hráčů – zvláště při šampionátu čtyř hráčů, který bude při hodnocení projektu testován. Při hře čtyř hráčů se `xforto00` nejvíce díky svému Winrate blíží `dt.wpm_c`. Pro 100 i 250 desek zde bylo Winrate druhým nejlepším, pro hru 50 desek pak třetím nejlepším. Ačkoliv je Winrate při hře dvou hráčů spíše nižší, v hodnoceném turnaji pro 4 hráče má naše umělá inteligence výhodu a umisťuje se na přednějších příčkách. Nutno podotknout, že právě pro turnaj čtyř hráčů byla získána trénovací a validační data, které následně využívá klasifikátor logistické regrese a vybírá nejlepší možné útoky z daných vyfiltrovaných zajímavých tahů.

Number of Played Boards	**xforto00 Winrate [%]**	**Stats of All AIs in Tournament [%]**
250	**33.60**	dt.ste 40.67 xforto00 33.60 dt.wpm_c 33.33 dt.sdc 25.17 xlogin00 5.87 dt.rand 5.50
100	**31.75**	dt.ste 40.83 xforto00 31.75 dt.wpm_c 30.93 dt.sdc 30.00 dt.rand 6.85 xlogin00 5.51
50	**31.50**	dt.ste 43.10 dt.wpm_c 32.26 xforto00 31.50 dt.sdc 30.17 dt.rand 5.17 xlogin00 4.69

Tabulka 1: Testování AI xforto00 pro turnaj 4 hráčů

Number of Played Boards	**xforto00 Winrate [%]**	**Stats of All AIs in Tournament [%]**
100	**31.33**	dt.sdc 52.85 dt.wpm_c 46.15 dt.ste 41.67 dt.ste 41.67 dt.rand 20.00 xlogin00 10.83
50	**32.00**	dt.sdc 51.67 dt.wpm_c 49.12 dt.ste 41.67 xforto00 32.00 dt.rand 18.18 xlogin00 10.53

Tabulka 2: Testování AI xforto00 pro turnaj 3 hráčů

Number of Played Boards	xforto00 Winrate [%]	Stats of All AIs in Tournament [%]
100	**39.00**	dt.wpm_c 82.50 dt.sdc 70.00 dt.ste 60.00 dt.rand 57.50 xforto00 39.00 xlogin00 35.00
50	**43.00**	dt.wpm_c 70.00 dt.sdc 65.00 dt.rand 60.00 dt.ste 60.00 xforto00 43.00 xlogin00 30.00

Tabulka 3: Testování AI xforto00 pro turnaj 2 hráčů

8 Zhodnocení řešení projektu

Ačkoliv naše AI s pomocí WPM pečlivým způsobem dokáže vyfiltrovat zajímavé tahy, logistická regrese slouží jako přídavné rozhodnutí, které je natrénováno na turnaji pro čtyři hráče. Strojové učení je tedy do řešení projektu zakomponováno jako výběr z nejlepších vyfiltrovaných tahů, které získáme pro prohledání stavového prostoru. Přiložený tabulkový soubor ve složce `./aiStatistics` ukazuje výsledky turnajů v jednotlivých iteracích vývoje projektu. Nutno podotknout, že původně se trénovací AI `trainAI` jmenovala `xforto00` a testovaná (odevzdávaná a využívající strojového učení) `xforto00test` (nikoliv `xforto00` jako ve finální odevzdané verzi). Můžeme vidět, že ačkoliv se přidáním strojového učení Winrate pro 2 hráče spíše snížilo (původně se také filtrovaly i tahy s menší pravděpodobností úspěšného útoku než 50 procent), tak při finálním zhodnocení se Winrate pro hodnocenou hru pro 4 hráče zvýšilo od začátku po přidání strojového učení zhruba o 10 procent a AI se dokázala umístit na druhém místě. Toto umístění pro nás bylo klíčové, protože právě na turnaji pro 4 hráče se naše AI bude vyhodnocovat.

Logistickou regresi, která byla pro strojové učení využita, je výhodná svojí jednoduchostí implementace a snadnou pochopitelností. Dále pro nás také bylo důležité vybrat metody strojového učení a prohledávání stavového prostoru, které dokážou zachovat časové limity implementované ve hře nejen pro konstrukci samotné AI (kde probíhá natrénování modelu logistické regrese), ale i při samotné hře, kde se využívá postupně inkrementujících Fischer clock na základě průběhu hry. Při implementaci naší AI jsme na časové limity museli klást proto velký důraz, stejně jako na zkoumání chování a dodržení časových limitů na více zařízeních.

Uživatel si může vygenerovat grafy trénování modelu logistické regrese. Tento úsek je však nutné v `xforto00.py` odkomentovat, protože finální řešení by nemělo dle zadání zapisovat na disk. Dané grafy jsou ukládány do souboru pod názvem `learn_graph_lr.png`.

Pro ukázání generování dat pro strojové učení je v archivu pro ukázku dostupná i trénovací AI `trainAI`, která umožňuje získávat nové vektory a příslušné třídy určené k trénování nebo validaci, které následně využívá strojové učení u finální odevzdávané AI `xforto00`. Jedná se opět o pythonovský balíček (obdobně jako `xforto00`). AI `trainAI` tedy fungovala jako pomocná umělá inteligence, co nám právě umožnila získat trénovací a validační data s příslušnými třídami. Pokud je spuštěna právě `trainAI`, tak se do trénovacích nebo validačních souborů zapíší nové hodnoty. Tato AI je umístěna v archivu jako názorná ukázka toho, jak jsme získávali trénovací a validační data a jejich ohodnocené třídy.