VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Umělá inteligence a strojové učení

# Umělá inteligence pro Válku kostek

2. január 2022
Andrej Ježík (xjezik03), Branislav Dubec (xdubec01),
Tomáš Moravčík (xmorav41), Jindřich Březina (xbrezi21)

# 1 Úvod

Dice Wars je strategická hra, v ktorej sa hráči snažia expandovať svoje územie útokom na susedné polia počas svojho ťahu. Každé územie obsahuje počet kociek reprezentujúcich hráča a jeho silu. Cieľom hry je podmaniť si všetky územia a teda eliminovať všetkých protihráčov.
Projekt je klient-server implementácia bakalárskej práce na FIT VUT.
Github: https://github.com/ibenes/dicewars
Sample Data: https://jeziks.sk/nextcloud/index.php/s/jiaRsbe6gP3emzB

## 1.1 Hra

Hra je ťahovo založená pre 2 a viac hráčov s prvkom náhody. Hráči sa v danom poradí striedajú na ťahy. Ťah jedného hráča pozostáva z obmedzeného počtu akcií, ktoré sú útok, presun a ukončenie ťahu. Skóre sa počíta ako najvyšší počet susediacich území patriacich hráčovi. Každé územie môže mať 1 až 8 kociek, pričom povinne musí byť aspoň 1 kocka prítomná v poli.

Presun nastáva premiestnením kociek z vlastných polí s 2 a viac kockami do susedného vlastného pola. Po presune v originálnom poli zostane najmenej 1 kocka a v cieľovom môže byť maximálne 8 kociek.

Útok prebieha presunutím kociek z pola hráča na pole protihráča. Útok je vyhodnotený hodom všetkými kockami útočník a obrancu na príslušných poliach a porovná sa ich súčet. V prípade, že útočníkov súčet je vyšší, útok je úspešný a obranca stráca územie a jeho kocky, inak útočník stratí všetky kocky až na jednu.

# 2 Umelá inteligencia

Úlohou práce bolo vytvoriť umelú inteligenciu (AI) pre hru Dice Wars. AI je implementovaná v jazyku python.

Pozorovaním hier, analýze implementácie Dice wars a sledovaním vytvorených botoch sme sa rozhodli vytvárať trénovací dataset serializovaním konfigurácie hracej dosky pri hrách o 4 hráčoch (AI bude hrať len v hrách o 4 hráčoch) po každom ťahu hráča. Pre tento účel bolo každému hráčovi (vytvorení originálni boti) vo funkcii `ai_turn` pridaná funkcia `dicewars/shared/extract_features`, ktorá pridá serializovanú konfiguráciu hracej dosky do numpy array. Na konci hry sa pošle každému hráčovi správa o konci hry a víťazovi. Funkcia `dicewars/shared/dump_data` potom každú serializáciu upraví.

Pre vytvorenie väčšieho počtu hier je skript `create_games.py`, ktorý zavolá `run ai only game` so 4 botmi, náhodne vybraných z listu botov ponúknutých v kostre Dice Wars.

## 2.1 Serializácia hry

Konfigurácia pre každého hráča (vždy v poradí) vyberie 8 pozorovaných vlastností, počet kociek, veľkosť najväčšieho regiónu, počet regiónov, počet polí, počet polí kde je počet kociek väčší ako 4, počet polí kde je počet kociek menších ako 4, počet ľahkých cieľov a počet slabých polí na hranici.

Na konci hry sa pridalo ohodnotenie stavu z pohľadu hráča na ťahu. Ak prehral, ohodnotenie je 0. Ak vyhral, ohodnotenie je v rámci $0.5, 0.6, 0.7, 0.8, 0.9, 1$, kde značí, za koľko ťahov vyhral. Toto je kvôli nevedomosti o počte ťahov v jednej hre, čo znamená, že z pohľadu hráča, ktorý vyhral boli naozaj dôležité stavy smerujúce ku koncu hry ako stavy z počiatku hry. To sme využili v rozdeľovaní ťahov do tried podľa ich významnosti na výherný stav. Stavy z počiatku hry (prvej pätiny) hodnotíme 0.5, z nasledujúcej časti (druhej pätiny), a tak ďalej.

Výsledný vektorjednej serializovanej hry má dĺžku $32 + 1$, ohodnotenie stavu do 7 tried. Pre trénovanie sú jednotlivé ohodnotenia premenené na triedu, t.j. $0.5 = 1, 0 = 0...$

## 2.2 Rozdelenie datasetu

Skript `prepare_ds.py` vytvorí trénovací a testovací dataset. Tieto datasety rozdelí v pomere 7 ku 3, a následne ich náhodne zamieša. Na týchto datasetoch sa trénuje.

### 2.3 Trénovanie

Pre trénovanie sa používa knižnica `PyTorch`. Skript `train_model.py` vytvorí model AI, čo je lineárna neurónová sieť. Architektúra bola zvolená experimentálne, výsledný model vyzerá nasledovne:

- Vstupný vektor o dĺžke 32 je prepojený aktivačnou funkciou `ReLU` na vrstvu s 32 parametrami.
- Následná, 32 parametrová vrstva, je prepojená aktivačnou funkciou `ReLU` na vrstvu so 16 parametrami.
- Následná, 16 parametrová vrstva, je prepojená aktivačnou funkciou `ReLU` na vrstvu s 8 parametrami.
- Výstupná 6 parametrová vrstva využíva aktivačnú funkciu `softmax`.

Ako optimalizátor je zvolený algoritmus Adam s $learning\ rate = 0.001$ a `weight_decay = 0.001`, a ako stratová funkcia je `NLLLoss`, ktorá je dobrá na klasifikáciu `n` tried.

Pomocou sledovania úspešnosti nášeho modelu sme sa rozhodli pre trénovanie počas 800 epóch, po ktorých prekročení nie vždy model správne generalizoval.

## 3 Implementácia

AI sa na začiatku svôjho ťahu vždy snaží presunúť kocky do polí na hranici. Tento presun je inšpirovaný na vytvorených botoch. Následne sa vykonajú útoky, ktoré sa vracajú rekurzívnou funkciou založenou na algoritme $Max^N$. Následne sa snaží presunúť kocky k poliam, ktoré sú ohrozené. Tento presun je takisto inšpirovaný na vytvorených botoch.

### 3.1 Implementácia $Max^N$

Algoritmus $Max^N$ je rozšírený algoritmus minimax pre n hráčov. Algoritmus funguje podobne ako Mini-Max, pri prehľadávaní sa uvažuje n hráčov v poradí, vyhodnocovacia funkcia vracia hodnotu výhry pre každého hráča.
Naša implementácia $Max^N$ algoritmu je vo funkcii $maxn\_calc$. Na začiatku sa pošle poradie hráčov v poradí tak, aby predpokladala nasledujúci ťah každého hráča. Najskôr pre každého hráča spraví presun kociek k hranici. Následne sa vyberie rozvetvenie stavu. Toto rozvetvenie sa prejaví tak, že funkcia $order\_attacks$, ktorá vracia všetky útoky z listu možných útokov, ktorých pravdepodobnosť víťazstva a následného udržania je dostatočne vysoká (experimentálne 0.4 alebo keď je útok z políčka, kde počet kociek je 8), a zoradí ich podľa veľkosti. Následne funkcia $create\_ways$ vytvorí rôzne permutácie týchto útokov. Napríklad ak máme útoky [(16,12),(16,13),(20,12)], tak táto funkcia vráti tri možné cesty, [(16,12)], [(16,13), (20,12)], [(20,12)]. Potom sa rekurzívne zavolá funkcia pre ďalšieho hráča rovnakým spôsobom.
Táto funkcia vytvorí list najlepších ťahov. Po experimentovaní sme zistili, že je najlepšie spúšťať znovu túto funkciu, ako slepo veriť tejto postupnosti; pravdepodobne kvôli nedeterministickému výsledku útokov.

## 4 Postup na vytvorenie modelu

Pre serializáciu hry sa pozmenil skript $ai\_driver$, aby sa na konci hry poslala každému hráčovi správa o ukončení. Funkcia $end$ je pridaná do každého bota zo skriptu $dicewars/shared$. Tiež je upravená funkcia $ai\_turn$, ktorá v každom ťahu je zavolá funkciu $extract\_features$.

Pre vytvorenie datasetu sa spúšťa skript $create\_games$. Tento skript spúšťa nekonečno hier, o veľkosti 4 hráčov, z listu botov, ktoré sú náhodne vybrané. Tento skript vytvorí súbor $sample.csv$, ktorý obsahuje jednotlivé serializované dosky hry s ohodnotením.

Ďalší skript pre je $prepare_ds$. Tento skript rozdelí $sample.csv$ na trénovacie a testovacie dáta v pomere 7:3. Trénovacie dáta sú náhodne zamiešané.
Skript $train\_model$ natrénuje a vytvorí model a uloží ho do priečinku s našou AI.

# 5 Záver

## 5.1 Vyhodnotenie

Úspešnosť AI po natrénovaní dosahuje 50-55% výhernosť.