Protokol k semestrální práci z BI-ZUM

FIT ČVUT, LS 2019/2020

Jméno studenta: Ondřej Staníček
Username: staniond
Název semestrální práce: AI pro hru FlappyBird pomocí neuroevoluce

OSNOVA
1. Zadání semestrální práce.
Pomocí neuroevoluce vyvinout umělou inteligencni pro hru flappy bird.

2. Stručný rozbor, analýza problému/ zadání.
Jakožto základ práce jsem si vybral open source implementaci hry FlappyBird v pythonu, kterou jsem poté upravil. https://github.com/sourabhv/FlapPyBird

Hráč ovládá ptáčka a jeho úkolem je skákat mezi dírami v trubkách a vydržet co nejdéle bez naražení do země/trubky. Ovládání je jednoduché, ptáček buď skočí, nebo ne. Skore je přímo úměrné času, který hráč vydržel nenarazit.

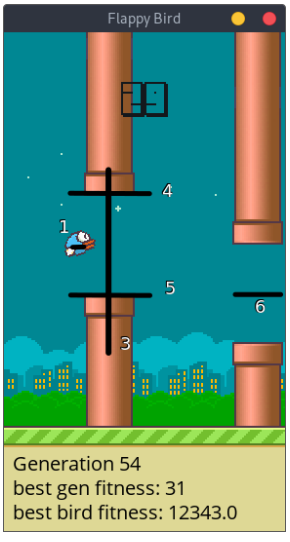
Cílem je tedy vytvořit umělou inteligenci, která se pomocí informací z herního světa v každém časovém bodě hry rozhodne, jestli má ptáček skočit, nebo ne.

3. Výběr metody.
Výběr metody je obsažen v zadání SP.

4. Popis aplikace metody na daný problém.
Jelikož řeším problém metodou neuroevoluce, rozdělil bych popis řešní na 2 části: neuronová síť a samotné trénování neutonové sítě pomocí neuroevoluce.

Neuronová síť
Ptáčka bude řídít neuronová síť, která se v každém bodě hry rozhodne, jestli má ptáček skočit, nebo ne. K tomu potřebuje informace z herního světa.Vybral jsem následujících 6 (viz obrázek):

1. y souřadnice ptáčka – výška, ve které se nachází
2. vertikální rychlost ptáčka – jak rychle padá/stoupá
3. x souřadnice nejbližší trubky – jak daleko je nejližší díra
4. y souřadnice nejbližší horní mezery
5. y souřadnice nejbližší dolní mezery – aby věděla, kam se má „vlézt“
6. y sořadnice další mezery – přidal jsem až po testování, trubky jsou velmi blízko u sebe a informace pouze o nejbližší mezeře nestačila



Neuronová síť má tedy 6 vstupů, které jsem normalizoval, aby byly v intervalu (0, 1). Dále je 8 neuronů ve skryté vrstvě a 2 výstupy. Výstupy jsou interpretovány tak, že jeden neuron má jako svou hodnotu procentuální šanci, jestli má ptáček skočit a druhý neuron šanci na „nedělání nic“. Výstupní vrstva má aktivační funkci softmax, což zajišťuje, aby výstupy byly v podobě pravděpodobnosti (jejich součet je 1). V každém časovém bodě hry se tedy do neuronové sítě pošlou dané informace a podle výstupů se rozhodne, jestli skočit, či nikoliv.

Trénování pomocí neuroevoluce
Jakožto jedinci vystupují ptáčči a jejich genotyp je vektor vah jejich neuronové sítě. Protože ptáčči neovlivňují nijak herní svět, můžu celou generaci jedinců pustit do jednoho světa a tím i vytvořit pěknou vizualizaci celé generace.

Inicializace:
Každý jedinec dostane neuronovou síť s náhodně vytvořenými váhami.

Fitness:
Fitness je přímo úměrná době, kterou jedinec přežije v herním světě. (hra má 30 snímků za sekundu, tedy při normální herní rychlosti dostane jedinec 1 bod fitness každou 1/30 sekundy). Navíc jsou jedinci penalizováni (odečet fitness) za naražení do země, nebo za naražení do trubky v místě dalekém od mezery – čím blíže mezeře, tím menší penalizace.

Mutace:
Každou váhu neuronové sítě jedince změním tak, že k ní přičtu náhodné malé číslo (může být i záporné) s předem nastavenou pravděpodobností.

Křížení:
Pokud nastane křížení, je 50% pravděpodobnost, že si daní 2 jedinci vymění všechny váhy asociované s daným neuronem. Implementoval jsem i variantu, kdy si jedinci vyměnují váhy jednotlivě, to ale nebylo tak efektivní.

Selekce:
Implementoval jsem jak turnajovou, tak ruletovou selekci, ale v praxi lépe funguje turnajová, kde vybírám 2 nejlepší jedince z 20% náhodně vybrané populace.

Hlavní logika tedy spočívá v tom, že až všichni jedinci dané generace zemřou, přidám do nové generace dosud nejlepšího jedince – celkově nejlepšího, ne pouze z dané generace (dle testů toto přináší lepší výsledky). Poté vyberu pomocí turnajové selekce 2 jedince z minulé generace, s nastavenou pravděpodobností je zkřížím, každého zmutuju a přídám do nové generace. To opakuji dokud není v nové generaci dostatek jedinců a ty poté pošlu do herního světa.

Je možnost kdykoliv ukončit proces učení a pustit do herního světa jedince s nejlepším fitness. Testováním jsem zjistil, že vhodné hodnoty pravděpodobností pro mutaci jsou 10% a pro křížení 5%. Rychlost hry je v době učení neomezená (limitována výpočetním výkonem).

5. Implementace
Projekt je psaný v jazyce Python. Pro implementaci neuronových sítí jsem si vybral framework Pytorch. Veškerý kód genetického algoritmu je psaný mnou.

6. Reference
Jako inspiraci a informace o neuronových sítích, se kterými jsem se setkal popré, jsem použil serii videí na youtube o neuroevoluci, kde se na zjednodušené verzi hry implementuje podobný algoritmus:
https://www.youtube.com/watch?v=c6y21FkaUqw