Framework pro vývoj AI enginů pro piškvorky

Obsah

1. Přehled
2. Postup práce
3. Komponenty aplikace
4. Vytvoření vlastního enginu
5. Použití frameworku
6. Tipy pro vývoj enginů
7. Příklady enginů
8. Testování a vyhodnocení
9. Osvědčené postupy
10. Struktura projektu (pro vývojáře)
11. Časté problémy a řešení
12. Získání pomoci

Přehled

Tento projekt poskytuje framework pro vývoj a testování různých AI enginů pro hru piškvorky. Zahrnuje nástroje pro vizualizaci, možnosti tréninku a funkce pro porovnávání, které vám pomohou vyvíjet a hodnotit různé AI strategie.

Postup práce

1. Začínáme:
   - Naklonujte repozitář
   - Prostudujte EngineLinear.py jako referenční implementaci
   - Vytvořte vlastní soubor enginu (např. MujEngine.py)
2. Proces vývoje:
   - Implementujte požadované metody enginu
   - Zaregistrujte svůj engine v SETTING slovníku v main.py
   - Trénujte svůj engine pomocí main.py
   - Ukládejte slibné verze do temp_engines/
3. Testování a vyhodnocení:
   - Testujte proti AI pomocí human_vs_ai.py
   - Porovnejte s jinými enginy pomocí main_komparace.py
   - Spusťte turnaje pomocí main_tournament.py
   - Iterujte a vylepšujte na základě výsledků

Komponenty aplikace

Hlavní aplikace

- `main.py` - Hlavní trénovací aplikace s vizualizací
  - Ovládání: tlačítka Play/Stop a Save
  - Konfigurace přes SETTING slovník

  - Vizualizace tréninku v reálném čase
- `main_komparace.py` - Nástroj pro porovnávání enginů
  - Porovnání dvou enginů
  - Detailní statistiky a vizualizace
  - Konfigurovatelný počet her
- `main_tournament.py` - Turnajový systém
  - Porovnání více enginů
  - Struktura každý s každým
  - Komplexní analýza výsledků
- `human_vs_ai.py` - Interaktivní testování
  - Hra proti trénovaným engineům
  - Ovládání myší
  - Vizualizace herní desky v reálném čase

Konfigurační nastavení

```
SETTING = {
    "RUN": True,         # Ovládá běh/pozastavení tréninku
    "SAVE": False,       # Při True uloží aktuální parametry enginu
    "Engine": EngineAI, # Váš engine pro trénink
    "Generations": 7,    # Počet generací pro trénink
    "PaMutation": 0.1    # Pravděpodobnost mutace
}
```

Vytvoření vlastního enginu

Základní požadavky

Každý engine musí: 1. Být třída implementující základní funkcionalitu hodnocení hry 2. Implementovat požadované metody pro práci s parametry a hodnocení desky 3. Umět ukládat/načítat své parametry pro budoucí použití

Struktura šablony

Použijte `EngineLinear.py` jako referenci. Váš engine musí implementovat tyto metody:

```
class VasEngine:
    def __init__(self):
        self.v_engine = "0.0.1"  # Sledování verze
        self.board = None
        self.initialize_parameters()

    def initialize_parameters(self):
        # Inicializace parametrů enginu
        # Příklad:
        self.parameters = {
```

```
            'param1': np.random.randn(size),
            'param2': np.zeros(size)
        }

    def mutate(self, mutation_rate=0.1, mutation_scale=0.1):
        # Implementace mutace parametrů pro trénink
        pass

    def evaluation(self):
        # Implementace hodnocení desky
        # Mělo by vracet číselnou hodnotu reprezentující stav desky
        # Kladné hodnoty favorizují hráče 1, záporné hráče -1
        pass

    def evaluate_board(self, board):
        self.board = board.copy()
        return self.evaluation()

    def get_parameters(self):
        # Vrací kopii parametrů
        return self.parameters.copy()

    def set_parameters(self, parameters):
        # Nastaví parametry ze uloženého stavu
        self.parameters = parameters.copy()

    def load_params(self, file=""):
        # Načte parametry ze souboru
        return self.set_parameters(np.load(file))
```

Reprezentace herní desky

- Herní deska je 5x5 numpy pole
- Hodnoty: 0 (prázdné), 1 (hráč 1), -1 (hráč 2)
- Vítězná podmínka: 4 v řadě (horizontálně, vertikálně nebo diagonálně)

Použití frameworku

Trénování enginů

Použijte `main.py` pro trénování vašeho enginu:

```
# V main.py nastavte třídu vašeho enginu
SETTING = {
    "Engine": VasEngine,  # Váš engine
    "Generations": 7,     # Počet generací
    "PaMutation": 0.1     # Pravděpodobnost mutace
```

```
}
```

Proces tréninku: 1. Vytvoří více instancí vašeho enginu 2. Nechá je hrát proti sobě 3. Vybírá vítěze na základě výsledků her 4. Aplikuje mutace pro vytvoření nových generací 5. Ukládá úspěšné enginy do `temp_engines/`

Testování proti jiným engineům

Použijte `main_komparace.py` pro porovnání dvou enginů:

```
comparison = EngineComparison(
    engine1_class=VasEngine,
    engine1_datafile="./temp_engines/vas_engine.npz",
    engine2_class=JinyEngine,
    engine2_datafile="./temp_engines/jiny_engine.npz",
    num_games=20,
    display_game=True
)
comparison.run_comparison()
```

Turnajový mód

Použijte `main_tournament.py` pro porovnání více enginů:

```
engine_configs = [
    (VasEngine, "./temp_engines/vas_engine1.npz"),
    (VasEngine, "./temp_engines/vas_engine2.npz"),
    (JinyEngine, "./temp_engines/jiny_engine.npz")
]

tournament = TournamentManager(
    engines_config=engine_configs,
    games_per_match=5,
    display_game=True
)
tournament.run_tournament()
```

Hra proti vašemu enginu

Použijte `human_vs_ai.py` pro interaktivní testování:

```
game = HumanVsAI(
    engine_class=VasEngine,
    engine_datafile="./temp_engines/vas_engine.npz",
    ai_starts=False  # True pro začátek AI
)
game.run_game()
```

Tipy pro vývoj enginů

Hodnotící funkce

Hodnotící funkce by měla: - Vracet vyšší hodnoty pro pozice výhodné pro hráče 1 - Vracet nižší hodnoty pro pozice výhodné pro hráče -1 - Zvažovat faktory jako: - Pozice kamenů - Potenciální vítězné linie - Blokování tahů soupeře - Kontrola desky

Správa parametrů

- Udržujte parametry ve slovníku pro snadné ukládání/načítání
- Používejte numpy pole pro efektivní výpočty
- Zvažte použití více sad parametrů pro různé fáze hry

Strategie mutace

- Vyvažte prozkoumávání vs. využívání
- Zvažte adaptivní míry mutace
- Testujte různé škály mutací

Příklady enginů

Lineární engine (poskytnutý příklad)

- Používá lineární regresi pro hodnocení desky
- Jednoduchý ale efektivní výchozí bod
- Dobrá reference pro práci s parametry

Možné přístupy

1. Konvoluční neuronové sítě
   - Rozpoznávání vzorů
   - Detekce prostorových funkcí
2. Hluboké neuronové sítě
   - Komplexní hodnocení vzorů
   - Více hodnotících kritérií
3. Heuristické přístupy
   - Ručně vytvořené funkce
   - Strategické vzory

Testování a vyhodnocení

Klíčové metriky

- Poměr výher
- Poměr remíz
- Průměrná délka hry
- Přesnost hodnocení pozic

Vizualizace

Framework poskytuje vizualizaci pro: - Průběh hry - Průběh tréninku - Výsledky turnajů - Srovnávací analýzu

Osvědčené postupy

1. Verzujte své enginy
2. Ukládejte úspěšné parametry
3. Testujte proti více soupeřům
4. Používejte vizualizační nástroje pro debugging
5. Implementujte postupná vylepšení
6. Dokumentujte strategii vašeho enginu

Struktura projektu (pro vývojáře)

Základní komponenty

- `Player.py`: Implementuje herní logiku a chování hráčů
- `GameBoard.py`: Zpracovává vizualizaci herní desky
- `Button.py`: UI komponenta pro ovládání
- `Manager.py`: Spravuje trénink a průběh hry
- `SaveHandler.py`: Zpracovává ukládání parametrů
- `StartStopHandler.py`: Řídí průběh tréninku

Vizualizační komponenty

- `colors.py`: Definice barev pro UI
- `my_dataclasses.py`: Datové struktury pro stav hry

Rozhraní enginu

Požadované metody pro vlastní enginy: - `initialize_parameters()` - `mutate()` - `evaluation()` - `evaluate_board()` - `get_parameters()` - `set_parameters()` - `load_params()`

Správa dat

- `temp_engines/`: Adresářová struktura pro uložené enginy
  - Konvence pojmenování: typEnginu_verze.npz
  - Příklad: LinEngine1.npz, ConvEngine1.npz

Průběh tréninku

1. Manager inicializuje hry
2. Hráči používají enginy pro tahy
3. Výběr vítězů na základě výsledků her
4. Mutace parametrů pro další generaci

5. Úspěšné enginy uloženy do temp_engines/

Časté problémy a řešení

1. Přetrénování na specifické soupeře

- Problém: Engine funguje dobře proti tréninkovým soupeřům, ale špatně proti ostatním
- Řešení:
  - Testujte proti různým strategiím
  - Implementujte regularizaci
  - Zvyšte různorodost tréninkové populace

2. Špatná generalizace

- Problém: Engine se nedokáže přizpůsobit novým situacím
- Řešení:
  - Zvyšte různorodost tréninku
  - Přidejte náhodnost do hodnocení
  - Vylepšete extrakci funkcí

3. Pomalé vyhodnocování

- Problém: Engine trvá příliš dlouho při rozhodování
- Řešení:
  - Optimalizujte výpočty
  - Používejte vektorizované operace
  - Zjednodušte hodnotící kritéria

4. Nestabilní trénink

- Problém: Výsledky tréninku jsou nekonzistentní
- Řešení:
  - Upravte parametry mutace
  - Implementujte včasné zastavení
  - Ukládejte úspěšné mezivýsledky

Získání pomoci

- Prostudujte vzorové implementace v EngineLinear.py
- Používejte vizualizační nástroje pro pochopení chování enginu
- Testujte proti základním engineům pro ověření vylepšení
- Analyzujte výsledky turnajů pro hodnocení výkonu
- Kontrolujte komentáře v kódu pro dodatečné detaily implementace

Pamatujte, že vývoj úspěšného enginu je iterativní proces. Začněte s jednoduchými implementacemi a postupně přidávejte složitost na základě analýzy výkonu.