СОФИЙСКИ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ
“ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА”

КУРСОВ ПРОЕКТ

по дисциплина „Интелигентни агенти с генеративен изкуствен интелект“

на тема: “Белот ИИ с Deep Q-Learning”

Рая Литова 5MI0800410

София, 2026г.

Основна функционалност

Проектът предоставя софтуерна среда за симулиране на игра на Белот и обучение на интелигентен агент. Основните функционалности включват:

1. **Симулация на игра:** Пълна имплементация на правилата на Белот (раздаване, валидни ходове, определяне на победител в трик, точкуване).
2. **Обучение (Training Mode):** Агентът се обучава чрез изиграване на хиляди игри срещу себе си, оптимизирайки своята стратегия чрез награди (rewards).
3. **Игра срещу AI (Play Mode):** Конзолен интерфейс, позволяващ на потребител да играе в отбор с AI срещу двама опоненти (AI).
4. **Запазване и зареждане:** Възможност за съхраняване на обучените модели във файлове.
5. **Unity проект:** Пълна имплементация на играта Белот, заедно с комуникация със сървър, на който се намира ИА.

Архитектура

Проектът е изграден на модулен принцип, разделящ логиката на играта от алгоритмите за обучение:

- **Logic Layer (Белот правила):** Дефинира правилата на играта Белот.
- **State Management (Управление на състоянието):** Съхранява информация за текущата ръка, изиграните карти и резултата.
- **Neural Network Layer (Невронна мрежа):** Дефинира архитектурата на AI.
- **RL Layer (Reinforcement Learning):** Обработват логиката на обучението и вземането на решения.
- **Execution Layer:** Входна точка за стартиране на обучение или игра.

Реализация

Системата използва Deep Q-Network (DQN) алгоритъм.

1. **Decision Making:** Агентът получава текущото състояние на играта, което се кодира в тензорен вид. Невронната мрежа предсказва очаквана награда за всяка възможна карта.
2. **Epsilon-Greedy Strategy:** По време на обучение AI балансира между изследване и експлоатация, за да не заседне в локални минимуми.
3. **Experience Replay:** Агентът съхранява минали игри в "памет" и периодично се обучава върху случайна извадка от тях, за да подобри стабилността на обучението.

Модел на данните

Системата поддържа два режима на точкова система: без коз и всичко коз.

Класът StateEncoder трансформира игралната ситуация във вектор с размер 106 бита:

1. **Собствена ръка (32 бита):** One-hot кодиране на наличните карти.
2. **Изиграни карти (32 бита):** Карти, които вече не са в игра.
3. **Текущо раздаване (32 бита):** Карти на масата.
4. **Обявление (6 бита):** Тип на играта (Боя, Без коз, Всичко коз).
5. **Текущ лидер (4 бита):** Кой играч печели раздаването в момента.

Използвани datasets

Проектът не използва външни статични набори от данни, защото в интернет няма готов dataset за разигравания на Белот. Той генерира собствени данни в реално време чрез:

1. **Self-play:** Агентът генерира "опити" (experiences) чрез игра срещу свои копия.
2. **Случайно раздаване:** Използва се библиотеката random за симулиране на реални раздавания от тесте с 32 карти.

Конфигурация

Learning Rate: 0.001 (Adam Optimizer).

Discount Factor (γ): 0.95 (колко се ценят бъдещите награди).

Batch Size: 32.

Memory Size: 10,000 записа.

Архитектура на мрежата: Входен слой: 106 неврона.

1. Скрит слой 1: 256 неврона (ReLU).
2. Скрит слой 2: 128 неврона (ReLU).
3. Изходен слой: 32 неврона (един за всяка възможна карта в тестето).

Използвани технологии и библиотеки

Python: Основен език на разработка.

PyTorch: Библиотека за изграждане и обучение на невронните мрежи.

NumPy: За математически операции и обработка на вектори.

Collections (deque): За управление на паметта на агента.

Проблеми и решения

Проблем: Агентът играе силни карти безсмислено, когато отборът му губи и не се стреми към запазване на покритие на високи карти.

Решение: Добавено допълнително наказание за изиграване на карти със стойност >=10 точки, ако разиграването не се печели от текущия играч или неговия партньор.

Проблем: Невронната мрежа предлага невалидни ходове.

Решение: Имплементирана е "маскировка", която задава стойност минус безкрайност на Q-стойностите за нелегални ходове, принуждавайки AI да избира само от позволените карти.

Проблем: Агентът не се стреми към печелене на последната ръка в разиграването

Решение: Допълнителна награда при печелене на последната ръка.

Проблем: Бавно учене на комплексни правила.

Решение: Въвеждане на бонус от +50/-50 точки за крайна победа/загуба в цялото разиграване, за да се насърчи дългосрочната стратегия.

Проблем: Агентът не разбира от стандартни сигнали на играча

Решение: Допълнителни награди при спазване на стандартни разигравания при сигнали и наказания в противен случай.