Informe de Proyecto: Aplicación de Técnicas Avanzadas de Aprendizaje por Refuerzo en Juegos de Múltiples Jugadores

Autor: Joaquin Vigna

Introducción

En este trabajo, exploramos la implementación y experimentación de algoritmos de aprendizaje avanzados en el contexto de juegos alternados de múltiples jugadores. Nos enfocamos específicamente en dos algoritmos clave: **CFR** (Counterfactual Regret Minimization) y **MCTS** (Monte Carlo Tree Search), aplicados a variantes de póker. Nuestro objetivo es no solo implementar estos algoritmos sino también investigar su rendimiento y efectividad en diferentes escenarios de juego.

Objetivos del Trabajo

El principal objetivo de este trabajo es profundizar en el entendimiento y aplicación práctica de los algoritmos CFR y MCTS en entornos de juego competitivos, así como explorar posibles extensiones y mejoras a estos métodos.

Entorno de Ejecución

PettingZoo es una biblioteca de juegos multiagentes que proporciona una amplia variedad de entornos diseñados específicamente para la investigación en aprendizaje por refuerzo. Este entorno será la base de nuestros experimentos, permitiéndonos simular y analizar de manera efectiva las interacciones y estrategias de los agentes en diferentes juegos.

Características Clave de PettingZoo

- **Variedad de Juegos**: Ofrece una extensa colección de juegos, incluyendo clásicos y nuevos desafíos, lo que nos permite explorar una amplia gama de escenarios y dinámicas de juego.
- **Compatibilidad con Aprendizaje por Refuerzo**: Diseñado para ser compatible con las técnicas y algoritmos de aprendizaje por refuerzo más comunes, facilitando la integración y experimentación.
- **Entorno Multiagente**: Especialmente orientado a entornos multiagentes, lo que lo hace ideal para estudiar juegos alternados de múltiples jugadores como los que abordaremos.

Implementación de los Juegos

En nuestro trabajo, la cátedra nos proporcionó una implementación de los juegos Kuhn Poker y Leduc Poker, que utilizaremos para nuestros experimentos. Estas implementaciones se basan en la biblioteca PettingZoo.

1. **Kuhn Poker**: Un juego de póker simplificado que servirá como campo de prueba para el algoritmo CFR. Experimentaremos tanto con versiones de 2 como de 3 jugadores.
2. **Leduc Poker**: Un juego de póker más complejo que será utilizado para probar y evaluar el algoritmo MCTS, así como las funciones de evaluación integradas.

Estos juegos nos proporcionarán una plataforma sólida para implementar, probar y analizar los algoritmos de aprendizaje en un contexto competitivo y controlado.

Metodología y Experimentación

Implementación de CFR

En este proyecto, hemos implementado el algoritmo Counterfactual Regret Minimization (CFR) para experimentar en juegos de póker, aunque su aplicación puede extenderse a otros contextos. Esta implementación se centra en desarrollar y refinar estrategias de juego en entornos de información imperfecta. El código completo para esta implementación se puede encontrar en el archivo `/agents/counterfactualregret.py` del proyecto.

Descripción de la Implementación

La implementación de CFR en nuestro proyecto consta de varias partes clave:

1. **Clase `Node`**: Representa un nodo en el árbol de juego. Cada nodo contiene información sobre el estado del juego (information set), el agente actual, la observación recibida y mantiene un registro del arrepentimiento acumulado y la política aprendida.
2. **Métodos de Actualización y Estrategia**: Dentro de la clase `Node`, se implementan métodos para actualizar los valores de arrepentimiento contrafactual (contrafactico) y para ajustar la estrategia actual basada en estos arrepentimientos.
3. **Clase `CounterFactualRegret`**: Extiende la clase `Agent` (basado en `PettingZoo`) y se utiliza para representar un agente que emplea CFR. Contiene métodos para realizar acciones, entrenar utilizando CFR y realizar el algoritmo de CFR de manera recursiva.
4. **Función `cfr_rec`**: Implementa la lógica del CFR de manera recursiva, calculando la utilidad de cada nodo y actualizando la estrategia y los arrepentimientos.

Naturaleza Agnóstica al Juego de las Implementaciones de Agentes

Es importante destacar un aspecto fundamental de nuestras implementaciones de agentes, incluido el agente que utiliza el algoritmo Counterfactual Regret Minimization (CFR). Estos agentes son **agnósticos al juego**, lo que significa que su diseño y funcionamiento no están limitados o definidos por las reglas o acciones específicas de un juego en particular, o por la cantidad de jugadores. En cambio, estos agentes se pueden utilizar en cualquier juego que cumpla con los requisitos básicos de la biblioteca PettingZoo, lo que los hace altamente adaptables y reutilizables.

Experimentación con Kuhn Poker

Kuhn Poker es un juego de póker simplificado que sirve como un modelo excelente para estudiar algoritmos de aprendizaje por refuerzo en juegos de información imperfecta. Este juego representa una versión reducida del póker tradicional, lo que lo convierte en un entorno ideal para experimentos y análisis en el campo de la teoría de juegos y la inteligencia artificial.

Descripción del Kuhn Poker

- **Jugadores**: El juego clásico involucra a 2 jugadores, aunque puede adaptarse para 3 jugadores.
- **Baraja**: Se utiliza una baraja reducida, típicamente con tres cartas (por ejemplo, un As, un Rey y una Reina).
- **Dinámica del Juego**: Cada jugador recibe una carta y tiene la opción de apostar (bet) o pasar (check). El juego tiene una estructura de apuestas limitada y presenta oportunidades para farolear/mentir y realizar estrategias basadas en la información limitada.

Experimentación Planeada

En nuestro proyecto, experimentaremos con Kuhn Poker de la siguiente manera:

1. **Con 2 Jugadores**: Inicialmente, nos enfocaremos en la versión clásica de Kuhn Poker con 2 jugadores. Aquí, aplicaremos y evaluaremos el algoritmo de Counterfactual Regret Minimization para desarrollar estrategias efectivas y analizar cómo los agentes aprenden y se adaptan a lo largo de múltiples juegos.
2. **Expansión a 3 Jugadores**: Posteriormente, expandiremos nuestra experimentación para incluir una versión de 3 jugadores del juego. Esta variante presenta desafíos adicionales y complejidades, permitiéndonos explorar cómo los algoritmos se adaptan a un entorno de juego más dinámico y a la presencia de un jugador adicional.

Estas experimentaciones nos permitirán obtener una comprensión más profunda de la efectividad del algoritmo CFR y cómo se puede adaptar a diferentes configuraciones de juego.

Kuhn Poker con 2 Jugadores: resultados

CFR vs Jugador Aleatorio

En la primera fase de nuestras experimentaciones, enfrentamos un agente implementando el algoritmo Counterfactual Regret Minimization (CFR) como jugador 1 contra un agente que elige acciones de manera aleatoria. Los resultados obtenidos son significativos:

Podemos observar que la estrategia aprendida por el agente CFR es mucho más efectiva que la estrategia aleatoria, ya que en promedio gana más fichas. Esto se debe a que el agente CFR aprende a jugar de manera óptima en cada estado del juego, mientras que el agente aleatorio simplemente toma decisiones aleatorias.

> Cabe destacar que en este caso el agente CFR es siempre el primer jugador, por lo que tiene una desventaja inicial.

CFR vs CFR

En una segunda fase, experimentamos enfrentando dos agentes que utilizan el algoritmo CFR. Los resultados de estos enfrentamientos apuntan a un interesante fenómeno en la teoría de juegos.

Podemos concluir que llegamos a un Equilibrio de Nash, donde el primer jugador (jugador 1) tiene una recompensa esperada de aproximadamente -1/18:

> "The game has a mixed-strategy Nash equilibrium; when both players play equilibrium strategies, the first player should expect to lose at a rate of −1/18 per hand (as the game is zero-sum, the second player should expect to win at a rate of +1/18). There is no pure-strategy equilibrium." - Wikipedia

Estos resultados son indicativos de la robustez y efectividad del algoritmo CFR en la búsqueda de estrategias óptimas en juegos de información imperfecta.

```
...     0%|          | 0/1000000 [00:00<?, ?it/s]
      100%|██████████| 1000000/1000000 [01:25<00:00, 11730.64it/s]
      -0.056521
```

Conclusión de la Experimentación en Kuhn Poker con 2 Jugadores

Tras las series de experimentaciones realizadas en el contexto del Kuhn Poker con 2 jugadores, podemos concluir satisfactoriamente que el algoritmo Counterfactual Regret Minimization alcanzó su objetivo. Los resultados indican que el agente CFR fue capaz de aprender y adaptarse eficientemente a las dinámicas del juego, superando significativamente a un oponente que toma decisiones de manera aleatoria.

La parte más relevante de estos experimentos fue la confirmación de que el agente CFR, cuando se enfrenta a un oponente que también utiliza una estrategia basada en CFR, llega a un Equilibrio de Nash. Este resultado es un hito importante, ya que demuestra la capacidad del algoritmo CFR para encontrar estrategias que son óptimas en un sentido teórico y práctico, incluso en un entorno de juego tan simplificado y abstracto como el Kuhn Poker.

Kuhn Poker con 3 Jugadores

La extensión de nuestro estudio al Kuhn Poker de tres jugadores representa un paso hacia un entorno de juego más complejo y desafiante. Este formato introduce una dinámica adicional y requiere estrategias más sofisticadas debido a la presencia de un tercer jugador. Esta variante nos permite examinar la adaptabilidad y el rendimiento del algoritmo CFR en un contexto más intrincado.

CFR vs CFR vs CFR

En esta etapa, llevamos a cabo un experimento donde tres agentes, cada uno operando con el algoritmo CFR, compitieron entre sí.

Hipótesis de Utilidad: $u_1 < u_2 < u_3$

Anticipamos una relación específica en las utilidades de los tres agentes, con el tercer jugador u_3 obteniendo la mayor utilidad, seguido por el segundo u_2 y el primero u_1. Esta hipótesis se basa en la ventaja del tercer jugador al actuar último en cada ronda, obteniendo así más información que los otros dos jugadores. Esta fase nos brinda insights valiosos sobre las dinámicas de juegos con información imperfecta y las estrategias óptimas en tales entornos.

Resultados de la Experimentación

Los resultados confirman nuestra hipótesis inicial: el tercer jugador obtiene consistentemente la mayor utilidad, seguido por el segundo y luego el primer jugador.

> "A family of Nash equilibria for 3-player Kuhn poker is known analytically, which makes it the largest game with more than two players with analytic solution. The family is parameterized using 4–6 parameters (depending on the chosen equilibrium). In all equilibria, player 1 has a fixed strategy, and he always checks as the first action; player 2's utility is constant, equal to –1/48 per hand. The discovered equilibrium profiles show an interesting feature: by adjusting a strategy parameter β (between 0 and 1), player 2 can freely shift utility between the other two players while still remaining in equilibrium; player 1's utility is equal to $-\frac{1+2\beta}{48}$ (which is always worse than player 2's utility), player 3's utility is $\frac{1+\beta}{24}$." - Wikipedia

```
...   100%|██████████| 5000000/5000000 [08:08<00:00, 10226.51it/s]
      Average rewards: {'agent_0': -0.0259104, 'agent_1': -0.0192664, 'agent_2': 0.0451768}
```

Estos hallazgos corroboran las teorías existentes y demuestran la consistencia del algoritmo CFR con las expectativas teóricas en juegos de múltiples jugadores.

Estimación de Valor de Estados en Juegos Complejos

Una de las consideraciones críticas en la aplicación de inteligencia artificial en juegos es la gestión efectiva de la profundidad en los árboles de juego. En muchos casos, los juegos poseen un espacio de estado tan extenso que hace inviable explorar todas las posibles trayectorias del juego hasta su conclusión. Esta limitación impone la necesidad de emplear estrategias de estimación de valor de estados en ciertos niveles de profundidad, en lugar de optar por una exploración completa.

Profundidad del Árbol de Juego y Explosión Combinatoria

La profundidad del árbol de juego se define por la cantidad de movimientos hacia adelante que el algoritmo analiza antes de tomar una decisión. En contextos de juegos complejos, esta profundidad puede incrementarse exponencialmente, dando lugar al fenómeno conocido como "explosión combinatoria". Esta situación resulta en un costo computacional elevado y, en muchas ocasiones, insostenible para una exploración completa y detallada.

Estrategias para la Estimación de Valor

Para superar este desafío, implementamos estrategias específicas que nos permiten evaluar los estados del juego en profundidades particulares:

1. **Funciones de Evaluación**: Estas funciones son esenciales para asignar valores numéricos a los estados del juego, basándose en sus características inherentes. Varían en complejidad, desde heurísticas basadas en reglas simples hasta modelos avanzados desarrollados mediante técnicas de aprendizaje automático.
2. **Monte Carlo Tree Search (MCTS)**: El MCTS es una técnica clave que utiliza simulaciones aleatorias partiendo del estado actual para estimar el valor de los estados. Esta estrategia es especialmente valiosa en juegos con un vasto número de posibles estados futuros, ya que proporciona una manera eficiente de evaluar opciones sin la necesidad de una exploración exhaustiva del árbol de juego.

Estas metodologías nos permiten abordar de manera efectiva los retos presentados por la complejidad y el tamaño de los juegos, mejorando significativamente la toma de decisiones en entornos de juego de alta profundidad y con gran cantidad de posibilidades.

Funciones de Evaluación

CFR Mejorado con Estimación de Valor

Hemos desarrollado una versión avanzada del algoritmo Counterfactual Regret Minimization, denominada `EnhancedCounterFactualRegret`, presente en `agentes/counterfactualregretv2.py`. Esta variante incorpora un elemento crucial: la estimación de valor en profundidades limitadas del árbol de juego.

Características Clave

1. **Estimación de Valor en Profundidad Limitada**: En vez de explorar el árbol de juego en su totalidad, `EnhancedCounterFactualRegret` emplea una función de estimación de valor (`value_estimator`) al alcanzar una profundidad máxima (`max_depth`), optimizando el manejo de juegos con grandes espacios de estado.
2. **Recursión Modificada para la Estimación de Valor**: La función `cfr_rec` se adapta para incluir la lógica de estimación de valor cuando se llega a `max_depth`, utilizando `estimate_value` para evaluar el estado actual del juego.
3. **Selección de Acciones Mejorada**: En el método `action`, se añade una validación para verificar si la observación está almacenada. Si no es así, se invoca a `action_selection` para elegir una acción. Por ultimo, se opta por una acción aleatoria, asegurando así una continuidad en la toma de decisiones.

Implementación de Función de Evaluación "Dummy" en Kuhn Poker

En el contexto de Kuhn Poker (khun2), hemos implementado una función de evaluación "dummy" para demostrar de forma práctica cómo se valoran los diferentes estados del juego. Esta función simplificada no busca optimizar el juego, sino ilustrar cómo se toman decisiones estratégicas basadas en la evaluación de los estados.

Elección de Kuhn Poker para la Demostración

Elegimos Kuhn Poker por su estructura simple y su naturaleza de información imperfecta, ideal para una demostración clara y visual de las evaluaciones en el juego.

Construcción de la Función de Evaluación

La función de evaluación "dummy" para khun2 se enfoca en:

- La carta en mano del agente.
- La última acción realizada por el oponente.

Esta aproximación permite visualizar el impacto directo de la evaluación en la estrategia del agente.

```
def evaluate_kuhn_state(game: AlternatingGame, agent: AgentID):
    its_my_turn = game.agent_selection == agent
    if game.done() or game.terminated():
        return game.rewards[agent]

    observation = game.observe(agent)
    value = 0

    hand = observation[0]

    if hand == '2':  # tengo K
        value = 1
    elif hand == '1': # tengo Q
        value = 0
    else: # tengo J
        value -= 1

    # veo la ultima accion del oponente
    if len(observation) >= 2:
        last_oponent_action = observation[-1 if its_my_turn else -2]
        if last_oponent_action == 'b':
            value -= 0.5
        elif last_oponent_action == 'p':
            value += 0.5

    return value
```

Resultados del CFR Mejorado con Estimación de Valor

Al enfrentar nuestro agente CFR mejorado, que incorpora estimación de valor, contra un agente que toma decisiones aleatorias, los resultados han sido positivos. Como era de esperar, el agente CFR con estimación de valor mostró una ventaja significativa sobre el agente aleatorio, lo que refleja la eficacia de la estrategia de aprendizaje por refuerzo en juegos de información imperfecta.

Un hallazgo sorprendente de estas experimentaciones es el aumento en el promedio de las recompensas obtenidas por el agente CFR con estimación de valor en comparación con el agente CFR standard (sin limitación de niveles) enfrentado al mismo tipo de oponente en experimentos anteriores. Esta observación sugiere que la introducción de la estimación de valor en profundidades limitadas del árbol de juego no solo mantiene la efectividad del

algoritmo CFR, sino que en ciertos contextos, puede mejorar el rendimiento del agente. Pero eso queda a criterio del docente poder discutirlo.

Monte Carlo Tree Search

Importancia de la Estimación en Leduc Poker

En juegos de información imperfecta como Leduc Poker, la estimación de valor adquiere una importancia crucial debido a la complejidad inherente de estos juegos. Leduc Poker, más complejo que Kuhn Poker, proporciona un escenario ideal para evaluar la efectividad de la técnica Monte Carlo Tree Search (MCTS).

Desafíos de Leduc Poker

Leduc Poker presenta desafíos únicos, incluyendo rondas de apuestas y una fase de revelación, lo que añade complejidad y hace impráctica la exploración exhaustiva del árbol de juego. MCTS, con su enfoque en la exploración balanceada y la explotación mediante simulaciones aleatorias, es particularmente adecuado para este entorno.

Implementación y Estrategia de MCTS en Leduc Poker

MCTS se basa en la construcción de un árbol de decisiones mediante simulaciones aleatorias, seleccionando posteriormente la mejor acción basada en estos resultados.

1. **Exploración vs. Explotación**: MCTS encuentra un equilibrio entre explorar nuevas acciones y explotar las ya efectivas. Esto es vital en juegos como Leduc Poker, con un amplio espacio de decisiones.
2. **Simulaciones Aleatorias**: MCTS maneja la incertidumbre y variabilidad en Leduc Poker, considerando una amplia gama de posibles escenarios futuros.
3. **Selección de Acciones con UCB**: Utilizando la estrategia Upper Confidence Bound para árboles, MCTS perfecciona su elección de acciones, mejorando la toma de decisiones estratégicas a lo largo del juego.

   > nota: La implementación permite utilizar otro tipo de estrategia de selección de acciones.

4. **Retropropagación**: Cada simulación aporta información valiosa para actualizar el árbol, enriqueciendo las decisiones futuras.
5. **Implementación Específica para Leduc Poker**: MCTS es implementado considerando las peculiaridades de Leduc Poker, incluyendo rondas de apuestas e información oculta.

Consideraciones Clave

- **Tiempo de Entrenamiento**: La complejidad de Leduc Poker puede requerir un tiempo considerable para simulaciones, afectando el tiempo de entrenamiento del algoritmo MCTS.
- **Eficiencia y Ajustes**: Es crucial equilibrar la eficiencia computacional y la efectividad al ajustar parámetros como el número de simulaciones y rollouts.

Implementación de MCTS para Evaluación y Acción

```
# Evaluación del juego con MCTS durante el entrenamiento
def mcts_evaluate(game: AlternatingGame, agent: AgentID):
    if game.done() or game.terminated():
        return game.rewards[agent]
    else:
        mcts = MonteCarloTreeSearch(game=game, agent=agent, simulations=100,
rollouts=10)
        return mcts.estimate()
```

```
# Selección de acción utilizando MCTS
def mcts_action(game: AlternatingGame, agent: AgentID):
    mcts = MonteCarloTreeSearch(game=game, agent=agent, simulations=3,
rollouts=1)
    return mcts.action()
```

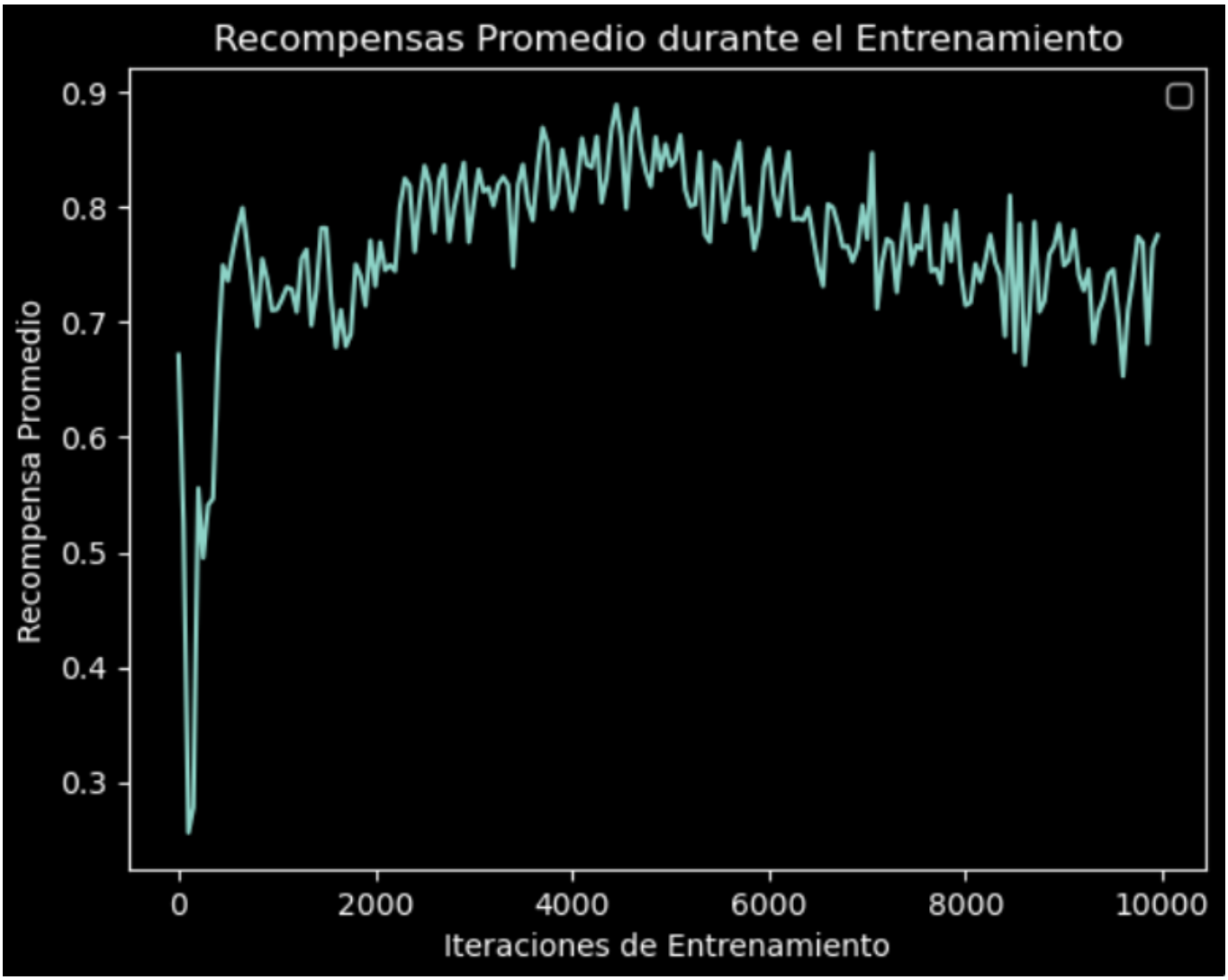
> Nota: En el entorno PettingZoo, al clonar el juego se incluyen las cartas de los oponentes. Para evitar un conocimiento imposible en la realidad, limitamos el número de simulaciones.

Resultados del Entrenamiento de CFR-MCTS en Leduc Poker

Evaluación del Desempeño del Agente

Después de implementar y entrenar nuestro agente utilizando CFR y MCTS en Leduc Poker, se realizaron varias partidas contra un agente que selecciona acciones de manera aleatoria.

Los resultados son alentadores: en promedio, nuestro agente CFR-MCTS sostiene un balance positivo frente a estrategias no optimizadas. Este resultado sirve como indicador de la efectividad de nuestra implementación de MCTS en el contexto de Leduc Poker.



La gráfica muestra la evolución de las recompensas promedio durante el entrenamiento del agente CFR-MCTS. Se observa que, después de un ascenso inicial, el agente mantiene un rendimiento consistente a lo largo de 10,000 iteraciones de entrenamiento, reafirmando la estabilidad de la estrategia aprendida frente a oponentes aleatorios.

> Nota: Se recomienda precaución al intentar replicar este entrenamiento, debido al extenso tiempo y recursos computacionales que requiere.

Experimentación: CFR pesado

Además de las metodologías y resultados ya discutidos, es importante mencionar investigaciones paralelas realizadas durante este proyecto. Una variante de CFR que incorpora pesos en los arrepentimientos —con el objetivo de acelerar la convergencia hacia el equilibrio de Nash— fue explorada.

A pesar de que esta alternativa prometía una convergencia más rápida, los experimentos no mostraron una mejora significativa. Al comparar esta versión con la implementación estándar de CFR, ambos alcanzaron un reward cercano a 1/18 (en Khun2) después de

aproximadamente 250 iteraciones. Dada la ausencia de ventajas observables, se decidió **no incluir** esta variante en la implementación final de la notebook; sin embargo, se consideró relevante mencionar este esfuerzo en el informe para dar una imagen completa del proceso exploratorio y del rigor aplicado en la búsqueda de optimizaciones.

Conclusión

Este proyecto ha explorado la aplicación de técnicas avanzadas de aprendizaje por refuerzo en juegos de múltiples jugadores con información imperfecta, centrándose en el Counterfactual Regret Minimization (CFR) y Monte Carlo Tree Search (MCTS).

Principales Logros

- **Implementación Exitosa de CFR**: La implementación de CFR y su extensión `EnhancedCounterFactualRegret` han demostrado la viabilidad de mejorar la eficiencia del algoritmo mediante la estimación de valor en profundidades específicas en el contexto de juegos como Kuhn Poker.
- **Efectividad de MCTS en Leduc Poker**: MCTS ha probado ser una herramienta efectiva en Leduc Poker, un juego con mayor complejidad estratégica, superando consistentemente a los agentes con decisiones aleatorias.

Observaciones Clave

- Las funciones de evaluación 'hardcodeadas' han mostrado un impacto significativo en la estrategia y toma de decisiones de los agentes.
- Se destacó la importancia de un equilibrio entre exploración y explotación en MCTS, un reto crucial en juegos con espacios de estado amplios y elementos de incertidumbre.
- Se identificaron retos específicos de la implementación en juegos de información imperfecta, ofreciendo insights sobre cómo superarlos.

Direcciones Futuras

Para investigaciones futuras, se identifican caminos prometedores:

1. **Contraste con Agentes Avanzados**: Contrastar el rendimiento de nuestros agentes con otros más avanzados proporcionará una comprensión más profunda del estado competitivo en juegos complejos.
2. **Refinamiento de Algoritmos**: Hay un potencial considerable para el ajuste fino de los algoritmos, buscando optimizaciones a través de la variación de parámetros y técnicas.
3. **Extensión a Nuevos Juegos**: Aplicar las técnicas desarrolladas a otros juegos de información imperfecta podría revelar más sobre la generalización y adaptabilidad de estas

estrategias.

En conclusión, hemos sentado una base sólida para la utilización de técnicas de aprendizaje por refuerzo avanzadas en juegos de estrategia de múltiples jugadores y hemos abierto la puerta a futuras exploraciones en el campo de la inteligencia artificial en juegos.

Descargo de Responsabilidad

Participación de la Inteligencia Artificial en la Composición del Texto

Este documento incluye secciones cuya composición textual ha sido asistida por una Inteligencia Artificial (IA). Es importante destacar que, aunque la IA ha contribuido en la redacción de ciertas partes del texto para mejorar la claridad y estructura del mismo, los siguientes elementos son enteramente responsabilidad del alumno:

- **Desarrollo del Código**: Todo el código presente en este trabajo ha sido desarrollado y escrito por el alumno, reflejando su comprensión y aplicación práctica de los conceptos aprendidos.
- **Conclusiones**: Las conclusiones extraídas de la experimentación y análisis son fruto del criterio y razonamiento del alumno, basadas en los resultados obtenidos y las observaciones realizadas durante el desarrollo del proyecto.
- **Decisiones Metodológicas y Conceptuales**: Todas las decisiones relacionadas con la metodología, enfoque del proyecto, y la interpretación de los conceptos teóricos son producto del trabajo independiente del alumno.

Propósito de la Asistencia de IA

El uso de la IA se ha limitado a proporcionar asistencia en la redacción para facilitar una comunicación clara y efectiva. En ningún momento, la IA ha influido en las decisiones técnicas, analíticas o conceptuales que conforman la esencia y los resultados del proyecto.