UNIVERSIDAD SIMÓN BOLÍVAR
DPTO. COMPUTACIÓN Y TEC. DE INF.
CI-5437 INTELIGENCIA ARTIFICIAL I

Informe Proyecto 3

Estudiantes
LEONEL GUERRERO
EROS CEDEÑO

Profesor
CARLOS INFANTE

20 de julio de 2023

Contenido General

1 Introducción **1**

2 Descripción del juego **1**

3 Descripción del algoritmo **2**

4 Implementación **2**

5 Ejecución del programa **3**

6 Conclusiones **4**

1. Introducción

La inteligencia artificial ha avanzado significativamente en los últimos años, y una de las áreas en las que ha demostrado ser particularmente efectiva es en la toma de decisiones en juegos. Uno de los juegos más populares en los casinos es el Blackjack, y en este proyecto se explorará cómo se puede utilizar el algoritmo de búsqueda de árbol Monte Carlo (Monte Carlo Tree Search) para crear un programa que pueda jugar al Blackjack de manera efectiva.

El Blackjack es un juego de cartas en el que el objetivo es obtener una mano con un valor lo más cercano posible a 21 sin pasarse. El crupier reparte dos cartas a cada jugador y a sí mismo, y los jugadores pueden pedir más cartas o plantarse en cualquier momento. El crupier también puede pedir más cartas hasta que su mano alcance un valor de al menos 17. El jugador que tenga una mano con un valor más cercano a 21 sin pasarse gana la partida.

En este proyecto, se utilizará el algoritmo Monte Carlo Tree Search para crear un programa que pueda jugar al Blackjack de manera efectiva. El algoritmo de búsqueda de árbol Monte Carlo es una técnica de inteligencia artificial que se utiliza para tomar decisiones en juegos complejos. El algoritmo simula múltiples juegos posibles y utiliza la información obtenida para tomar la mejor decisión posible.

2. Descripción del juego

El Blackjack es un juego de cartas que se juega comúnmente en los casinos. El objetivo del juego es obtener una mano con un valor lo más cercano posible a 21 sin pasarse. A continuación, se describen en detalle las reglas del juego de Blackjack:

1. **Valores de las cartas:** En el Blackjack, las cartas numéricas (2-10) tienen el valor que indican, las cartas con figuras (J, Q, K) valen 10 puntos cada una, y el As puede valer 1 u 11 puntos, dependiendo de lo que sea más conveniente para el jugador.
2. **Inicio del juego:** El crupier reparte dos cartas a cada jugador y a sí mismo. Una de las cartas del crupier se muestra boca arriba, mientras que la otra se mantiene boca abajo.
3. **Turno del jugador:** El jugador tiene varias opciones durante su turno:
   - **Pedir carta:** El jugador puede pedir una carta adicional para mejorar su mano. Puede seguir pidiendo cartas hasta que esté satisfecho con su mano o se pase de 21 puntos, lo que resulta en una pérdida inmediata.
   - **Plantarse:** Si el jugador está satisfecho con su mano actual, puede decidir plantarse y no recibir más cartas.
   - **Doblar:** En ciertas situaciones, el jugador puede optar por doblar su apuesta original y recibir una sola carta adicional. Esta opción generalmente se elige cuando el jugador tiene una mano prometedora.
   - **Dividir:** Si el jugador recibe dos cartas del mismo valor, puede optar por dividir su mano en dos manos separadas. Cada mano se juega de forma independiente, y se debe realizar una apuesta adicional igual a la apuesta original.
4. **Turno del crupier:** Una vez que todos los jugadores han completado sus turnos, es el turno del crupier. El crupier revela su segunda carta y sigue un conjunto de reglas predefinidas:
   - Si la mano del crupier tiene un valor de 16 o menos, debe pedir una carta adicional.
   - Si la mano del crupier tiene un valor de 17 o más, debe plantarse.
5. **Determinación del ganador:** Después de que el crupier haya completado su turno, se determina el ganador. Las siguientes situaciones pueden ocurrir:
   - Si el jugador tiene una mano con un valor más cercano a 21 que la del crupier sin pasarse de 21, el jugador gana y recibe un pago igual a su apuesta.

- Si el crupier se pasa de 21 puntos, todos los jugadores que no se hayan pasado ganan automáticamente.
- Si el jugador y el crupier tienen el mismo valor de mano, se produce un empate y se devuelve la apuesta al jugador.

6. **Blackjack:** Si un jugador recibe un As y una carta con valor 10 en sus dos primeras cartas, se considera un "Blackjackz el jugador gana automáticamente, a menos que el crupier también tenga un Blackjack. En ese caso, se produce un empate.

3. Descripción del algoritmo

El algoritmo de Monte Carlo Tree Search (MCTS) es una técnica de búsqueda de árbol utilizada en juegos y otros problemas de toma de decisiones. El algoritmo se basa en la simulación de múltiples juegos posibles y la evaluación de los resultados para determinar la mejor jugada posible.

El algoritmo MCTS consta de cuatro fases principales:

- **Selección:** En la primera fase, se selecciona un nodo del árbol de búsqueda actual. El nodo se selecciona de acuerdo con una política de selección, que puede ser determinística o estocástica. La política de selección se utiliza para equilibrar la exploración de nuevas opciones y la explotación de opciones conocidas.
- **Expansión:** En la segunda fase, se expande el nodo seleccionado agregando uno o más nodos hijos al árbol de búsqueda. Cada nodo hijo representa una posible jugada o acción.
- **Simulación:** En la tercera fase, se simula un juego completo a partir del nodo hijo recién agregado. La simulación se realiza utilizando una política de simulación, que puede ser aleatoria o determinística.
- **Retroceso:** En la cuarta y última fase, se retrocede a través del árbol de búsqueda actualizando los valores de los nodos visitados. Los valores se actualizan de acuerdo con el resultado de la simulación realizada en la fase anterior.

El algoritmo MCTS se repite hasta que se alcanza un límite de tiempo o se completa un número determinado de simulaciones. Una vez que se completa el algoritmo, se selecciona la jugada con el valor más alto en el nodo raíz del árbol de búsqueda.

En el contexto del juego de Blackjack, el algoritmo MCTS se puede utilizar para determinar la mejor jugada posible en cada turno. La política de selección se puede basar en la probabilidad de ganar la partida, mientras que la política de simulación puede ser aleatoria o basada en reglas predefinidas. La evaluación de los resultados se basa en el valor esperado de la mano del jugador después de completar el juego.

4. Implementación

La implementación del algoritmo de Monte Carlo Tree Search (MCTS) se realizo en el lenguaje de programación Python. El código fuente del proyecto se encuentra disponible en el repositorio de GitHub. La implementación del juego de Blackjack se basa en la creación de objetos que representan las cartas, el estado del juego y la ejecución del algoritmo de búsqueda de árbol.

En primer lugar, se define una clase `Card` que representa una carta en el juego de Blackjack. La clase `Card` tiene atributos como `value` (valor de la carta), `suit` (palo de la carta)

y `numeric_value` (valor numérico de la carta). Además, se implementan métodos para representar la carta como una cadena, comparar cartas, realizar operaciones matemáticas y obtener los valores y palos de la carta.

A continuación, se define una clase `State` que representa el estado del juego de Blackjack. La clase `State` tiene atributos como `dealer_cards` (cartas del crupier), `remaining_cards` (cartas restantes en el mazo), `hand` (cartas en la mano del jugador), `parent` (nodo padre en el árbol), `simulation_number` (número de simulaciones realizadas desde el estado) y `simulation_value` (valor acumulado de las simulaciones realizadas desde el estado). La clase `State` implementa varios métodos para manipular el estado del juego, como `take_card()` (tomar una carta del mazo), `not_terminal()` (verificar si el estado es terminal), `ucb()` (calcular el valor UCB para la selección del nodo), `get_best_move()` (obtener la mejor jugada posible), `is_fully_expanded()` (verificar si el estado está completamente expandido), `create_child()` (crear un nuevo estado hijo), `simulate()` (simular el juego desde el estado), y otros métodos auxiliares.

Finalmente, se define una clase `MCTS` que representa el algoritmo de Monte Carlo Tree Search para el juego de Blackjack. La clase `MCTS` tiene un constructor que recibe el nodo raíz del árbol y el número de iteraciones a realizar en la simulación. La clase `MCTS` implementa varios métodos para ejecutar el algoritmo `MCTS`, como `run()` (ejecuta la simulación), `select()` (selecciona un nodo para expandir), `expand()` (expande el nodo seleccionado), `simulate()` (simula el juego), y `backpropagate()` (actualiza los valores de los nodos visitados). También se incluye un método `get_best_action()` que devuelve la mejor jugada posible a partir del nodo raíz del árbol.

La implementación del algoritmo de Monte Carlo Tree Search en el juego de Blackjack se basa en la creación de objetos que representan las cartas y el estado del juego, y la utilización de la clase MCTS para ejecutar el algoritmo de búsqueda de árbol. Esta implementación permite simular múltiples juegos posibles y determinar la mejor jugada posible en cada turno del juego de Blackjack. Cabe destacar que esta implementación puede ser adaptada a otros juegos de cartas y problemas de toma de decisiones similares.

5. Ejecución del programa

Para ejecutar el programa del simulador de Blackjack, se debe utilizar el archivo `blackjack_simulator.py` y proporcionar un argumento opcional que indique el número de iteraciones a realizar en la simulación. Si no se proporciona un argumento, se utilizará un valor predeterminado de 100 iteraciones.

El programa se ejecuta en la línea de comandos y muestra el estado actual del juego después de cada iteración. El usuario puede elegir tomar una carta, plantarse o salir del juego en cualquier momento. El programa utiliza el algoritmo de Monte Carlo Tree Search para determinar la mejor jugada posible en cada turno.

Para ejecutar el programa, se debe abrir una terminal y navegar hasta la carpeta que contiene el archivo `blackjack_simulator.py`. Luego, se debe ejecutar el siguiente comando:

```
python blackjack_simulator.py [iteration_number]
```

Donde `[iteration_number]` es un número entero que indica el número de iteraciones a realizar en la simulación (opcional).

El programa imprimirá un mensaje de bienvenida y el número de iteraciones a realizar. A continuación, se mostrará el estado inicial del juego y se ejecutará el ciclo principal del juego. Después de cada iteración, se mostrará el estado actual del juego y se pedirá al usuario que tome una decisión. Si el usuario decide salir del juego o si el juego termina, se mostrará un mensaje de fin de juego y el programa se cerrará.

6. Conclusiones

En este proyecto se implementó el algoritmo de Monte Carlo Tree Search (MCTS) en el juego de Blackjack para determinar la mejor jugada posible en cada turno. Se utilizó Python como lenguaje de programación y se crearon clases para representar las cartas, el estado del juego y el algoritmo MCTS.

Se realizaron una pequeña cantidad de pruebas de manera manual con diferentes valores de iteraciones y se observó que el algoritmo MCTS fue capaz de determinar la mejor jugada posible en la mayoría de los casos. Sin embargo, en algunos casos, el algoritmo no pudo determinar la mejor jugada posible debido a la complejidad del juego y la aleatoriedad de las cartas.

El algoritmo de Monte Carlo Tree Search es una técnica efectiva para determinar la mejor jugada posible en el juego de Blackjack. Sin embargo, se requiere de un número suficiente de iteraciones para obtener resultados precisos y la aleatoriedad del juego puede afectar la precisión del algoritmo.

En general, este proyecto permitió adquirir conocimientos sobre el algoritmo de Monte Carlo Tree Search, su implementación en Python y su aplicación en el juego de Blackjack. Además, se aprendió sobre la representación de objetos en Python y la manipulación de estructuras de datos complejas.