Automatizando el juego del triqui
Reporte del primer problema del curso
"IA: Representación y solución de problemas"

Edgar Andrade

31 de agosto de 2020

1. Introducción

La automatización de juegos cuya resolución requiere algún grado de inteligencia fue una tarea muy relevante durante las primeras etapas de la inteligencia articial. Los juegos constituyen un muy buen conejillo de indias, toda vez que definen un entorno completamente regimentado y relativamente sencillo, fácilmente susceptible a la formalización. El juego del triqui (o tres-en-línea) es un ejemplar muy favorable a este respecto, sobre todo para la representación formal, y más bien sencilla, de un *task environment* [1, §3.1], así como para ilustrar la metodología de búsqueda de soluciones en un espacio de estados.

El juego del triqui involucra dos jugadores, quienes hacen sus jugadas por turnos en un tablero de 3×3. El primer jugador arranca poniendo X en alguna casilla, luego el segundo pone una O en alguna de las casillas vacías, luego vuelve el turno para el jugador 1 y así en adelante. El juego termina cuando no hay más casillas vacías o cuando alguno de los dos jugadores logra poner tres de sus figuras en línea (ver Figura 1).

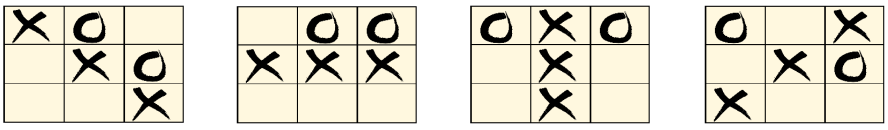

Figura 1: Cuatro maneras de ganar el juego del triqui con las X. La victoria con las O es similar.

Nuestro problema consiste en implementar un agente que sea competente para jugar al triqui. Este problema admite una solución completa, toda vez que el agente que implementamos nunca podrá ser derrotado.

2. Métodos

La solución al problema requiere una definición formal previa a la implementación en python, que es la siguiente:

Estado inicial: Situación del entorno desde el cual comienza el juego. En el caso del triqui, el estado inicial es el tablero vacío.

Jugador(s): Define cuál jugador tiene el turno en el estado s, el cual puede ser O o X.

Posibles acciones(s): Descripción de las posibles acciones del $Jugador(s)$, dado un estado s. En este caso, poner o bien una O o una X en una casilla vacía.

Función de transiciones(s, a): Descripción del entorno que resulta de la ejecución de la acción a por el $Jugador(s)$ en el estado s. Junto con el estado inicial y las posibles acciones, la función de transiciones define el espacio de estados del juego.

Prueba de objetivo(s): Permite determinar si el juego se termina cuando se obtiene el estado s.

Función de utilidad(s): Definida sólo cuando el juego se termina en el estado s y especifica la utilidad asociada al estado s. En el caso del triqui, asumiremos que si el ganador son las X, la utilidad es 1; si el ganador son las O, la utilidad es -1; en caso de empate, la utilidad es 0.

Una vez implementado el problema así definido, se procedió a implementar el algoritmo `minmax` mediante la función `minmax-decision` (ver Figura 2; Russell and Norvig 1, §5.2). Este algoritmo crea un árbol de estados, partiendo desde un estado dado s, atribuyéndole un valor `minmax` a cada estado con base en la función de utilidad arriba descrita. Los pagos para el jugador con las X son positivos, y los del jugador 2 con las O son negativos, así que el primero buscará estados que maximicen la utilidad, mientras que el segundo buscará estados que la minimicen (ver Figura 3). Estas funciones definen el back-end de la aplicación.

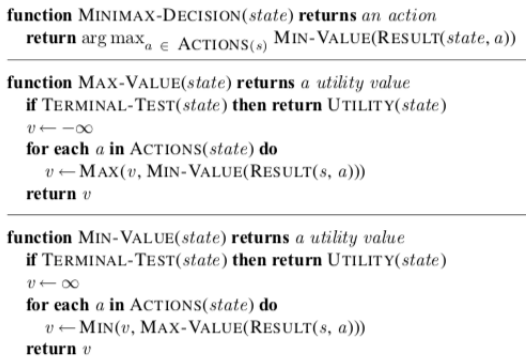

Figura 2: Pseudo-código de las funciones que implementan la toma de decisiones del algoritmo minmax.

Como front-end, se impementó una página web usando la librería Dash de python, y fue desplegada en un servidor EC2 en AWS.

3. Resultados

El programa de agente se implementó para jugar con las O, de tal manera que cuando el jugador humano, con las X, hace una jugada, el computador corre el algoritmo `minmax` sobre el tablero resultante. Al comienzo del juego existen muchas opciónes, por lo que el computador debe generar un árbol relativamente grande y se debe esperar algún tiempo para obtener la respuesta de las Os. No obstante, las siguientes jugadas se obtienen muchísimo más rápido.

Dado que el agente genera el árbol de estados completo, previendo todos los caminos posibles, sus acciones son óptimas. Si el humano no juega bien, el computador sabrá encontrar cuál jugada lleva a la victoria. El mejor jugador humano encontrará la jugada óptima, la cual ya ha sido prevista por el computador; en este caso el humano no podrá obtener más que un empate contra el computador.

4. Discusión

El algoritmo que implementa el programa de agente es muy poderoso, permitiendo que el agente nunca pierda y logre ganar sin piedad ante un jugador

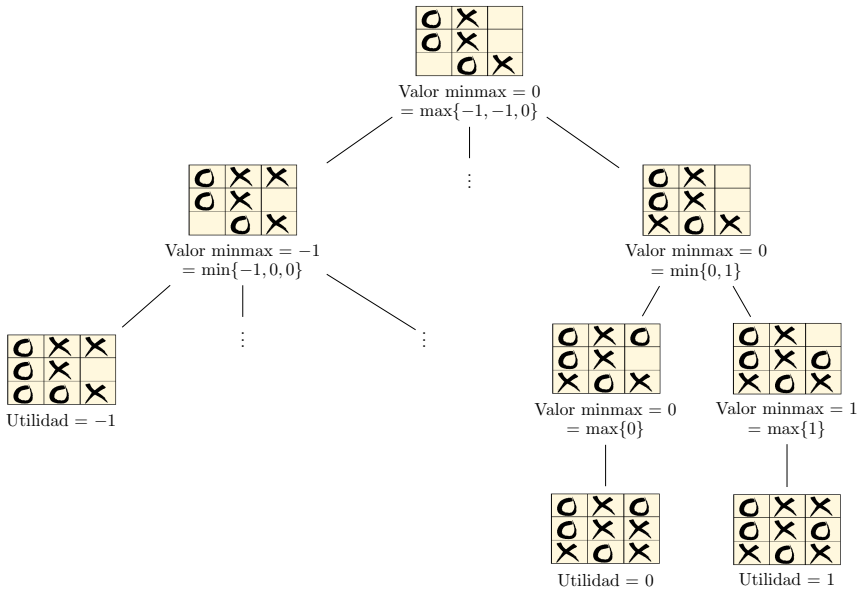

Figura 3: Árbol de decisión minmax. El jugador de las X busca maximizar el valor de las opciones a su disposición. El jugador de las O busca minimizar dicho valor. Por ejemplo, el nodo superior corresponde a un estado en donde juegan las X, quien debe responder con la opción de la derecha, la cual maximiza los valores `minmax`.

humano ingenuo. No obstante, la creación del árbol completo de decisión es un método muy ineficiente para obtener la primera jugada por parte de las O, por lo que se buscará la creación de una tabla de decisión que pueda guardarse en un archivo, para ser cargada en memoria al comenzar el juego. También falta incorporar la posibilidad de que el computador juege con las X.

5. Conclusiones

Se implementó de forma satisfactoria un programa de agente para el juego del triqui que es imbatible. A pesar de algunos procesos ineficientes, un jugador humano puede disfrutar de una partida de triqui contra un adversario temible.

Referencias

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson Education Limited, 2016.