Cálculo de *Estrategias Ganadoras* mediante Simulaciones

Adrían Arellano
Jonathan Ocampo

Motivación

Dentro de la variedad de juegos existentes, en particular, podemos interesarnos en juegos por turnos de dos jugadores sobre un tablero rellenable. Estos destacan por su baja cantidad de turnos y, en general, una combinación de jugadas posibles muy grande.
En estos juegos, una manera de afrontar el problema de *hallar una estrategia ganadora*, es: encontrar los estados del tablero para los cuales algún jugador gana o termina (hojas), evaluar quién gana, y asignarles una utilidad. En este caso, una estrategia ganadora requiere evaluar todos los estados hojas, sin embargo, este cálculo se vuelve muy costoso por la gran cantidad de estados posibles.
Si abandonamos la idea de evaluar todas las hojas, nos podemos concentrar en obtener una estrategia, no perfecta, pero buena, tan solo evaluando algunas hojas. Para elegir cuales hojas evaluar, nos afirmaremos de un modelo de aprendizaje reforzado.

1. Introducción

Podemos modelar un juego de dos jugadores en un tablero usando la tupla (V, D, U), donde V son los posibles estados del tablero, D es un subconjunto de V^2 que caracteriza las reglas del juego y $U : V \to \{-1, 0, 1\}$ es una función de utilidad tal que $U(v)$ indica que tan beneficiosa es para un jugador cambiar el estado de juego a v producto de la elección que hizo en su turno. Se puede mostrar que U esta caracterizada por la utilidad de los estados hojas de la siguiente manera:

$$U(v) = \begin{cases} \mathbb{1}_{\{v \text{ es un estado ganador}\}}(v) \\ \text{si } v \text{ es hoja del digrafo } (V, D) \\ \\ -\max\{k \mid \exists\ (v, u) \in D, U(u) = k\} \\ \text{en otro caso.} \end{cases}$$

Una vez calculado U, se puede mostrar que contiene toda la información necesaria para construir una estrategia ganadora.
La principal dificultad de un algoritmo que calcule U es encontrar las hojas de (V, D), dado que en general dicho digrafo suele ser muy grande y no se tiene a D como un conjunto de arcos explícitos. Este problema suele complicarse mucho más si el tamaño de V es extremadamente grande computacionalmente.
Para resolver el problema, buscaremos alguna estrategia buena caracterizada por una política π^*, la cual puede ser aproximada por políticas π_n obtenidas por iteraciones. Cada iteración es construida usando una aproximación de la función de utilidad u_n obtenida del conocimiento almacenado de n iteraciones. Luego, dicha función de utilidad es usada para calcular las siguientes ecuaciones deducidas de Bellman Óptimo aplicadas en GPI.

$$u_n(v) = \begin{cases} U(v) \\ \text{si } v \text{ es una hoja conocida} \\ \\ 0 \\ \text{si } v \text{ no es conocido} \\ \\ \sum_{(v,u)} u_n(u)\pi_n(u|v) \\ \text{en otro caso} \end{cases}$$

$$\pi_{n+1}(v) = \Delta \arg\max_{(v,u)\in D} u_n(u),$$

donde $u_n(v)$ nos entrega la utilidad esperada dada de usar la política π_n. Se puede probar que dicha estrategias son mejores a medida que cada iteración pasa.
Finalmente, cada conocimiento se irá guardando al simular partidas usando una política π_n. Se puede probar que si estas partidas se parecen, se puede encontrar una estrategia óptima sin haber visitado todos los nodos.

2. Presentación

En la presentación se revisará

- Definiciones.
- Ejemplos de juegos del tipo a estudiar.
- Ejemplos de un juegos calculables y no calculables por los algoritmos.
- Explicación de un modelo de aprendizaje reforzado.
- Explicación del algoritmo que construye GPI.
- Ejemplos de estrategias obtenidas por aprendizaje reforzado.

3. Bibliografía

- Apuntes del curso.
- Schrittwieser, J., Antonoglou, I., Hubert, T. et al. Mastering Atari, Go, chess and shogi by planning with a learned model. Nature 588, 604–609 (2020)..
- Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.