Makarena: Deducción evolutiva de funciones de evaluación aplicadas a *Minimax*

Propuesta de tesis

Ignacio Slater M.
Departamento de Ciencias de la Computación
Universidad de Chile

Nancy Hitschfeld
Departamento de Ciencias de la Computación
Universidad de Chile
Profesora guía

Santiago, Chile
27 de abril de 2022
`Versión 0.1.2204231628`

1. Introducción

El concepto de *máquinas que aprenden* fue propuesto por primera vez por *Alan Turing* en 1950 con una simple pregunta: «¿Pueden las máquinas pensar?» [2]. Siguiendo esa pregunta, no es descabellado preguntarse: «¿Puede un computador jugar como una persona?»
El estudio de *teoría de juegos* se le atribuye a *von Neumann* en su paper de 1928 [1] y desde entonces se ha desarrollado como un área de investigación. Entre los avances de la teoría de juegos se encuentra el desarrollo de algoritmos capaces de idear estrategias similares a las de los jugadores humanos, llamaremos a estos «*jugadores artificiales*».
En este documento nos centraremos en un algoritmo en particular, *Minimax*, debido a que ha sido ampliamente estudiado y utilizado [1, 3, 8, 11, 14]. Sin embargo, este algoritmo depende de una *función de evaluación* que otorga un valor numérico a cada estado del juego para discriminar qué jugada es «mejor» que otra, pero esta función se desprende de estudiar en detalle el juego [4] o por «tanteo» dado que la función de evaluación es **específica del dominio**[17]. Este trabajo propone una manera automática de encontrar la función de evaluación del estado del juego sin importar el dominio de éste.

2. Planteamiento del problema

Esta sección introducirá varios conceptos necesarios para plantear el problema, y para esto se dividirán las definiciones en dos partes: *Teoría de juegos* y *Algoritmos evolutivos*.

2.1. Teoría de juegos

2.1.1. Función de evaluación

Definición 2.1 (**Recurso**)**.** Se le dirá recurso a cada elemento en el juego que acerque a un jugador a una jugada óptima.

Se le dirá **función de evaluación**[13] $v_i(s)$ a la función que determina la cantidad de recursos que tiene el jugador i en el estado s. A su vez, el **estado del juego** se define como el par a_i, a_{-i} que representa las acciones de los jugadores que llevan a una configuración específica del juego (un ejemplo de configuración puede ser la posición y cantidad de piezas en el tablero).

2.1.2. Juegos de suma cero

Los ***juegos de suma cero*** [15] (*zero sum games*) son una representación matemática usada en *teoría de juegos* y *teoría económica* para describir una situación donde existen dos «jugadores» que se enfrentan entre sí. Un *juego es de suma cero* si la ventaja que lleva un jugador es igual a la pérdida del otro.
Los *juegos de suma cero* son un tipo de juegos de suma constante donde la suma de ganancia y pérdida es igual a cero.
En un juego de suma cero, todos los jugadores perciben la misma ganancia de cada recurso. Para entender esto podemos considerar un juego con tres estados: empate, ganador y perdedor, con puntajes 0, 1 y -1 respectivamente.

- **Empate:** Ningún jugador tiene ventaja; ambos jugadores tienen 0 puntos.
- **Ganador:** El jugador ganó, por lo que tiene 1 punto; el otro jugador pierde así que tiene -1 puntos.
- **Perdedor:** El jugador perdió, por lo que tiene -1 puntos; el otro jugador ganó así que tiene 1 punto.

De esta forma, para todo estado del juego, la suma de los puntajes es 0.

2.1.3. *Minimax*

Minimax[3] (*MM*) es un algoritmo de decisión utilizado en múltiples ámbitos para minimizar la pérdida posible para el «peor caso» (*maximum loss*). Se formuló originalmente para juegos de suma cero de n jugadores, pero se ha extendido a juegos más complejos y a problemas de decisión con incertidumbre.

Definición 2.2 (**Valor *Maximin***)**.** El valor *maximin* de un estado del juego es la máxima cantidad de recursos que puede obtener un jugador, sin conocer las jugadas de su oponente.
Formalmente:

$$\underline{v_i} = \max_{a_i} \min_{a_{-i}} v_i(a_i, a_{-i})$$

Donde:

- i es el índice del jugador actual.
- $-i$ representa a todos los jugadores oponentes.
- a_i es la jugada del jugador i.
- a_{-i} son las jugadas de todos los oponentes.
- v_i es la función de evaluación del estado del juego en el turno del jugador i.

Definición 2.3 (**Valor *Minimax***)**.** El valor *minimax* de un estado del juego es la mínima cantidad de recursos que un oponente puede forzar al jugador a perder, sin saber las jugadas del jugador.
Formalmente:

$$\overline{v_i} = \min_{a_{-i}} \max_{a_i} v_i(a_i, a_{-i})$$

Definición 2.4. Se define la función MM para un estado del juego s como:

$$MM_i(s) = \begin{cases} v_i(s) & \text{si } s = F \\ \underline{v_i}(s) & \text{si es el turno del jugador } i \\ \overline{v_i}(s) & \text{si es el turno del oponente } -i \end{cases}$$

Con F representando el estado final del juego.

Definición 2.5 (**Algoritmo *Minimax***)**.** Sea `t` un árbol de estados del juego, donde cada nodo del árbol es un estado del juego evaluado con la función de evaluación v_i, y donde cada arco del árbol es una jugada de un jugador (para simplificar diremos que cada jugada representa un turno y que los jugadores toman turnos alternados, primero i y luego $-i$). Luego, cada nodo tendrá tantos hijos como jugadas pueda realizar cada jugador en su turno.
Se define el algoritmo *minimax* para un árbol de estados `t` como:

```
fun minimax(t: StateTree, player: Player in [i, -i]): Double =
  if (t.isTerminal()) {
    t.value
  } else if (player == -i)  {
    t.children.maxOf { child -> minimax(child, i) }
  } else {
    t.children.minOf { child -> minimax(child, -i) }
  }
```

Para entender mejor el algoritmo, considere el árbol de estados de la fig. 1. Cada nodo del árbol representa un estado del juego, y cada arco representa una jugada de un jugador. Con esto podemos definir una función de evaluación arbitraria donde el valor de cada nodo es la suma de puntajes acumulados en el camino que conecta la raíz del árbol con el nodo. Aplicando esta función de evaluación al algoritmo *minimax* se toman las decisiones indicadas por las flechas rojas en la figura, tomando el mínimo o máximo dependiendo de quién sea la jugada. Finalmente se escoge la secuencia de jugadas marcadas en verde.

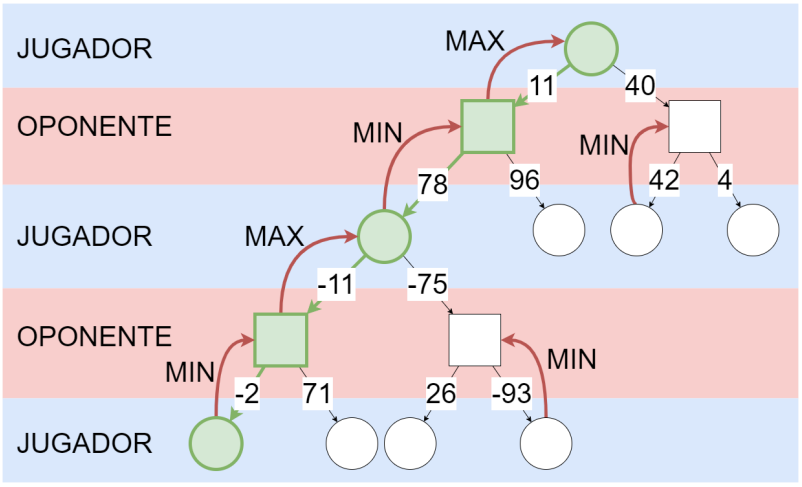

Figura 1: Árbol de estados de un juego.

2.2. Algoritmos evolutivos

2.2.1. Algoritmos genéticos

Los *algoritmos genéticos*[10, 9] son una clase de algoritmos evolutivos que utilizan una *población de individuos* para resolver un problema. En este contexto, los individuos son una secuencia de valores que representan una **solución al problema**. A cada individuo se le asigna una **aptitud** (*fitness*) que es un valor numérico que representa la calidad de la solución, de esta forma podremos discriminar qué soluciones son más apta para resolver el problema. Específicamente, los algoritmos genéticos pueden definirse de la siguiente forma:

1. Se crea una población inicial de soluciones aleatorias.
2. Se seleccionan pares de individuos siguiendo alguna *estrategia de selección*.
3. Se cruzan los pares de individuos siguiendo algún *operador de cruza* para obtener nuevos individuos (hijos).
4. Se mutan los nuevos individuos siguiendo algún *operador de mutación* y una *tasa de mutación*.[1]
5. Se evalúa la solución más apta de la nueva población respecto a un criterio de aceptación (e.g., margen de error). Si se cumple el criterio, se termina el algoritmo, si no, se vuelve al paso 2.

2.2.2. Programación genética

La *programación genética* (GP)[7] es un tipo de algoritmo genético donde el espacio de solución son **programas**. Para lograr esto se define a cada individuo como una representación abstracta de un programa, esta representación podría ser un **árbol de sintaxis abstracta**, una **pila de ejecución**, entre otras.
Un caso de uso típico de la *programación genética* es hacer una *regresión simbólica*[6] a partir de un conjunto de valores, donde el objetivo final es encontrar una función que se aproxime bien a los valores de entrada.

3. Estado del arte

3.1. *Artificial Ant Problem*

El **problema de las hormigas artificiales** [5] es un problema que busca encontrar un programa que, al ser ejecutado para una hormiga le permita encontrar toda la comida en un entorno dado, con una cantidad limitada de movimientos. En [5], *Koza* propone una solución al problema utilizando programación genética para encontrar un programa que maneje a la hormiga. Para esto, se definen las primitivas [16]:

[1] Probabilidad (usualmente baja) de que un individuo mute.

- `if (foodAhead) { child1() } else { child2() }`
- `moveForward()`
- `turnLeft()`
- `turnRight()`

De esta forma, los programas que se generan mediante GP podrá moverse en 3 direcciones y verificar si tiene comida en frente.
La *función de evaluación* simplemente contará la cantidad de comida acumulada por la hormiga luego de una cierta cantidad de movimientos. De esta forma, el *fitness* del programa será directamente proporcional a dicha cantidad de comida. Una implementación completa se puede encontrar en [12].

4. Pregunta de investigación

Dada una representación del estado de un juego. ¿Es posible definir un algoritmo que encuentre una función de evaluación del estado del juego para la toma de decisiones del algoritmo *Minimax*?

5. Hipótesis

Es posible utilizar *programación genética* para derivar la función de evaluación como un árbol de sintaxis abstracta.

6. Objetivos

- First itemtext
- Second itemtext
- Last itemtext
- First itemtext
- Second itemtext

7. Metodologías

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit.

Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

8. Resultados esperados

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Referencias

[1] J. v. Neumann. «Zur Theorie der Gesellschaftsspiele». En: *Mathematische Annalen* 100.1 (dic. de 1928), págs. 295-320. ISSN: 0025-5831, 1432-1807. DOI: 10.1007/BF01448847. URL: http://link.springer.com/10.1007/BF01448847 (visitado 24-04-2022).

[2] A. M. Turing. «I.—COMPUTING MACHINERY AND INTELLIGENCE». En: *Mind* LIX.236 (1 de oct. de 1950), págs. 433-460. ISSN: 1460-2113, 0026-4423. DOI: 10.1093/mind/LIX.236.433. URL: https://academic.oup.com/mind/article/LIX/236/433/986238 (visitado 20-04-2022).

[3] Ky Fan. «Minimax Theorems». En: *Proceedings of the National Academy of Sciences of the United States of America* 39.1 (ene. de 1953), págs. 42-47. ISSN: 0027-8424. pmid: 16589233. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1063722/ (visitado 18-04-2022).

[4] Claude E. Shannon. «Programming a Computer for Playing Chess». En: *Computer Chess Compendium*. Ed. por David Levy. New York, NY: Springer New York, 1988, págs. 2-13. ISBN: 978-1-4757-1970-3 978-1-4757-1968-0. DOI: 10.1007/978-1-4757-1968-0_1. URL: http://link.springer.com/10.1007/978-1-4757-1968-0_1 (visitado 26-04-2022).

[5] John R. Koza. «Genetic Programming as a Means for Programming Computers by Natural Selection». En: *Statistics and Computing* 4.2 (1 de jun. de 1994), págs. 87-112. ISSN: 1573-1375. DOI: 10.1007/BF00175355. URL: https://doi.org/10.1007/BF00175355 (visitado 22-04-2022).

[6] Riccardo Poli y col. *A Field Guide to Genetic Programming*. [Morrisville, NC: Lulu Press], 2008. 233 págs. ISBN: 978-1-4092-0073-4.

[7] William B. Langdon y Riccardo Poli. *Foundations of Genetic Programming*. Springer Science & Business Media, 9 de mar. de 2013. 265 págs. ISBN: 978-3-662-04726-2. Google Books: zsaqCAAAQBAJ.

[8] Michael Maschler, Eilon Solan y Shmuel Zamir. *Game Theory*. Cambridge: Cambridge University Press, 2013. ISBN: 978-0-511-79421-6. DOI: 10.1017/CBO9780511794216. URL: http://ebooks.cambridge.org/ref/id/CBO9780511794216 (visitado 18-04-2022).

[9] Milad Taleby Ahvanooey y col. «A Survey of Genetic Programming and Its Applications». En: *KSII Transactions on Internet and Information Systems (TIIS)* 13.4 (2019), págs. 1765-1794. ISSN: 1976-7277. DOI: 10.3837/tiis.2019.04.002. URL: https://www.koreascience.or.kr/article/JAKO201919761177651.page (visitado 20-04-2022).

[10] John H Holland. *Adaptation in Natural and Artificial Systems - An Introductory Analysis with Applications to Biology, Control, and Artificial I*. 2019. ISBN: 978-0-262-27555-2. URL: http://www.vlebooks.com/vleweb/product/openreader?id=none&isbn=9780262275552 (visitado 26-04-2022).

[11] Kiran K Thekumparampil y col. «Efficient Algorithms for Smooth Minimax Optimization». En: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/hash/05d0abb9a864ae4981e933685b8b915c-Abstract.html (visitado 20-04-2022).

[12] *Deap/Ant.Py at B8513fc16fa05b2fe6b740488114a7f0c5a1dd06 · DEAP/Deap*. Col. de Félix-Antoine Fortin y Ben Elliston. Distributed Evolutionary Algorithms in Python, 27 de abr. de 2022. URL: https://github.com/DEAP/deap/blob/b8513fc16fa05b2fe6b740488114a7f0c5a1dd06/examples/gp/ant.py (visitado 27-04-2022).

[13] *Evaluation Function*. En: *Wikipedia*. 27 de mar. de 2022. URL: https://en.wikipedia.org/w/index.php?title=Evaluation_function&oldid=1079533564 (visitado 18-04-2022).

[14] *Minimax*. En: *Wikipedia*. 12 de mar. de 2022. URL: https://en.wikipedia.org/w/index.php?title=Minimax&oldid=1076761456 (visitado 18-04-2022).

[15] *Zero-Sum Game*. En: *Wikipedia*. 5 de abr. de 2022. URL: https://en.wikipedia.org/w/index.php?title=Zero-sum_game&oldid=1081052640 (visitado 18-04-2022).

[16] DEAP Project. *Artificial Ant Problem — DEAP 1.3.1 Documentation*. URL: https://deap.readthedocs.io/en/master/examples/gp_ant.html (visitado 27-04-2022).

[17] Charles R. Dyer. *CS 540 Lecture Notes: Game Playing*. URL: https://pages.cs.wisc.edu/~dyer/cs540/notes/games.html (visitado 26-04-2022).