Aplicación de aprendizaje por refuerzo mediante Alpha Zero en el juego de Ultimate Tic-Tac-Toe

David Moisés Aguilar Paredes, *20141933, Ingeniería Mecatrónica,* Ivonne Rocío Heredia León, *20151795, Ingeniería Industrial,* Eduardo Andre Cuya Nizama, *20151192, Ingeniería Industrial,* y Carlos Alberto Sosa Pezo, *20161310, Ingeniería Informática.*

***Resumen*—Ultimate tic-tac-toe es una variante del juego de tres en raya que, a diferencia del juego base, no resulta trivial para algoritmos de decisión como minimax o poda alfa-beta debido a la cantidad de estados posibles. En este artículo, se propone la aplicación del algoritmo de aprendizaje por refuerzos AlphaZero en dicho juego, comparando su desempeño frente a un árbol de búsqueda de Monte Carlo (MCTS) y deep q-learning. Se espera que AlphaZero alcance un nivel de habilidad semejante o mayor a dichos algoritmos.**

I. Introducción

EL presente proyecto busca analizar la performance de tres agentes que utilizan distintos métodos de inteligencia artificial por medio del juego Ultimate Tic Tac Toe, el cual consiste en jugar una partida de Tres en raya donde cada casilla es a su vez un tablero de Tres en raya. Este juego ha sido analizado por diversos investigadores [1], usuarios de GitHub [2] e incluso existen papers en IEEE [3]; Sin embargo, no se ha realizado una comparación de desempeño entre los algoritmos Alpha Zero, Monte Carlo y deep q-learning. Debido a que existe una constante búsqueda por incrementar el desempeño de estos algoritmos con el objetivo de mejorarlos es que es importante realizar este análisis. Aun con las restricciones de tiempo de entrenamiento y recursos computacionales bajo los que se realiza este experimento, en comparación con la implementación original de AlphaZero por parte de DeepMind [4], se espera que el modelo mejore su rendimiento en el juego de Ultimate tic-tac-toe con los sucesivos entrenamientos y pueda presentar una habilidad semejante o superior a las de otros algoritmos con los que se va a enfrentar. Esto se medirá mediante el número de victorias obtenidas frente a estos algoritmos en diferentes estados de aprendizaje, a los que se espera que venza en por lo menos un 50% de las veces en sus etapas finales.

II. Metodología

A. Técnicas base (baselines) para comparación

El uso de técnicas adversariales en Ultimate Tic Tac Toe (UTTT) son numerosas ([1],[5], [6]). La implementación de MiniMax de Amar y Binyamin [5], utiliza profundidad limitada para acelerar la respuesta del algoritmo; asimismo, se utilizaron tres funciones diferentes para evaluar la conveniencia de los estados posibles, una suerte de heurística, la primera de ellas propone la visualización del movimiento anterior en el tablero pequeño para decidir sobre la mejor opción, la cual es dada por una función de pesos creada por los autores; la segunda propone asignar un valor constante muy alto a los estados terminales donde gana Max y un valor negativo en donde gana Min, en caso exista un empate se asignará un valor de cero para ese estado y en caso no sea un estado terminal asigna un valor dado por la suma de los pesos por las celdas ganadas menos la suma de los pesos de las celdas por las celdas del enemigo, por último, la tercera es muy parecida a la segunda pero utiliza una función para asignar los valores más compleja. Chen, Doan y Xu [1] utilizan técnicas como minimax implementado con alfabeta, Monte Carlo Tree Search (MCTS), deep q-learning y un modelo híbrido entre minimax y MCTS. la función de evaluación de minimax utilizada es la cantidad de mini tableros ganados por Max menos los mini tableros perdidos. Como baselines, se utilizarán los algoritmos Monte Carlo Tree Search. y el modelo de deep q-learning implementado por Chen,Doan y Xu [1]. Debido a que Minimax y alfabeta tienen una complejidad temporal muy grande.

B. Enfoques propuestos

Ultimate Tic-Tac-Toe tiene una cuadrícula de 3x3 de juegos regulares de tic-tac-toe. Llamaremos al tablero grande como tablero general y cada pequeño tablero como mini tablero.

El primer jugador comienza colocando una pieza donde desee, luego el siguiente jugador debe colocar su pieza en el mini tablero de acuerdo con el movimiento del oponente. Por ejemplo, si el primer jugador juega en la casilla superior derecha del mini tablero inferior izquierda, entonces su oponente tiene que jugar en el mini tablero superior derecho. Si el primer jugador juega en el recuadro superior izquierdo del mini-tablero inferior izquierdo, entonces su oponente tendría que jugar en el mini-tablero superior izquierdo. En la figura 1, podemos observar una explicación gráfica de las reglas del juego.

Quien gane 3 mini tableros colocados consecutivamente, en una fila, columna o diagonal, gana el juego Ultimate Tic-Tac-Toe. Si un mini tablero ya se ganó o está lleno, el jugador enviado puede elegir cualquier otro mosaico para colocarse.

Para la implementación del Ultimate Tic Tac Toe ha realizar, los estados estarán definidos mediante un tablero de 9x9, en el cual cada casilla representa una marca (0 para casilla vacía, 1 para el jugador 1 y -1 para el jugador 2).

La entrada se representará como un arreglo de 9x9 y la acción realizada. El output será el tablero actualizado y la lista de acciones posibles a realizarse para el siguiente movimiento.

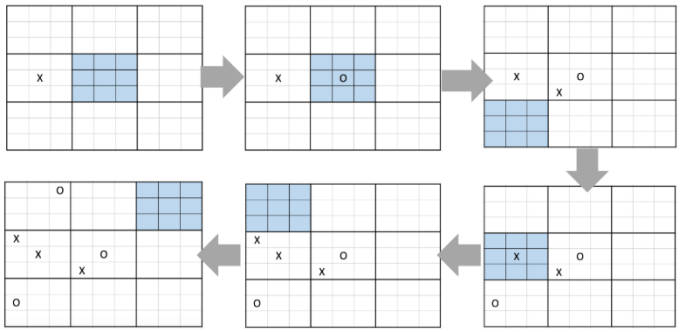

Fig. 1. Explicación gráfica Ultimate Tic Tac Toe

La condición de término del juego será cuando uno de los jugadores logre completar 3 mini tablero consecutivos o no haya movimientos posibles restantes en la lista de movimientos, es decir el juego termine en empate.

Para solucionar el problema se utilizará los algoritmos de AlphaZero y deep q-learning.

III. Experimentación y Resultados Preliminares

A. Setup experimental

Para la creación del entorno de juego se utilizó el lenguaje de programación de Python. Por otro lado, para el desarrollo de la red neuronal, se utilizó el marco de trabajo Keras. Como base para la implementación de AlphaZero, se tomó la elaborada por el estudiante de doctorado en Ciencia de la Computación en la Universidad de Stanford Surag Nair [7].

La generación de datos del modelo se da a partir de las simulaciones del juego. Lo único que se tuvo que especificar fueron las reglas del juego, a partir de ahí, se ejecutan iteraciones en las que se simulan partidas mediante MCTS para generar la probabilidad de tomar una acción dado un estado. Producto de la simulación de los juegos, se obtienen los estados de los tableros, los cuales servirán de input para nuestra red neuronal, la cual busca generar un modelo de predicción tanto de la política como del resultado esperado de la partida. Finalmente, se enfrenta, para cada iteración, al modelo generado con el último mejor modelo en 10 enfrentamientos. Si la razón de duelos ganados por el nuevo modelo sobre total de encuentros con un ganador supera el valor de 0,6; se acepta el nuevo modelo como mejor.

Las métricas de evaluación que se utilizaron fueron los tiempos de ejecución tanto del entrenamiento como de la etapa de prueba, así como la fracción de juegos ganados entre los juegos totales hasta cada juego.

En esta entrega, se mostrará el rendimiento del agente de aprendizaje por refuerzos contra un agente aleatorio y el aprendizaje de AlphaZero; para ello, se entrenará ambos por separado.

El modelo de aprendizaje por refuerzos fue probado de manera local en computadores con sistema operativo Windows 10, 12 GB RAM y procesador Intel Core i5. Por otro lado, el modelo de AlphaZero fue probado de manera local en un computador con sistema operativo macOS Catalina 10.15.5, 16 GB de RAM y procesador Intel Core i7.

Asimismo, se realizaron los siguientes experimentos: AlphaZero vs Random player y AlphaZero vs Deep q-learning. Se probaron dos configuraciones para la red neuronal de Alpha Zero, las cuales se mostrarán a continuación.

	Test 1
NumIters	8
NumEps	100
TempThreshold	15
UpdateThreshold	0.6
MaxLenOfQueue	20000
NumMCTSSims	25
ArenaCompare	10
CPUCT	1
NumItersForTrain ExamplesHistory	20
Learning Rate	0.001
Dropout	0.3
Épocas	10
Batch Size	64
Num Channels	512

Fig. 2. Hiperparámetros de dos experimentos del algoritmo AlphaZero.

B. Resultados alcanzados

Para el agente que emplea deep q-learning se utilizaron 1000 partidas en 40 iteraciones para el entrenamiento y 100 partidas para el testeo del agente; asimismo,la red neuronal cuenta con 4 capas densas, tres con función de activación ReLu y en la otra capa se usa una función lineal. La implementación base se obtuvo de Banerjee [8] y se modificó la red neuronal para igualarla en parámetros respecto al modelo de AlphaZero. En el caso del AlphaZero, se utilizaron 11 épocas, con una duración total de 14 horas, 16 minutos y 24 segundos, para entrenar la red neuronal; asimismo, la red neuronal cuenta con 3 capas convolutivas y 2 capas densas, ambas con función de activación ReLu. En la figura 2, se observa la frecuencia de victorias, pérdidas y empates del agente de aprendizaje por refuerzos contra un agente aleatorio como parte de un entrenamiento que duró 8 horas, 34 minutos y 20 segundos.

En el caso de AlphaZero, la red neuronal busca predecir dos variables - la política y la del resultado esperado - para, así, mejorar su optimalidad respecto a modelos pasados.

En la figura 4, se muestra la frecuencia de juegos ganados por el agente contra el mejor modelo anterior.

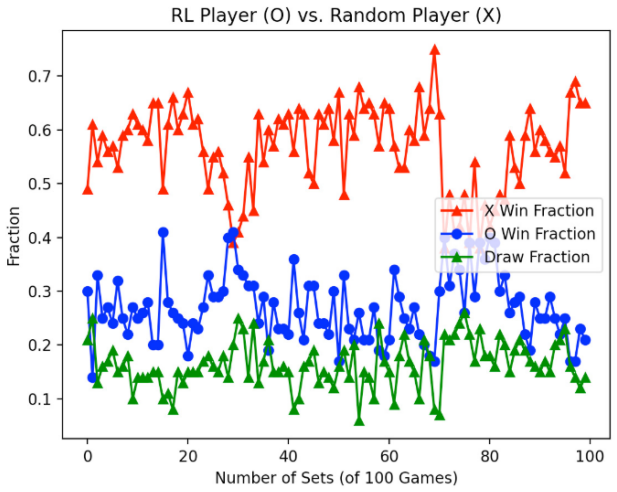

Fig. 3. Frecuencia de victorias agente de aprendizaje por refuerzos vs agente aleatorio.

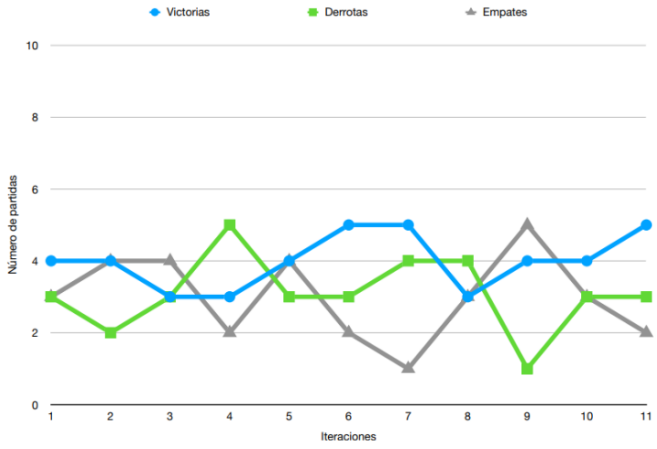

Fig. 4. Resultados contra el anterior mejor modelo

Puede verse de la gráfica que, si bien el número de victorias y empates ha sido inestable durante las iteraciones, las derrotas han mantenido una número menor respecto a las victorias en la mayoría de los casos, lo que coincide con que una mayor cantidad de ejemplos de entrenamiento da a la red una mayor probabilidad de aprender a jugar mejor que un modelo anterior.

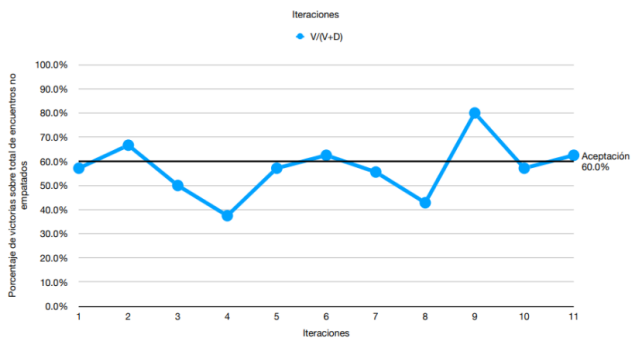

Fig. 5. Victorias/ (Victorias + Derrotas) para cada modelo generado.

Sin embargo, en la razón de victoria sobre la totalidad de encuentros excluyendo los empatados se puede notar un patrón errático y un pico en la iteración 9 correspondiente a las 5 victorias obtenidas comparándolas con 1 derrota en esa iteración.

IV. Conclusiones

Como se ha podido evidenciar mediante los experimentos realizados en este proyecto, el algoritmo de AlphaZero tiene un mejor desempeño frente a un agente aleatorio, un agente basado en deep q-learning y a otro basado en MCTS. Se puede observar que promedio del ratio de victorias de AlphaZero contra el agente aleatorio es de 63.72%; asimismo, se puede notar un incremento en el ratio de victorias a medida que la iteración del algoritmo es mayor; lo mismo sucede en el experimento AlphaZero vs Deep q Learning, con la diferencia en que la media es 71.25%. Esto no sucede en AlphaZero vs MCTS, en este experimento se puede notar que el ratio de victorias es mayor que el 50% en el 40% de las veces en las que se enfrentaron ambos algoritmos, la media en este experimento fue de 51.25%. Algunas de las posibles causas de este rendimiento son la cantidad de iteraciones que se utilizaron para entrenar al agente no fueron suficientes y la red que se utilizó fue simplificada.

Ratio de Victorias		RandomPlayer	DeepQLearning	MCTS
Alpha Zero	Checkpoint 11	65.00%	80.00%	65.00%
	Checkpoint 9	80.00%	75.00%	65.00%
	Checkpoint 6	55.00%	65.00%	30.00%
	Checkpoint 2	55.00%	60.00%	45.00%

Fig. 6. Ratio de victorias de Alpha Zero contra los modelos aleatorio, monte carlo tree search y Deep Q-learning.

Ratio de Victorias (Sin Empate)		RandomPlayer	DeepQLearning	MCTS
Alpha Zero	Checkpoint 11	86.67%	94.12%	57.89%
	Checkpoint 9	94.12%	93.75%	65.00%
	Checkpoint 6	78.57%	92.86%	37.50%
	Checkpoint 2	78.57%	85.71%	52.94%

Fig. 7. Ratio de victorias sin considerar empates de Alpha Zero contra los modelos aleatorio, árbol de búsqueda de Monte Carlo y deep q-learning.

Se puede observar una notoria mejora en el indicador de victorias que excluye los empates en el caso de AlphaZero vs el agente aleatorio, ya que se obtuvo una media de 84.48%; asimismo, el incremento entre iteraciones es mayor que en el ratio de victorias; así como en el experimento AlphaZero vs Deep q Learning en el cual se obtuvo una media del indicador de 91.61%; sin embargo, la mejoría no es muy notoria para el MCTS, 53.33%; por lo que se podría descartar a los empates como posible sesgo.

V. Sugerencias para trabajos futuros

En primer lugar, se puede mejorar la implementación de AlphaZero pues, en este proyecto se utilizó una versión simplificada. Esto es posible debido a que la propuesta original de DeepMind es de una arquitectura residual de 20 capas.

En segundo lugar, los enfoques desarrollados en este trabajo pueden ser aplicados a juegos determinísticos similares como

las variaciones del tic tac toe (tic-tac-toe 3d, tic-tac-toe,etc) y otros como ajedrez, shogi, etc.

VI. Repositorios de trabajo

- Aprendizaje por refuerzo (Bifurcación de shayakbanerjee/ultimate-ttt-rl) https://github.com/Cubi123/ultimate-ttt-rl
- AlphaZero (Bifurcación de suragnair/alpha-zero-general) https://github.com/csosapezo/alpha-zero-ultimate-tic-tac-toe

VII. Contribuciones de cada integrante

- Pruebas del agente aprendizaje por refuerzos: Eduardo Cuya, Ivonne Heredia
- Implementación del entorno de juego usado por AlphaZero: David Aguilar, Carlos Sosa

Bibliografía

[1] P. and Jesse Doan and Edward Xu. (2018). Ai agents for ultimate tic-tac-toe, [Online]. Available: https://web.stanford.edu/~jdoan21/cs221paper.pdf.

[2] Shayakbanerjee. (2017). Reinforcement learning based ultimate tic tac toe player, [Online]. Available: https://github.com/shayakbanerjee/ultimate-ttt-rl.

[3] Sneha Garg, Dalpat Songara, and Saurabh Maheshwari. (2017). The winning strategy of tic tac toe game model by using theoretical computer science, [Online]. Available: https://ieeexplore.ieee.org/document/8003944.

[4] DeepMind. (2017). Mastering chess and shogi by self-play with ageneral reinforcement learning algorithm, [Online]. Available: https://arxiv.org/pdf/1712.01815.pdf.

[5] E. Adi Ben Binyamin. (2017). Ai agent for ultimate tic tac toe game, [Online]. Available: https://www.cse.huji.ac.il/~ai/projects/2013/U2T3P/files/AI_Report.pdf.

[6] Subrahmanya Sista. (2016). Adversarial game playing using monte carlo tree search, [Online]. Available: https://etd.ohiolink.edu/!etd.send_file?accession=ucin1479820656701076&disposition=inline.

[7] Shayak Banerjee. (2017). Using aprendizaje por refuerzos to play ultimate tic-tac-toe, [Online]. Available: https://medium.com/@shayak_89588/playing-ultimate-tic-tac-toe-with-reinforcement-learning-7bea5b9d7252.

[8] Surag Nair. (2017). A simple alpha(go) zero tutorial, [Online]. Available: https://web.stanford.edu/~surag/posts/alphazero.html.