



OpenTruco: Deep RL en el Truco Argentino

Entorno de OpenSpiel para el aprendizaje de estrategias con información imperfecta

Marcos Piotto
mpiotto@udesa.edu.ar

Segundo Santos Torrado
ssantostorrado@udesa.edu.ar

Lucas Vitali
lvitali@udesa.edu.ar

Resumen

El Truco Argentino es un juego de naipes tradicional sudamericano con información imperfecta, donde las apuestas *-betting-* y el engaño *-bluffing-* son componentes esenciales de su estrategia.

En este trabajo, presentamos OpenTruco, un entorno de simulación desarrollado en el *framework* OpenSpiel de DeepMind, que permite el entrenamiento y evaluación de agentes de aprendizaje por refuerzo profundo en el Truco Argentino.

Asimismo, evaluamos el desempeño de algoritmos de *reinforcement learning* y enfoques basados en *regret minimization*, analizando su capacidad para aprender estrategias óptimas en un contexto adversarial bajo incertidumbre.

Palabras clave: Truco Argentino, OpenSpiel.

1. Objetivos

El objetivo principal de este proyecto es el desarrollo y la publicación de OpenTruco¹, un entorno de aprendizaje por refuerzo para el juego de naipes Truco Argentino. Este *environment* fue diseñado para ser compatible con el *framework* OpenSpiel [1], la librería de referencia desarrollada por DeepMind para la investigación en *reinforcement learning* y algoritmos de búsqueda y planificación en juegos.

A través de esta implementación, se busca aportar un modelo computacional eficiente y estandarizado de un juego de suma cero e información imperfecta. Si bien el Truco comparte dinámicas centrales con el Póker —como las apuestas o *betting*, y la mecánica del engaño o *bluffing*—, posee particularidades estratégicas y una relevancia cultural que lo convierten en un dominio de prueba desafiante para el desarrollo de agentes inteligentes.

Adicionalmente, este trabajo tiene como fin validar la robustez del entorno propuesto mediante el desarrollo de agentes basados en aprendizaje reforzado profundo, así como *regret minimization*,

comparando sus desempeños con estrategias *baseline* aleatorias y basadas en búsqueda. De este modo, se pretende demostrar la utilidad de OpenTruco como plataforma para la experimentación en *reinforcement learning* en juegos adversariales con información imperfecta.

2. Introducción

En esta sección se presentan los fundamentos teóricos y metodológicos que sustentan el desarrollo de OpenTruco. Primero, se describen las reglas de la variante de Truco Argentino seleccionada para la implementación. Posteriormente, se ofrece una caracterización formal del juego desde la perspectiva de la Teoría de Juegos Algorítmica, justificando la elección del *framework* OpenSpiel. Finalmente, se realiza un análisis exhaustivo de los paradigmas algorítmicos aplicables al desarrollo de agentes inteligentes en juegos de información imperfecta, presentando los métodos disponibles en OpenSpiel que resultan pertinentes para este dominio.

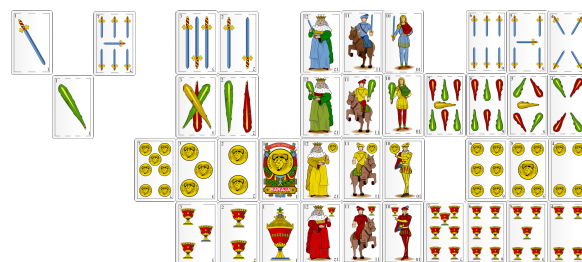


Figura 1: Jerarquía de valores de las cartas para el Truco. Cada naipes pierde contra los de su izquierda, empatando con los del mismo nivel y gana contra los de su derecha.

2.1. Reglas del Truco Argentino

Las reglas seleccionadas para el desarrollo de OpenTruco se basan en el reglamento oficial publicado por la Asociación Argentina de Truco [2]. En particular, se implementó la lógica correspondiente a la modalidad de Truco 1 vs. 1 sin Flor a 30 puntos. Esto define una partida entre dos jugadores individuales, excluyendo la mecánica de azar

¹Repositorio de OpenTruco: <https://github.com/segusantos/open-truco>



de la "Flor", con una puntuación objetivo de 30 puntos para ganar el juego. La partida se divide en dos etapas de puntuación: las "malas" (primeros 15 puntos) y las "buenas" (puntos restantes).

El juego utiliza una baraja española de 40 naipes (sin ochos, nueves ni comodines). La partida se estructura en "manos", donde cada jugador recibe tres cartas. El objetivo de cada mano es ganar al menos dos de las tres rondas, o la primera en caso de empates sucesivos.

La dinámica estratégica se rige por dos flujos de apuestas independientes:

1. **Truco:** Apuesta sobre el resultado de las rondas. Es escalable (Truco, Retruco, Vale Cuatro) y permite aumentar el valor de la mano en juego. Puede realizarse en cualquier momento del turno del jugador.
2. **Envío:** Apuesta sobre el valor numérico de la combinación de cartas de un mismo palo. Se resuelve antes de finalizar la primera mano y añade una capa de decisión temprana basada en su evaluación.

2.2. Teoría de Juegos Algorítmica

Desde una perspectiva formal, el Truco Argentino se modela como un juego extensivo de dos jugadores, de suma cero e información imperfecta [3]. Esta clasificación determina las propiedades fundamentales del dominio y orienta la selección de algoritmos apropiados.

- **Juego extensivo:** El Truco se desarrolla en forma secuencial, donde los jugadores alternan turnos y las decisiones se toman en función del historial de acciones observables. Esta estructura se representa naturalmente mediante un árbol de juego.
- **Suma cero:** La estructura de recompensas es estrictamente competitiva; la utilidad obtenida por un agente (u_i) es exactamente la inversa de la de su oponente (u_{-i}), tal que $\sum u = 0$. En este contexto, el concepto de solución adecuado es el Equilibrio de Nash [4], donde ningún jugador puede mejorar unilateralmente su utilidad esperada.
- **Información imperfecta:** A diferencia de juegos como el Ajedrez o el Go, los jugadores no poseen conocimiento completo del estado global: las cartas del oponente permanecen ocultas. Esto obliga a los agentes a operar sobre conjuntos de información (*information sets*), agrupando estados del juego que son indistinguibles desde la perspectiva del jugador.
- **Estocasticidad y engaño:** El reparto aleatorio de cartas introduce incertidumbre en el nodo raíz de cada mano. Además, las mecánicas

de apuesta habilitan el *bluffing* (engaño), donde un agente puede realizar acciones agresivas con una mano débil para inducir al oponente a retirarse.

Esta caracterización motiva la elección de OpenSpiel [1] como *framework* de desarrollo. OpenSpiel provee las abstracciones necesarias para representar juegos extensivos con información imperfecta, incluyendo soporte nativo para conjuntos de información, nodos de azar, y una amplia biblioteca de algoritmos de aprendizaje y búsqueda específicamente diseñados para estos dominios.

2.3. Paradigmas Algorítmicos para Juegos de Información Imperfecta

El desarrollo de agentes inteligentes en juegos de información imperfecta puede ser abarcado desde paradigmas algorítmicos específicos que difieren fundamentalmente de los métodos diseñados para información perfecta. A continuación, se presenta una taxonomía de los enfoques aplicables al Truco, clasificados según su fundamento teórico.

2.3.1. Métodos de Búsqueda

IS-MCTS. *Information Set Monte Carlo Tree Search* [12] adapta el algoritmo MCTS clásico para juegos de información imperfecta. A diferencia de MCTS estándar, IS-MCTS opera sobre conjuntos de información en lugar de estados concretos, y utiliza técnicas de *determinization* (muestreo de estados consistentes con la observación actual) para guiar la búsqueda. OpenSpiel incluye una implementación de IS-MCTS que resulta útil como *baseline* para evaluar agentes entrenados.

2.3.2. Minimización de Arrepentimiento Contrafactual

Los algoritmos basados en *Counterfactual Regret Minimization* (CFR) constituyen el paradigma dominante para aproximar equilibrios de Nash en juegos extensivos de información imperfecta. A diferencia del aprendizaje por refuerzo clásico, estos métodos minimizan el arrepentimiento contrafactual: la diferencia entre la utilidad obtenida y la que se habría obtenido al tomar acciones alternativas.

CFR. *Counterfactual Regret Minimization* [5] fue el primer algoritmo en demostrar convergencia al equilibrio de Nash en juegos extensivos de información imperfecta. El método mantiene contadores de arrepentimiento acumulado para cada acción en cada conjunto de información, y actualiza la estrategia proporcionalmente a los arrepentimientos positivos. CFR requiere recorrer el árbol de juego completo en cada iteración, lo cual resulta prohibitivo para juegos de gran escala. OpenSpiel incluye una implementación tabular de CFR.



MCCFR. *Monte Carlo CFR* [6] extiende CFR mediante técnicas de muestreo que evitan la enumeración exhaustiva del árbol. Las variantes principales incluyen *external sampling* (muestrea las acciones del oponente y los nodos de azar) y *outcome sampling* (muestrea una trayectoria completa del juego). Ambas variantes preservan las garantías de convergencia de CFR con alta probabilidad, pero reducen drásticamente el costo computacional por iteración. OpenSpiel proporciona implementaciones verificadas de ambas variantes.

Deep CFR. *Deep Counterfactual Regret Minimization* [7] combina el marco teórico de CFR con redes neuronales para aproximar las estrategias y los arrepentimientos acumulados. Esto elimina la necesidad de almacenar tablas explícitas de arrepentimiento, permitiendo escalar a juegos con espacios de estados intratables para métodos tabulares. Deep CFR entrena redes de ventaja (*advantage networks*) que predicen los arrepentimientos contrafactuales para cada conjunto de información. El algoritmo fue validado en variantes de Póker a gran escala.

2.3.3. Aprendizaje por Refuerzo con Self-Play

Los métodos de *Deep Reinforcement Learning* pueden adaptarse a juegos de información imperfecta mediante esquemas de *self-play*, donde el agente entrena compitiendo contra versiones anteriores de su propia política [9]. Si bien estos métodos no garantizan convergencia a equilibrios de Nash y pueden exhibir dinámicas cíclicas, resultan efectivos para explorar políticas diversas.

DQN. *Deep Q-Networks* [10] aproxima la función de valor-acción $Q(s, a)$ mediante una red neuronal profunda. El algoritmo emplea *experience replay* y *target networks* para estabilizar el entrenamiento. En juegos de información imperfecta, DQN opera sobre observaciones parciales del estado, aprendiendo a tomar decisiones bajo incertidumbre.

PPO. *Proximal Policy Optimization* [11] es un algoritmo de gradiente de política que optimiza una función objetivo sustituta con restricciones sobre la magnitud de las actualizaciones. En juegos de suma cero, PPO se combina con *self-play* para generar oponentes adaptativos.

2.3.4. Métodos Híbridos: Aprendizaje por Refuerzo y Teoría de Juegos

Una línea de investigación busca combinar las fortalezas del aprendizaje por refuerzo profundo con las garantías de convergencia de la teoría de juegos clásica.

NFSP. *Neural Fictitious Self-Play* [8] integra el aprendizaje por refuerzo con el concepto de *Fictitious Play*. El algoritmo entrena simultáneamente dos redes neuronales: una red DQN que aprende una *mejor respuesta aproximada* contra la estrategia histórica promedio del oponente, y una red de aprendizaje supervisado que modela la *política promedio* del propio agente a lo largo del entrenamiento. La mezcla de ambas redes, ponderada por un parámetro de anticipación η , define la política de comportamiento. NFSP demostró convergencia a estrategias aproximadamente Nash en Leduc Poker y rendimiento competitivo en Limit Texas Hold'em.

Todos los algoritmos descritos se encuentran implementados en OpenSpiel, lo cual facilita la experimentación comparativa. Para la validación inicial de OpenTruco, seleccionamos NFSP por su balance entre fundamentos teóricos y eficiencia computacional.

3. Desarrollo

En esta sección se describe la implementación de OpenTruco como un entorno compatible con el *framework* OpenSpiel. Se presenta el modelado formal del juego, incluyendo la representación del estado, el espacio de acciones y la estructura de recompensas. Asimismo, se discuten aspectos técnicos relevantes como el muestreo consistente desde conjuntos de información y el uso de *reward shaping* basado en potencial para acelerar el entrenamiento de agentes.

OpenSpiel modela los juegos como máquinas de estados finitos, donde cada estado encapsula toda la información necesaria para determinar las transiciones legales y las recompensas. Para juegos de información imperfecta, el *framework* distingue entre el *estado del mundo* (completo) y la *observación o estado de información* que percibe cada jugador. Esta distinción resulta fundamental para algoritmos como CFR y MCCFR, que operan sobre conjuntos de información, así como para métodos de muestreo como IS-MCTS, que requieren generar estados consistentes con las observaciones del agente.

3.1. Modelado del Entorno

El entorno OpenTruco implementa la interfaz *Game* y *State* de OpenSpiel, exponiendo las funcionalidades necesarias para el entrenamiento y evaluación de agentes. A continuación, se detalla la representación del estado, el espacio de acciones y la estructura de recompensas.

3.1.1. Espacio de Estados

El estado del juego se representa mediante un vector de características que codifica tanto la información privada del jugador (su mano) como la información pública observable (cartas jugadas,



apuestas, puntuación). La representación tensorial se estructura de la siguiente manera:

- **Jugador observador** ($n_p = 2$ bits): codificación *one-hot* del jugador cuya perspectiva se representa.
- **Mano privada** ($n_c = 40$ bits): vector binario indicando las cartas que el jugador posee y no ha jugado.
- **Historial de rondas** ($3 \times 2 \times 40 = 240$ bits): para cada una de las tres rondas, se codifican las cartas jugadas por cada jugador mediante vectores *one-hot*.
- **Análisis de rondas** ($3 \times 3 + 3 \times 2 = 15$ bits): ganadores de cada ronda (incluyendo empate) y jugador líder en cada una.
- **Puntuación del juego** (2 valores normalizados): puntos acumulados de cada jugador, normalizados por el objetivo (30 puntos).
- **Nivel de Truco** (4 bits): codificación *one-hot* del nivel actual de apuesta (1 a 4 puntos, esto es, sin apuesta, Truco, Retruco y Vale Cuatro).
- **Secuencia de Envído** ($4 \times 3 = 12$ bits): hasta cuatro llamadas de Envído codificadas como *one-hot* sobre los tres tipos posibles (Envído, Real Envído, Falta Envído).
- **Estado del Envído** (2 bits): indicadores de si el Envído fue resuelto y/o bloqueado.
- **Puntuación de Envído** (2 valores normalizados): valor de Envído propio (siempre visible) y del oponente (solo si fue revelado tras aceptar la apuesta).
- **Mano** (2 bits): indicador del jugador que es “mano” en la ronda actual.

La dimensión total del tensor de observación es $|\mathcal{O}| = 321$ componentes, proporcionando una representación compacta pero suficientemente expresiva del estado de información de cada jugador.

3.1.2. Espacio de Acciones

El espacio de acciones de OpenTruco es discreto y de tamaño fijo $|\mathcal{A}| = 46$, compuesto por:

- **Acciones de carta** (40 acciones): cada una de las 40 cartas de la baraja española corresponde a una acción. En cada estado, solo son legales las cartas que el jugador posee y no ha jugado.
- **Acciones de Envído** (3 acciones): llamar Envído, Real Envído o Falta Envído. Solo son legales durante la primera mano y si no se ha resuelto previamente.

- **Acción de Truco** (1 acción): subir la apuesta de Truco (Truco \rightarrow Retruco \rightarrow Vale Cuatro). Legal cuando el jugador tiene derecho a subir.
- **Acciones de respuesta** (2 acciones): aceptar o rechazar una apuesta pendiente (Envído o Truco).

La máscara de acciones legales se actualiza dinámicamente según el estado del juego, respetando las reglas de precedencia entre apuestas y las restricciones de turno.

3.1.3. Recompensas y Reward Shaping

El objetivo fundamental del agente en OpenTruco es maximizar la probabilidad de ganar la partida. Esto implica modelar el episodio como la secuencia completa de manos desde el inicio hasta que un jugador alcanza los 30 puntos. Bajo esta formulación, las recompensas se definen de manera binaria:

$$R_{\text{terminal}}(s) = \begin{cases} +1 & \text{si el agente gana} \\ -1 & \text{si el agente pierde} \end{cases} \quad (1)$$

Esta definición garantiza que la política óptima maximice el *win rate*, independientemente del margen de puntos. Sin embargo, presenta un desafío práctico significativo: la dispersión de recompensas (*sparse rewards*). El agente solo recibe señal de aprendizaje al finalizar partidas que pueden extenderse por decenas de manos, dificultando la asignación de crédito (*credit assignment*) a las acciones individuales.

Una alternativa natural sería utilizar los puntos acumulados como recompensa intermedia, aprovechando que la diferencia de puntuación constituye una heurística informativa sobre el progreso del juego. No obstante, maximizar los puntos no es estrictamente equivalente a maximizar la tasa de victoria: un agente podría preferir estrategias conservadoras que aseguren puntos menores en lugar de arriesgar por victorias más decisivas.

Para reconciliar ambos objetivos —acelerar el entrenamiento con señales densas sin alterar la política óptima— implementamos *reward shaping* basado en potencial, siguiendo el marco teórico de Ng, Harada y Russell [13]. El teorema central de este enfoque establece que, dada una función de potencial $\Phi : \mathcal{S} \rightarrow \mathbb{R}$, la transformación de recompensas:

$$R'(s, a, s') = R(s, a, s') + \gamma \Phi(s') - \Phi(s) \quad (2)$$

preserva la política óptima del problema original. Es decir, la política que maximiza el retorno bajo R' también maximiza el retorno bajo R , garantizando *correctitud* teórica.

Para OpenTruco, definimos el potencial como la diferencia de puntuación normalizada:

$$\Phi(s) = \frac{\text{puntos}_{\text{agente}}(s) - \text{puntos}_{\text{oponente}}(s)}{30} \quad (3)$$



donde 30 es la puntuación objetivo del juego. Este potencial toma valores en $(-1, 1)$ y satisface la condición de ser cero en estados terminales, ya que la diferencia de puntos pierde relevancia una vez finalizada la partida.

La recompensa transformada proporciona señales más densas durante el entrenamiento: cada mano que incrementa la ventaja de puntuación genera una recompensa positiva proporcional, mientras que las pérdidas de puntos se penalizan inmediatamente. Crucialmente, las garantías teóricas del *potential-based reward shaping* aseguran que esta heurística acelera el aprendizaje sin modificar qué política resulta óptima.

La Figura 2 ilustra el efecto del *reward shaping* en la convergencia del entrenamiento, comparando la evolución de la pérdida con y sin la transformación de recompensas.

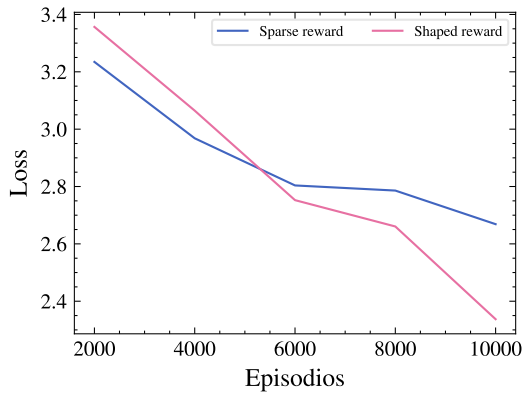


Figura 2: Comparación de curvas de pérdida con y sin *reward shaping* basado en potencial. El moldeado de recompensas acelera el aprendizaje del algoritmo NFSP.

3.1.4. Muestreo desde Conjuntos de Información

Varios algoritmos de OpenSpiel, incluyendo IS-MCTS y ciertas variantes de CFR, requieren la capacidad de generar estados del mundo consistentes con la observación actual del jugador. Esta funcionalidad se implementa mediante el método *ResampleFromInfostate*, que reconstruye un estado plausible redistribuyendo las cartas ocultas del oponente.

La implementación presenta una complejidad particular derivada de la mecánica del Envío. Cuando el Envío es aceptado y resuelto, ambos jugadores revelan su puntuación. Este valor pasa a formar parte de la información pública y, por tanto, cualquier estado muestreado debe respetar esta restricción: las cartas asignadas al oponente deben producir exactamente la puntuación de Envío revelada.

Para garantizar esta consistencia, el algoritmo de muestreo:

1. Identifica las cartas ya jugadas por el oponente (información pública).
2. Enumera las combinaciones de cartas disponibles que, junto con las cartas jugadas, producen la puntuación de Envío requerida.
3. Selecciona uniformemente una combinación válida y la asigna a la mano del oponente.

Este procedimiento preserva la coherencia del estado de información y permite que algoritmos basados en muestreo operen correctamente en presencia de información revelada condicionalmente.

3.2. Entrenamiento de Agentes

El objetivo de esta sección es demostrar que el entorno OpenTruco permite entrenar agentes que exhiben aprendizaje progresivo, validando así la correcta implementación del entorno y su compatibilidad con los algoritmos de OpenSpiel.

3.2.1. Configuración Experimental

Entrenamos agentes utilizando *Neural Fictitious Self-Play* (NFSP), seleccionado por su balance entre fundamentos teóricos sólidos y eficiencia computacional. El entrenamiento se realizó durante 10000 pasos del algoritmo.

Se evaluaron seis configuraciones de hiperparámetros, variando:

- El tamaño de las capas ocultas de las redes neuronales.
- El parámetro de anticipación η , que controla la mezcla entre la política de mejor respuesta y la política promedio.

Durante el entrenamiento, se registraron las pérdidas (*loss*) de ambas redes neuronales y se evaluó periódicamente el rendimiento del agente contra dos *baselines*: un agente que selecciona acciones uniformemente al azar (*Random*) y un agente basado en IS-MCTS con 100 simulaciones por decisión.

3.2.2. Resultados

La Figura 5 presenta la evolución de la pérdida de la red de política promedio durante el entrenamiento. Se observa una reducción consistente de la pérdida en todas las configuraciones, indicando que la red aprende a imitar la distribución de acciones del agente a lo largo del tiempo.

Las Figuras 3 y 4 muestran la tasa de victoria del agente NFSP contra los *baselines*. Contra el agente aleatorio, la mejor configuración alcanza un 70 % de tasa de victoria tras 10,000 episodios. Contra IS-MCTS, las configuraciones más efectivas logran tasas de victoria del 65 %, demostrando que los agentes logran superar a un oponente basado en búsqueda.

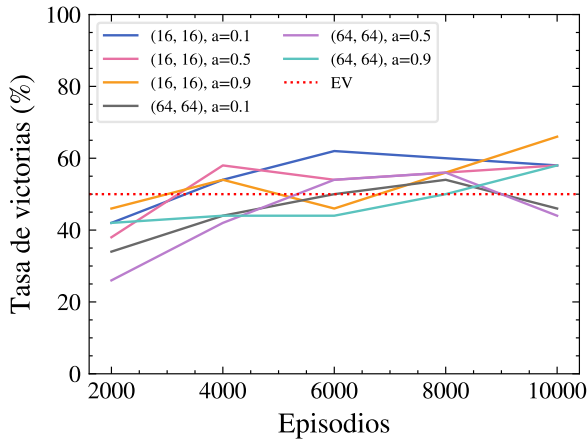


Figura 3: Tasa de victoria contra agente aleatorio durante el entrenamiento.

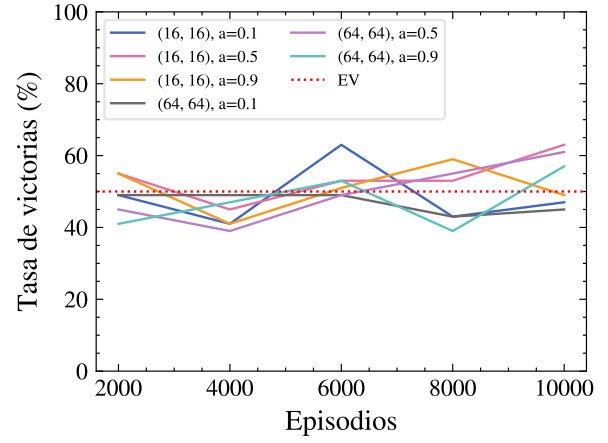


Figura 4: Tasa de victoria contra IS-MCTS (100 simulaciones).

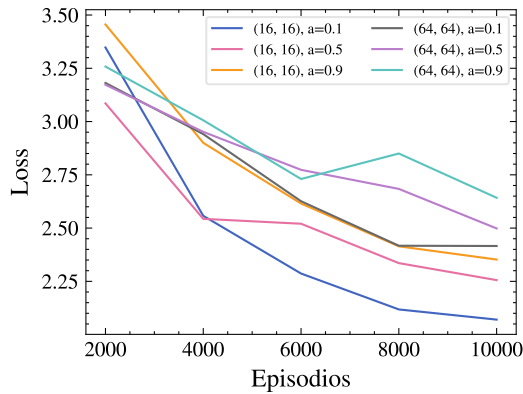


Figura 5: Evolución de la pérdida de la red de política promedio durante el entrenamiento de NFSP. Todas las configuraciones exhiben una tendencia decreciente, indicando aprendizaje efectivo.

4. Conclusiones

En este trabajo presentamos OpenTruco, un entorno de aprendizaje por refuerzo para el Truco Argentino desarrollado como extensión del *framework* OpenSpiel. La implementación modela fielmente las reglas oficiales de la modalidad 1 vs. 1 sin Flor a 30 puntos, incluyendo las mecánicas de apuestas (Truco y Envío) que caracterizan la riqueza estratégica del juego.

La principal contribución de este trabajo es el desarrollo de un entorno funcional y validado para un juego de información imperfecta con dinámicas de *bluffing* y *betting*. Los experimentos realizados con NFSP demuestran que los agentes exhiben aprendizaje progresivo: las curvas de pérdida decrecen consistentemente, y las tasas de victoria contra *baselines* (aleatorio e IS-MCTS) mejoran a lo largo del entrenamiento. Esto confirma que OpenTruco está correctamente implementado y es compatible con la infraestructura de algoritmos de OpenSpiel.

4.1. Trabajo Futuro

En cuanto a las posibles extensiones y mejoras del proyecto llevado a cabo, identificamos las siguientes líneas de desarrollo futuro.

Entrenamiento exhaustivo de agentes. Los experimentos presentados constituyen una validación del entorno más que una búsqueda de políticas óptimas. Un estudio más profundo debería explorar sistemáticamente los algoritmos disponibles en OpenSpiel: MCCFR y Deep CFR para aproximar equilibrios de Nash con garantías teóricas, y métodos de RL con *self-play* (DQN, PPO) para explorar políticas alternativas. Esta comparación permitiría identificar qué paradigma resulta más efectivo para el dominio del Truco.

Interfaz gráfica para evaluación humana. Las métricas automáticas contra *baselines* algorítmicos no capturan completamente la calidad de un agente de Truco. Las dinámicas de apuestas y *bluffing* son inherentemente humanas: un buen jugador de Truco no solo maximiza su tasa de victoria, sino que también “lee” al oponente y adapta su estrategia de engaño. Desarrollar una interfaz que permita enfrentar agentes entrenados contra jugadores humanos proporcionaría una evaluación más fidedigna de su rendimiento y revelaría si los agentes aprenden comportamientos de *bluffing* emergentes.



Referencias

- [1] M. Lanctot, E. Lockhart, J.-B. Lespiau et al., *OpenSpiel: A Framework for Reinforcement Learning in Games*, 2020. arXiv: 1908.09453 [cs.LG]. dirección: <https://arxiv.org/abs/1908.09453>.
- [2] Asociación Argentina de Truco, *Reglamento Oficial de Truco Argentino*, Consultado vía Wayback Machine (Internet Archive), ASART, 2022. dirección: <https://web.archive.org/web/20220102235725/https://asart.com.ar/wp-content/uploads/reglamento/reglamento-oficial-asart.pdf>.
- [3] Y. Shoham y K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2008.
- [4] J. F. Nash, "Equilibrium points in n-person games," *Proceedings of the National Academy of Sciences*, vol. 36, n.º 1, págs. 48-49, 1950.
- [5] M. Zinkevich, M. Johanson, M. Bowling y C. Piccione, "Regret Minimization in Games with Incomplete Information," en *Advances in Neural Information Processing Systems*, vol. 20, Curran Associates, Inc., 2007.
- [6] M. Lanctot, K. Waugh, M. Zinkevich y M. Bowling, "Monte Carlo Sampling for Regret Minimization in Extensive Games," en *Advances in Neural Information Processing Systems*, vol. 22, Curran Associates, Inc., 2009.
- [7] N. Brown, A. Lerer, S. Gross y T. Sandholm, "Deep Counterfactual Regret Minimization," en *International Conference on Machine Learning*, PMLR, 2019, págs. 793-802.
- [8] J. Heinrich y D. Silver, "Deep Reinforcement Learning from Self-Play in Imperfect-Information Games," *arXiv preprint arXiv:1603.01121*, 2016.
- [9] R. S. Sutton y A. G. Barto, *Reinforcement Learning: An Introduction*, 2.^a ed. Cambridge, MA: MIT Press, 2018.
- [10] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, n.º 7540, págs. 529-533, 2015.
- [11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford y O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [12] L. Kocsis y C. Szepesvári, "Bandit based Monte-Carlo planning," en *European Conference on Machine Learning*, Springer, 2006, págs. 282-293.
- [13] A. Ng, D. Harada y S. J. Russell, "Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping," en *International Conference on Machine Learning*, 1999. dirección: <https://api.semanticscholar.org/CorpusID:5730166>.