Toma de Decisiones, Inteligencia Artificial & Poker

Un Análisis Integral Moderno

Por: Tu Nombre

26 de febrero de 2025

La toma de decisiones bajo incertidumbre forma parte del día a día: desde pequeñas interacciones sociales hasta la elección de un determinado medio de transporte. Es cierto que, gracias a la tecnología, cada vez tenemos más información al alcance, y esta es cada vez más precisa. Sin embargo, siempre habrá información o conocimiento que quede fuera de nuestro alcance, tanto de forma individual como colectiva. Además, en el mundo real, el caos se involucra en el proceso: hay aleatoriedad y desviaciones respecto al resultado esperado; varianza. En algunas circunstancias, el caos puede ser tan predominante que lo esperado es no obtener el resultado esperado.

Esto representa un problema en el aprendizaje, no solo por las veces que algo sale mal cuando debería salir bien, sino también por las veces en las que algo sale bien por casualidad y lo atribuimos a una buena decisión. De manera puntual, se puede tomar la mejor decisión y perder, y se puede tomar la peor decisión y ganar... Entonces, ¿cómo aprendemos a tomar la mejor decisión?

> *"La vida real no es como el ajedrez. La vida real consiste en farolear, en pequeñas tácticas de engaño, en preguntarse qué va a pensar el oponente que pretendo hacer. De eso tratan los juegos en mi teoría."*
>
> John von Neumann

No es trivial averiguarlo, pero al menos parte de la respuesta se puede comprender mediante la teoría de juegos. En cada situación, debemos considerar lo que sabemos, lo que queremos lograr y la estrategia que maximizará nuestra utilidad esperada. La utilidad recibida en la mayoría de los escenarios del mundo real no se corresponde con la utilidad esperada y tampoco depende únicamente de nuestras acciones.

Se han desarrollado varios métodos de optimización en escenarios que pueden modelizarse como juegos de información incompleta, y en los últimos años, el que ha tenido mejor desempeño en su aplicación es el algoritmo *Counterfactual Regret Minimization* (CFR). El CFR busca encontrar soluciones óptimas en uno de los escenarios que mejor representan este caso: el póquer. A lo largo de este trabajo, se discutirá la implementación del algoritmo en este escenario, así como sus variantes y alternativas.

1. Introduction

1.1. Evolución de la toma computacional de decisiones en juegos

Los juegos, tradicionalmente ligados a la estrategia militar y política, y más recientemente al comercio y la economía, han demostrado ser un escenario idóneo para estudiar la toma de decisiones. En el *ámbito* computacional, la optimización en la toma de decisiones ha ido de la mano con la mejora de los sistemas de aprendizaje automático, una de las tres ramas fundamentales de la inteligencia artificial.

Históricamente, el ajedrez fue pionero en la investigación de la inteligencia artificial aplicada a los juegos. Existen referencias al siglo XVIII con el famoso *Turco Mecánico*, que resultó ser una ilusión mecánica y no un autómata real. Sin embargo, los primeros modelos computacionales capaces de resolver de forma efectiva posiciones sencillas de ajedrez surgieron en la primera mitad del siglo XX. Con la aparición de las computadoras en la década de 1950, se intensificó la carrera por desarrollar programas de ajedrez cada vez más fuertes.

Aunque las primeras técnicas se basaban casi exclusivamente en la fuerza bruta de cálculo, los investigadores pronto introdujeron estrategias de poda (como alfa-beta) y heurísticas más sofisticadas. El hito más destacado en esta línea de investigación fue el encuentro entre *Deep Blue* e *IBM* contra Garry Kasparov en 1997, en el que se mostró que una búsqueda combinada con heurísticas y funciones de evaluación diseñadas por grandes maestros podía superar a un campeón mundial.

La cuestión nunca fue enseñar a una máquina a *razonar* el ajedrez del mismo modo que lo hace un ser humano, sino más bien enseñarle a calcular de forma extremadamente eficiente. En los últimos años, la investigación ha continuado en la optimización de funciones de búsqueda y en el uso de modelos de aprendizaje profundo para aproximar el valor de una posición. Esto ha llevado al ajedrez y a otros juegos de información completa a niveles inalcanzables para la

mayoría de los jugadores humanos de manera sistemática (por ejemplo, los logros de AlphaGo en el Go [**?**]).

Mientras que el ajedrez y otros juegos de información completa han sido dominados por métodos de búsqueda y evaluación, los juegos de información imperfecta presentan un desafío distinto: la incertidumbre sobre el estado real del juego. En este contexto, el análisis probabilístico y el aprendizaje basado en experiencia juegan un papel central. En estos escenarios, no es suficiente calcular la *mejor* jugada basándose en el estado actual, pues en muchos casos desconocemos información crucial y no podemos identificar con precisión la situación exacta en la que nos encontramos. Así, la historia previa y las decisiones anteriores de todos los jugadores determinan una distribución de probabilidad sobre el conjunto de estados posibles.

Entre los juegos de información imperfecta, el póquer ha sido el predilecto para la investigación. La variante *Texas No-Limit Hold'em* se consideraba un reto inabordable hasta 2015, cuando el equipo CPRG de la Universidad de Alberta desarrolló el programa *Cepheus*, incorporando una variante del algoritmo *Counterfactual Regret Minimization* (CFR), conocida como *CFR+*[Tuomas Sandholm, 2019]. Cepheus logró una estrategia *exploit-proof* en la modalidad *Heads-Up Limit Hold'em*, lo que implica que ningún oponente podría explotarlo de manera sistemática a largo plazo. Sin embargo, su alcance no se extendía a la versión *No-Limit*, que es considerablemente más compleja en términos de espacio de acciones y estrategias.

Desde entonces, varias iniciativas han continuado perfeccionando las técnicas de CFR y sus variaciones. Destacan proyectos como *Libratus* (2018) y *Pluribus* (2019), desarrollados por Tuomas Sandholm y Noam Brown en la Universidad Carnegie Mellon, que han establecido nuevos hitos en la resolución parcial de variantes más complejas de póquer.

2. Estado del Arte

2.1. Estado del arte – Resolución de juegos de información incompleta

2.1.1. Enfoque Tabular

El algoritmo CFR se ha consolidado como el principal marco de trabajo (*framework*) para la resolución de juegos de información imperfecta, especialmente en el póquer. En la mayoría de los estudios iniciales, se empleaba una implementación tabular, donde cada conjunto de información (*information set*) se representaba explícitamente. Debido a la enorme complejidad del póquer, era necesario aplicar técnicas de abstracción que redujeran el espacio de estados (por ejemplo, agrupar manos similares o situaciones análogas). Un ejemplo notable es el trabajo de DeepStack[**?**], que combinaba la aproximación tabular con un valor aproximado en algunas partes del árbol de decisión.

Tras la abstracción, se entrena el modelo iterando el CFR sobre el árbol reducido, lo que permite a la máquina refinar su estrategia al actualizar los valores de arrepentimiento (*regret values*) para cada acción posible en cada *information set*. El resultado es una estrategia mixta (probabilística) que, en teoría, converge hacia un equilibrio de Nash en el modelo reducido.

2.1.2. Enfoque con Redes Neuronales

Los avances más recientes apuntan a sustituir estas estructuras tabulares por redes neuronales profundas, aprovechando la capacidad de generalización que ofrece el aprendizaje profundo para representar las funciones de valor y las estrategias. Así, en lugar de almacenar de forma explícita la información en tablas, el sistema entrena una red que predice la acción o el valor esperado en base al contexto. Entre los trabajos más influyentes en esta línea se encuentran *Deep CFR*[**?**]

y *Double-Neural CFR*[**?**], donde se incrementa la escalabilidad a la vez que se busca mantener niveles de rendimiento competitivos o superiores a las implementaciones tabulares.

Una de las motivaciones para utilizar redes neuronales es la posibilidad de manejar espacios de estado e información dinámicos y de mayor dimensionalidad, evitando la necesidad de manuales (y a veces arbitrarias) abstracciones para cada situación. No obstante, la capacitación de modelos profundos trae aparejados retos en términos de requerimientos computacionales y la potencial dificultad de asegurar convergencia o estabilidad en el aprendizaje.

3. Teoría de Juegos

La Teoría de Juegos es una rama de las matemáticas y la economía que estudia la toma de decisiones en contextos estratégicos, donde el resultado para cada participante depende de las acciones de los demás. Surgida inicialmente en problemas militares y económicos, hoy sus aplicaciones se extienden a la política, la biología evolutiva, la informática, la psicología y muchas otras áreas. Este capítulo aborda los conceptos fundamentales, las representaciones formales y la clasificación de los juegos, así como ejemplos prácticos que ilustran la aplicación de estos modelos.

3.1. Conceptos Fundamentales

Un juego se puede modelar en forma de árbol de decisión y se compone de varios elementos esenciales:

- **Jugadores:** Los agentes que toman decisiones.
- **Estados:** Las diferentes situaciones o fases del juego.
- **Reglas:** El conjunto de normas que determinan la transición entre estados, el orden de juego y la asignación de pagos.
- **Recursos:** Elementos tangibles o intangibles (como piezas de ajedrez o fichas) con los que interactúan los jugadores.
- **Pagos:** Las recompensas o penalizaciones, expresadas numéricamente, que se asignan en los estados terminales.

La mayoría de los juegos son no cooperativos, lo que implica un conflicto de intereses: la ganancia de un jugador se traduce en la pérdida de otro (como en los juegos de suma cero).

Además, se pueden clasificar según:

- **Individuales vs. Cooperativos**
- **Información Completa vs. Información Incompleta**
- **Suma Cero vs. Suma Distinta de Cero**

Otros conceptos clave que se abordarán son la estrategia, el equilibrio y la solución del juego.

3.2. Representación: 0̈0c1rboles de Decisión y Forma Extensiva

Existen dos representaciones principales en Teoría de Juegos:

- **Forma Normal:** Representación matricial, ideal para juegos simultáneos.
- **Forma Extensiva:** Representación en forma de árbol, ideal para juegos secuenciales.

En este capítulo nos centraremos en la forma extensiva.

3.2.1. Definición y Estructura de un Árbol de Decisión

Un árbol de decisión es un modelo gráfico que representa:

- **Nodos:** Representan los estados o puntos de decisión.
- **Aristas:** Representan las acciones o transiciones entre estados.
- **Nodos Terminales:** Indican el fin del juego, donde se asignan los pagos.

Esta representación facilita el análisis de la secuencia de decisiones y permite evaluar estrategias en cada etapa del juego.

3.2.2. Comparación entre Forma Normal y Forma Extensiva

Mientras que la forma normal se utiliza para representar juegos en los que las decisiones se toman simultáneamente, la forma extensiva es especialmente útil para juegos secuenciales, pues:

- Permite visualizar el orden de las decisiones.
- Facilita la incorporación de información asimétrica entre jugadores.

3.2.3. Abstracción y Reducción de Nodos

Para optimizar el análisis y la implementación computacional (por ejemplo, en algoritmos como el *Counterfactual Regret Minimization* o CFR), es crucial abstraer y reducir el número de nodos del árbol. Esto implica agrupar situaciones equivalentes y eliminar redundancias sin perder la esencia del juego.

3.3. Información Incompleta y Conjuntos de Información (Information Sets)

En muchos escenarios, los jugadores no disponen de toda la información relevante. Se distinguen dos conceptos:

- **Información Incompleta:** Algunos elementos fundamentales (por ejemplo, la mano del oponente) son desconocidos.
- **Información Imperfecta:** Aunque la estructura del juego es conocida, la información sobre ciertos eventos es limitada.

3.3.1. Definición de Information Sets

Un **information set** es una agrupación de nodos del árbol de decisión que, para un jugador, son indistinguibles entre sí. Esta agrupación permite que, a pesar de la incertidumbre, el jugador formule una estrategia que minimice las pérdidas potenciales.

3.3.2. Ejemplo Visual

En la figura a continuación se muestra cómo se representan los information sets en un juego secuencial, donde se agrupan nodos que son indistinguibles para el jugador:

3.4. Ejemplo Práctico: Piedra, Papel y Tijeras en Forma Extensiva

Consideremos el clásico juego de Piedra, Papel y Tijeras con las siguientes reglas:

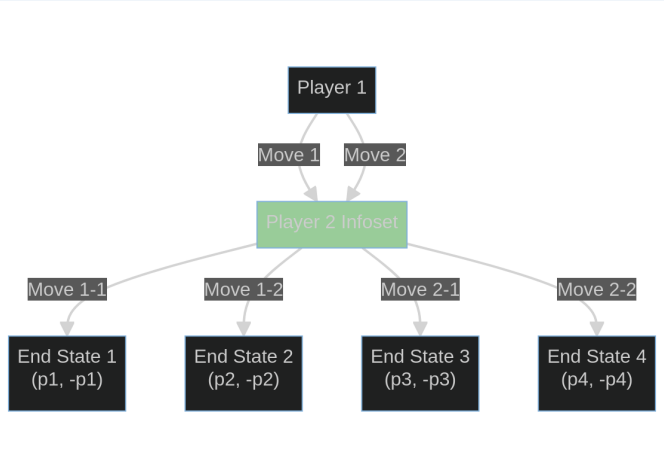

Figura 3.1: Ejemplo ilustrativo de un juego representado mediante **information sets** para capturar la información incompleta.

- Si ambos jugadores eligen la misma acción, recuperan su apuesta.
- Si un jugador elige piedra y el otro papel, gana quien eligió papel (+1 ud) y pierde el otro (-1 ud).
- Si un jugador elige tijeras y el otro papel, gana quien eligió tijeras (+1 ud) y pierde el otro (-1 ud).
- Si un jugador elige piedra y el otro tijeras, gana quien eligió piedra (+1 ud) y pierde el otro (-1 ud).

3.4.1. Representación en Forma Extensiva

El juego se representa mediante un árbol en el cual:

- Cada nodo corresponde a una decisión de un jugador.
- Se emplean **information sets** para representar la incertidumbre, ya que, por ejemplo, el jugador 2 no conoce la acción seleccionada por el jugador 1 al tomar su decisión.

3.4.2. Análisis del Equilibrio

Debido a la existencia de un número finito de estrategias puras (cada jugador elige entre piedra, papel o tijeras), se garantiza la existencia de al menos un equilibrio de Nash en estrategias mixtas. En este juego:

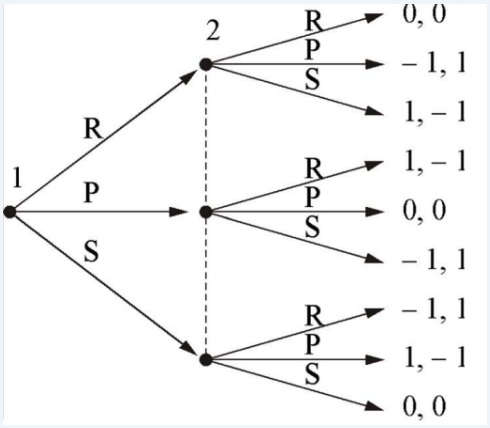

Figura 3.2: Representación en forma extensiva del juego de Piedra, Papel y Tijeras

- No es posible un equilibrio en estrategias puras, ya que la fijación de una acción por parte de un jugador puede ser explotada por el oponente.
- El equilibrio óptimo consiste en que ambos jugadores seleccionen cada una de las tres opciones con igual probabilidad, evitando así que uno pueda explotar la estrategia del otro.

3.5. Estrategia y Perfil Estratégico

La **estrategia** de un jugador, denotada por σ_i, es un vector probabilístico que asigna a cada acción disponible una probabilidad de ser elegida en un estado particular. El conjunto de estrategias de todos los jugadores constituye el **perfil estratégico** σ.

3.5.1. Estrategias Puras y Mixtas

- **Estrategia Pura:** El jugador asigna probabilidad 1 a una única acción en cada estado.
- **Estrategia Mixta:** Se asignan probabilidades positivas a dos o más acciones.

Formalmente, para un juego $\mathcal{J}$ y un estado $E \in \mathcal{J}$, siendo $A(E)$ el conjunto de acciones en ese estado, la estrategia del jugador i se expresa como:

$$\sigma_i = [p(a_0), p(a_1), \ldots, p(a_{n-1})],$$

donde $n = |A(E)|$ y cada $a_j \in A(E)$ con $p(a_j)$ representando la probabilidad de seleccionar la acción a_j.

3.6. Solución y Equilibrio

La resolución de un juego implica determinar un perfil estratégico que permita predecir:

- Las decisiones que tomarán los jugadores en cada situación.
- El resultado esperado a largo plazo.

Un perfil estratégico en el que ningún jugador tiene incentivo para desviarse de forma unilateral es denominado **equilibrio**.

3.6.1. Equilibrio de Nash

El **equilibrio de Nash** es aquel perfil estratégico σ en el que:

- Ningún jugador puede mejorar su ganancia esperada cambiando su estrategia de forma unilateral.
- Existe un conocimiento mutuo de las estrategias y de los pagos esperados.

En aplicaciones prácticas, como en el póquer, jugar según el equilibrio de Nash garantiza que, a largo plazo, no se obtendrán pérdidas, salvo que el oponente cometa errores estratégicos.

Ejemplo en Piedra, Papel y Tijeras

Si un jugador adopta una estrategia mixta no uniforme (por ejemplo, asignando probabilidades diferentes a piedra, papel y tijeras), el oponente puede explotar esta debilidad eligiendo de forma pura la acción que contrarresta la opción más frecuente. Así, el único equilibrio estable es que ambos jugadores jueguen de manera uniforme, asignando igual probabilidad a cada acción.

3.6.2. Equilibrio Bayesiano de Nash

En juegos con información incompleta, cada jugador debe formar creencias sobre el estado del juego y las acciones del oponente. En un **equilibrio bayesiano de Nash**, cada jugador maximiza su ganancia esperada dadas sus creencias y las estrategias adoptadas por los demás.

3.6.3. Equilibrio Bayesiano Perfecto

Este es un refinamiento del equilibrio bayesiano, que además de exigir la maximización de la ganancia esperada, impone condiciones adicionales en cada *information set*:

- **Estrategia:** Una distribución de probabilidad sobre las acciones disponibles, que depende del historial del juego.
- **Creencias:** Una asignación de probabilidades sobre los nodos del *information set*, basada en la información previa y el perfil estratégico conocido.

3.7. Clasificación de Juegos

Los juegos pueden clasificarse según diversos criterios, lo que permite enfocar el análisis en aspectos particulares de cada modelo.

3.7.1. Según la Información Disponible

- **Juegos de Información Completa:** Todos los jugadores conocen todas las acciones y los pagos posibles.
- **Juegos de Información Incompleta (Bayesianos):** Existen aspectos fijos del juego que son desconocidos para algunos jugadores.

3.7.2. Según la Información Pública de los Eventos

- **Juegos de Información Perfecta:** Cada jugador tiene conocimiento de todas las acciones previas y sus consecuencias.
- **Juegos de Información Imperfecta:** Al menos un jugador posee información limitada o privilegiada sobre ciertos eventos.

3.7.3. Según el Origen de los Incentivos

- **Juegos de Suma Cero:** La ganancia de un jugador es exactamente la pérdida del otro.
- **Juegos de Suma Distinta de Cero:** Se pueden generar situaciones de cooperación en las que la suma total de pagos es mayor.

3.7.4. Otras Clasificaciones

Adicionalmente, los juegos se pueden clasificar en:

- **Juegos Estáticos vs. Dinámicos:** Según si las decisiones se toman simultáneamente o de manera secuencial.
- **Juegos Simétricos vs. Asimétricos:** Dependiendo de si las estrategias y roles de los jugadores son equivalentes.

4. Regret Learning

4.1. Aprendizaje por Refuerzo

El aprendizaje por refuerzo consiste en una metodología de aprendizaje automático centrada en la idea de que un agente aprende mediante la interacción con un entorno, recibiendo **recompensas** o **penalizaciones** que guían la corrección de su comportamiento. Este proceso, enraizado en la psicología conductista, se fundamenta en principios básicos como el refuerzo positivo, el castigo y el contracondicionamiento. Al percibir un incentivo por las acciones acertadas y sanciones por las acciones erróneas, el agente ajusta gradualmente su estrategia o política con el objetivo de maximizar su ganancia acumulada en el largo plazo.
Existen diferentes aproximaciones al aprendizaje por refuerzo, entre ellas:

- **Aprendizaje por Refuerzo Supervisado:** En este enfoque, el agente recibe ejemplos de comportamiento deseado. Dichos ejemplos se presentan como pares $((\text{estado}, \text{acción}), \text{recompensa})$. Con base en estos datos, el agente aprende a replicar las decisiones que llevaron a mejores recompensas en el conjunto de entrenamiento, imitando en cierta forma la lógica de un modelo supervisado clásico.
- **Aprendizaje por Refuerzo No Supervisado:** A diferencia del caso anterior, aquí el agente no cuenta con ejemplos explícitos de qué acciones son deseables. En su lugar, explora el entorno mediante *prueba y error*, acumulando experiencia basada en retroalimentaciones (recompensas y penalizaciones) hasta determinar una estrategia que maximice la recompensa total. Ejemplos destacados incluyen el trabajo de *AlphaZero* y *Pluribus*, que combinan estas ideas con otras técnicas de reducción de complejidad y representación.

4.1.1. Algoritmos basados en la minimización del arrepentimiento

La *minimización del arrepentimiento* (*regret minimization*) constituye una de las piedras angulares del aprendizaje por refuerzo en contextos de teoría de juegos. La idea principal radica en evaluar cuán grande habría sido la ganancia si, en lugar de haber elegido la acción actual, se hubiese optado por otra acción más provechosa. Esa diferencia constituye el **arrepentimiento** o **regret**. Al entrenar un agente para minimizar sistemáticamente este arrepentimiento, logramos que su estrategia converja a un comportamiento cada vez más óptimo.

En juegos de información incompleta muy extensos, donde el **árbol de decisión** es de un tamaño inabordable, el condicionamiento se hace especialmente desafiante. Para afrontar esta dificultad, se introducen los *information sets*, o conjuntos de información, que agrupan nodos (estados) que son indistinguibles para cierto jugador. El objetivo es minimizar el arrepentimiento en cada *information set*, de manera que, al cabo de suficientes iteraciones, el perfil estratégico resultante se aproxime a un ϵ-equilibrio.

Los algoritmos de *regret minimization* generan un **equilibrio correlacionado** en muchos casos, tal y como se discute en [**?**]. Concretamente, el algoritmo CFR (*Counterfactual Regret Minimization*), descripto en próximas secciones, garantiza un subequilibrio cuando el número de iteraciones tiende a ser grande.

Ventajas:

- Permite resolver grandes juegos de información incompleta usando abstracciones.
- Con la debida reducción de complejidad, converge de forma eficiente a estrategias poco explotables.

Desventajas:

- Requiere almacenar los arrepentimientos y las estrategias para cada *information set*, lo cual puede demandar mucha memoria en juegos muy complejos.
- Algunas variantes necesitan recorrer buena parte del árbol de decisión en cada iteración, lo que incrementa el tiempo de cómputo.

4.2. Regret Matching

Regret Matching es un algoritmo de aprendizaje por refuerzo de carácter precursor al CFR. Su importancia radica en que, si cada jugador de un juego adopta esta metodología para actualizar sus estrategias, el perfil estratégico global converge a un equilibrio de Nash conforme el número de iteraciones tiende al infinito.

4.2.1. Idea general: minimización del arrepentimiento

La mecánica central de *Regret Matching* consiste en acumular los *regrets* o arrepentimientos de cada acción no elegida en cada iteración. Formalmente, si observamos que una acción distinta habría dado un mejor resultado en un cierto estado, aumentamos el peso de dicha acción. De esta forma, a lo largo de múltiples iteraciones, las acciones más prometedoras cobran más protagonismo en la estrategia final.

4.2.2. Definiciones formales

Para un jugador i, su **valor contrafactual** en un *infoset* I, según un perfil estratégico σ, se define como:

$$v_i(\sigma, I) = \sum_{z \in z[I]} \Big(u_i(z) \cdot \pi^{\sigma}_{-i}(z[I]) \cdot \pi^{\sigma}(z[I], z) \Big), \tag{4.1}$$

siendo $z[I]$ el conjunto de nodos terminales accesibles desde I, $u_i(z)$ la utilidad para el jugador i en el nodo terminal z, y $\pi^{\sigma}_{-i}(z[I])$ y $\pi^{\sigma}(z[I], z)$ las probabilidades de llegar hasta z según el perfil estratégico de todos los jugadores (excluyendo e incluyendo al jugador i, respectivamente).

El **arrepentimiento contrafactual** de una acción a en el infoset I en la iteración t es:

$$r^t_i(I, a) = v_i(\sigma^t_{I \to a}, I) \; - \; v_i(\sigma^t, I), \tag{4.2}$$

donde $\sigma^t_{I \to a}$ es el perfil estratégico de la iteración t modificado para forzar la elección de la acción a en el infoset I. El **arrepentimiento contrafactual acumulado** tras T iteraciones se define como:

$$R^T_i(I, a) = \sum_{t=1}^{T} r^t_i(I, a). \tag{4.3}$$

4.2.3. Actualización de estrategias (Regret Matching Rule)

En cada iteración, para cada acción a de I, se calcula una probabilidad proporcional a la parte positiva del arrepentimiento acumulado:

$$\sigma^{T+1}(I,a) = \begin{cases} \dfrac{\max\left(0, R_i^T(I,a)\right)}{\sum_{b\in A(I)} \max\left(0, R_i^T(I,b)\right)} & \text{si } \sum_{b\in A(I)} \max\left(0, R_i^T(I,b)\right) > 0, \\ \dfrac{1}{|A(I)|} & \text{en otro caso.} \end{cases} \tag{4.4}$$

El significado intuitivo de esta regla es: a mayor arrepentimiento positivo acumulado para una acción, mayor probabilidad de elegirla la próxima vez. Si todos los arrepentimientos resultan negativos o cero, se utiliza una estrategia uniforme.

4.2.4. Estrategia media y convergencia

Para garantizar la convergencia hacia un equilibrio de Nash, se toma la *estrategia media* a lo largo de las iteraciones, es decir, el promedio de las estrategias elegidas en cada paso. Así se evita la sobreexploitación al seleccionar únicamente la acción con mayor arrepentimiento positivo.

4.2.5. Ejemplo: Piedra, Papel y Tijeras

Sea el juego de Piedra, Papel y Tijeras descrito en el capítulo anterior. Aplicar *Regret Matching* implica lo siguiente:

1. Asignar un arrepentimiento inicial igual a 0 para las tres acciones {piedra, papel, tijeras}.
2. Jugar repetidas rondas del juego, registrando las recompensas obtenidas.
3. Para cada iteración, acumular en R_i^T cuánto se habría ganado de haberse escogido cada acción no realizada.
4. Actualizar la estrategia para la siguiente iteración según la regla de *Regret Matching*.
5. Con el paso del tiempo, el perfil estratégico medio converge a la estrategia *mixta uniforme*, que es el equilibrio de Nash para este juego.

4.3. De Regret Matching a CFR

Counterfactual Regret Minimization (CFR) puede considerarse una extensión más sofisticada de *Regret Matching*, que introduce mejoras en la forma de recorrer el árbol de decisión y calcular el arrepentimiento.
A continuación se destacan sus componentes fundamentales:

- **Recorrido secuencial del árbol:** En lugar de analizar simultáneamente cada acción y cada nodo terminal, CFR descompone el juego en **infosets** y avanza secuencialmente, actualizando los regrets de manera localizada.
- **Factorización de probabilidades:** Se distinguen las probabilidades del jugador que estamos analizando (π_i) de las del oponente (π_{-i}). Esto permite actualizar los arrepentimientos **contrafactuales**, que valoran la ganancia de 201cqué habría sucedido si...201d para cada acción.

4.3.1. Utilidad contrafactual en un infoset

Se define para un jugador i y un estado de información I la denominada *utilidad contrafactual*:

$$u_i(\sigma, I) = \frac{\sum_{h \in I,\, z \in Z} \pi^{\sigma}_{-i}(h)\, \pi^{\sigma}(h, z)\, u_i(z)}{\pi^{\sigma}_{-i}(I)}, \tag{4.5}$$

donde $\pi^{\sigma}_{-i}(h)$ es la probabilidad de llegar al historial h bajo la estrategia de los demás jugadores, y $\pi^{\sigma}(h, z)$ es la probabilidad de transitar desde h hasta un nodo terminal z. El denominador $\pi^{\sigma}_{-i}(I)$ representa la probabilidad de alcanzar el infoset I dadas las estrategias de los oponentes.

4.3.2. Arrepentimiento contrafactual inmediato

El **arrepentimiento contrafactual inmediato** tras T iteraciones se define:

$$R^{T}_{i,imm}(I) = \frac{\max\limits_{a \in A(I)} \sum\limits_{t=1}^{T} \pi^{\sigma^t}_{-i}(I)\big(u_i(\sigma^t|_{I \to a}, I) \;-\; u_i(\sigma^t, I)\big)}{T}. \tag{4.6}$$

Este valor mide cuán provechoso habría sido, en promedio, sustituir la acción efectiva por la acción a a lo largo de las iteraciones, ponderando la probabilidad de que I fuese alcanzado. Si garantizamos que este arrepentimiento disminuye, se puede demostrar que el **arrepentimiento global** también lo hace, lo cual lleva a la convergencia a un ϵ-equilibrio de Nash.

4.3.3. Variedades de CFR

Se han propuesto múltiples variantes de CFR para acelerar la convergencia o reducir la carga computacional:

- **Vanilla CFR:** Recorre exhaustivamente todas las ramas del juego en cada iteración. Garantiza altos niveles de precisión, pero es costoso en juegos extensos.
- **Chance-Sampled CFR:** Aleatoriza los sucesos o distribuciones de azar, de modo que el algoritmo actualiza los arrepentimientos de las ramas más probables. Reduce de manera significativa la complejidad en juegos como el póquer.
- **Monte Carlo CFR:** Emplea simulaciones Monte Carlo para explorar aleatoriamente partes del árbol, actualizando sólo los infosets visitados en esas muestras.
- **Deep CFR:** Utiliza redes neuronales para generalizar representaciones de estados y acciones, integrando la reducción de dimensionalidad con el proceso de regret minimization.

En los *state-of-the-art* en póquer computacional (como Libratus, Pluribus o DeepStack), se ha comprobado que las distintas variantes de CFR logran un desempeño notable, con tiempos de convergencia y necesidad de memoria gestionables a pesar de la enorme complejidad del juego.

5. Póquer

5.1. Reglas

El póquer es un juego de cartas que se juega con un mazo estándar de 52 cartas. El objetivo del juego es ganar fichas obteniendo la mejor combinación de cartas o provocando que el resto de juegadores abandonen la mano.

5.2. Modalidades

Existen diversas modalidades de póquer, cada una con sus propias reglas y estrategias.

5.2.1. Torneos

En los torneos de póquer, los jugadores jurgan entre ellos divididos en mesas de 6-9 jugadores hasta que uno de ellos tiene todas las fichas. Los pagos se asignan en función de la posición en la que ha sido eliminado un jugador, no siendo premiadas todas las posiciones. Los jugadores compran una entrada al torneo y reciben un número determinado de fichas inicial. A medida que los jugadores son eliminados, las mesas se reorganizan de forma dinámica la mesa final.

5.2.2. Cash

En los juegos de cash, los jugadores pueden comprar y retirar fichas en cualquier momento. El valor de las fichas es directamente proporcional al dinero real, y los jugadores pueden entrar y salir del juego cuando lo deseen.

5.3. Variantes

Atendiendo al límite máximo de apuesta, el póquer tiene varias variantes.

5.3.1. No Limit

En el póquer No Limit, no hay un límite en la cantidad que un jugador puede apostar. Un jugador puede apostar todas las fichas que dispone en juego en cualquier momento en el que le toque actuar, lo que se conoce como .ªll-in".

5.3.2. Limit

En el póquer Limit, hay un límite respecto al total en juego, en la cantidad que se puede apostar en cada ronda.

6. CFR en el póquer

6.1. Variante Push or Fold

En esta sección se aborda la implementación del CFR en el siguiente escenario: la variante del póquer No-Limit Hold'em (NLH) 'Push or Fold' de 2 jugadores con un stack para cada jugador de 10bb (big blinds - podemos ver esto como un stack de 10€ donde la ciega grande es 1€).

Al inicio de cada ronda, los jugadores hacen las apuestas obligatorias en función de su posición: 0.5bb para el jugador en primera posición ("small blind") y 1bb para el jugador en segunda posición ("big bling"). Por último, se reparte una mano (dos cartas) a cada jugador y comienza la ronda. El orden de juego es secuencial, y si el estado del juego no es terminal, el jugador que actúa, tiene dos opciones independientemente de su posición:

- Raisear all-in(r): subir la apuesta a todo su stack.
- Foldear(f): irse de la mano y perder su apuesta obligatoria.

Cuando llegamos a un estado terminal, se reparten los pagos. Si alguno de los dos jugadores ha foldeado los pagos son inmediatos, mientras que si ambos han ido all-in, se reparte el bote total proporcionalmente al valor estimado (EV) de la mano sostenida por cada jugador.

6.2. Abstracción

La abstracción consiste en agrupar situaciones (estados) similares para simplificar el árbol. En el caso del póquer se podrían agrupar distintos tamaños de apuesta en la misma situación, como si fueran la misma apuesta, reduciendo el árbol de juego considerablemente. Lo mismo con distintas

manos 5C-6C, 6C-5C, 6T-5T,.. = 65s, para distintas agrupaciones de cartas comunitarias, para distintos tipos de rivales en función de sus desviaciones (agresivo, pasivo, tight, loose..), etc.

Gracias a la librería Poker de PyPi, tenemos una representación de la tabla de manos iniciales posibles en el póquer. En total hay 169 combinaciones posibles iniciales para cada jugador, sin embargo, no todas tienen el mismo peso. En una baraja de 52 cartas y 4 palos, podemos tener 4 combinaciones de 'A' ,'K' en la mano SUITED (del mismo palo), y sin embargo, tendremos 16 combinaciones OFFSUITED (de diferentes palos). La matriz superior representa las combinaciones de cartas del mismo palo('s'), y la matriz inferior las combinaciones de cartas de diferentes palos('o').

6.2.1. Codificación del historial

El historial refleja de forma ordenada la información pública del juego, es decir: representa de forma secuencial las acciones de una partida. En un primer momento está vacío, y se va añadiendo un carácter conforme suceden las acciones: 'x' si se reparte una mano, 'r' si el jugador va all in, y 'f' si foldea. Este historial es esencial para el algoritmo CS-CFR, ya que permite rastrear las decisiones tomadas y calcular el arrepentimiento de cada acción.

6.2.2. Formato de una mano

Diferenciaremos los casos en los que las dos cartas sostenidas por el jugador son del mismo palo, o son de diferente palo. Usaremos la siguiente codificación:

-Si son del mismo palo se añade una *s* al final. Ejemplos: 'AKs', 'KQs'.

-Si son de distinto palo se añade una *o* al final. Ejemplos: 'AKo', 'KQo'.

6.2.3. Codificación de un Infoset

Para acceder a un infoset necesitaremos la información pública (el historial), y la información privada del propietario (la mano que está sosteniendo el jugador). Cada infoset estará definido por una clave string con el formato: manoActual + ' ' + historial. Esta codificación permite un acceso rápido y eficiente a la información necesaria para tomar decisiones en cada ronda de juego.

6.3. Implementación del CFR

6.4. Pruebas y obtención de resultados

6.5. Análisis y conclusiones

6.6. Aplicación en un escenario real

Bibliografía

[Tuomas Sandholm, 2019] Tuomas Sandholm, N. B. (2019). Libratus: Superhuman ai for heads-up no-limit poker. *Science*.