Práctica Interna

Un acercamiento Evolutivo de Aprendizaje Reforzado Profundo a Monopolio

Presentado por: Carlos Andres Coronado Arrazola

Fecha: 22 de diciembre de 2021

Universidad Privada Boliviana
Cochabamba - Bolivia

Tabla de Contenidos

Tabla de Figuras

Índice de Tablas

Capítulo 1

Presentación del problema

Una característica que siempre definió al ser humano es su inteligencia, no posee una gran fuerza o una alta resistencia, pero su increíble capacidad de aprendizaje fue el motivo de su rápida adaptación a entornos hostiles en sus inicios como especie. El conocimiento adquirido, en conjunto con su ingenio le permitió desarrollar herramientas para afrontar y superar exitosamente las distintas dificultades que se le presentaron. Además, con el desarrollo de la comunicación, el paso del conocimiento de generación en generación se volvió una realidad. Con siglos de conocimiento el aprendizaje se volvió nuestro poder.

Múltiples estudios indican que los inicios del homo-sapiens datan por alrededor de 300.000 años atrás, y aún con la transferencia de información a través de la comunicación; ya sea de voz en voz, o mediante la escritura en libros, papiros, quipus; los cambios más grandes en su historia empezaron a surgir en épocas recientes, más concretamente en el siglo XIX con la llegada de la primera revolución industrial. El impacto que causa es de tal magnitud que esta misma es demarcada como un punto de inflexión, no solo de la historia del avance tecnológico, sino de la historia de la humanidad; la cuál hasta el momento no había sido testigo de un crecimiento en la corriente de innovación tan grande. Un acontecimiento tan importante terminó trayendo consigo una época dorada de prosperidad. Trenes, teléfonos, automoviles, aviones, bombillas de luz; es complicado intentar comprender el hecho de que elementos cotidianos el día de hoy hayan sido ingeniados por mentes brillantes hace ya más de un siglo, inventos que fueron la primera escalera en el ascenso de la pendiente tecnológica que aparenta no tener fin aún con la tecnología de hoy en día. Las cadenas de producción, las primeras computadoras, la llegada del hombre a la luna, el internet, los modelos de Aprendizaje Automático (en inglés "Machine Learning"), los celulares; son algunos de los sucesos primordiales que dejaron su marca en la historia después de la primera revolución

industrial.

En la actualidad la humanidad se encuentra en la cuarta revolución que gira alrededor de la Inteligencia Artificial, campo que tiene sus principios en el año 1950 con el planteamiento de la pregunta *"¿las máquinas pueden pensar?"*, propuesta por Alan Turing en el inicio de su reconocido artículo *Computing Machinery and Intelligence*[1]. En el mismo artículo Turing esquema un juego, que más adelante se conocería como *"la prueba de Turing"*. El juego consiste de tres integrantes, dos individuos (A) y (B) de diferente sexo, con un interrogador (C). El objetivo de este juego es que (C) pueda identificar exitosamente quién es hombre y quién es mujer entre (A) y (B), a base de preguntas realizadas de manera individual. Partiendo de la pregunta *"¿las máquinas pueden pensar?"*, Turing replantea el juego, proponiendo que una máquina tome el lugar de (A) o (B) y que la nueva tarea del interrogador sea determinar quién es el humano y quién es la máquina, lo que da como resultado una nueva incógnita: *"¿podrá el interrogador (C) identificar correctamente a la máquina?"*. De la nueva pregunta surge un nuevo problema, y esta tiene la ventaja de dibujar una fina línea entre las capacidades físicas e intelectuales del humano [1], ¿es posible que las máquinas carguen con algo que pueda ser descrito como pensar pero que en esencia sea muy diferente a lo que el hombre hace?. Todas estas interrogantes no poseen una respuesta concreta aún, pero de lo que sí se está seguro hoy en día es que una máquina es capaz de aprender, lo que en conjunto con su poder de procesamiento, probablemente la convierta en la herramienta más poderosa vista por la humanidad.

Tras años de avance, actualmente (estando en la era del *"Big Data"*) se cuenta con modelos de Machine Learning que al ser alimentados con enormes cantidades de datos, son capaces de dar predicciones precisas o realizar una predicción correcta de datos futuros. Su versatilidad les permitió una gran variedad de aplicaciones, lo que desembocó en una rápida integración en las empresas, potenciando una nueva etapa de crecimiento gracias a la automatización de procesos, reducción en la tasa de fallo, etc. Además de esto, la Inteligencia Artificial, gracias al *Deep Learning*, logró conquistar tareas que se pensaba imposible para una máquina. Desde componer música a conducir un auto, como especie hemos logrado automatizar varias de las actividades que se atribuían como características de un ser humano, esto a base de enseñar a un programa de computadora a hacerlo, aunque aún no se tenga claro del porqué la Inteligencia Artificial es capaz de tales hazañas.

Así como no se tiene respuestas concretas a las preguntas desarrolladas por Turing, no existe igualmente una definición clara sobre lo que es exactamente la Inteligencia Artificial. Distintos conceptos aparecen dependiendo del campo y punto de vista desde el que se mire

al tema, pero una definición que se acerca al punto de vista de la informática define a la Inteligencia Artificial como *la automatización de la congnición [2]*, con cognición haciendo referencia a la compresión de conocimiento, en el que conocimiento es la compresión de información y la información es la compresión de datos. Basandose en la definición anterior y en lo mencionado sobre el *Big Data* se puede inferir que los datos son un aspecto de suma importancia en la Inteligencia Artificial. De estos dependerá el aprendizaje que el modelo tendrá. Existen tres tipos de Aprendizaje en la Inteligencia Artificial:

- Aprendizaje Supervisado
- Aprendizaje no Supervisado
- Aprendizaje Reforzado

En **Aprendizaje Supervisado** los datos contienen las características de una observación, o una entrada, e igualmente se tiene la salida de este mismo, o el *resultado*. Gracias a esto se puede entrenar al modelo, ajustándolo a los datos proveídos. En el **Aprendizaje no Supervisado** igualmente se tiene un set de datos, pero estos no contienen un resultado, por lo que no hay una guía concreta sobre que deben representar las observaciones. Lo que busca este tipo de aprendizaje es encontrar una *estructura* en estos datos. Para el **Aprendizaje Reforzado** no se tiene un set de observaciones, sino un ambiente y, contenido en este, un agente que va a aprender la manera óptima de interactuar con el ambiente que lo rodea a base de acciones permitidas en este mismo y que son decididas por el agente. Este tipo de aperndizaje puede llegar a se una herramienta muy poderosa, que cuenta con la capacidad de afrontar problemas en los que no se tiene un profesor o ente externo que dicte al modelo sobre que acciones son buenas o malas. De ahí es de donde viene el concepto de *reforzar* para el nombre de este tipo aprendizaje, pues las acciones positivas resultarán en una recompensa para el agente, reforzando dicha acción, de esa manera a base de esta experiencia el agente eventualmente llegará a conocer cuáles decisiones son las más favorecedoras. La hipótesis que se planteará más adelante funda sus bases en el aprendizaje reforzado.

Probablemente el ejemplo más conocido, y el que muestra el potencial del aprendizaje reforzado más claramente, es Alpha Go [3] que en el año 2015 fue noticia a nivel global al derrotar al campeón mundial de Go, Lee Sedol. Algo muy llamativo de este evento fue la propaganda de este mismo, en la que se mostraba a un clásico robot o androide con los ojos rojos y de aspecto maligno que refleja la visión general que la mayoría de las persona posee sobre la Inteligencia Artificial. La realidad sobre este tema es que, al menos a día de

hoy, la humanidad se encuentra lejos de desarrollar una Inteligencia Artificial que presente una amenaza. Así como lo fue el fuego hace miles de años, la Inteligencia Artificial es ahora la herramienta principal del progreso, y el campo del Aprendizaje Reforzado es uno de los más prometedores. Basicamente todo puede ser planteado como un problema de Aprendizaje Reforzado, con una analogía sobre cuál sería el ambiente del agente y los objetivos y decisiones posibles dentro de este ambiente, después de incontables iteraciones el agente tendrá una capacidad inhumana para desenvolverse exitosamente sobre el ambiente, todo sin escribir una línea de código extra. Además, se tiene la posibilidad de crear sistemas compuestos en los que haya varios agentes que tengan metas independientes pero que engranen y trabajen en conjunto para formar un sistema completo. Igualmente existen los sistemas multi-agente en los que se tiene multiples agentes que apunten a alcanzar con un trabajo cooperativo una meta global. Los campos sobre los que el aprendizaje reforzado puede ser aplicado son varios, algunos ejemplos son los siguientes:

- **Automatización industrial y la Robótica** ya sea en la planeación de procesos, control automatizado del equipo de una fábrica, control de calidad, etc.
- En la **Educación** como por ejemplo en un programa que aprenda cuáles son los errores más comunes en los estudiantes al momento de resolver problemas de cálculo, entonces muestre pautas de un texto o recomiende ejercicios que sobre el tema en el que se cometió el error.
- En la **Medicina y Salud** campo en el que ya es bastante usado para encontrar tratamientos óptimos, o para el uso del equipo médico.

De los modelos de aprendizaje reforzado aplicados en la industria, al menos los conceptos y técnicas utilizadas en estos vieron su nacimiento en los juegos. El mirar un juego desde un punto de vista de Aprendizaje Reforzado es bastante fácil, pues este debe tener un ambiente bien definido lo que hace fácil identificar la información relevante sobre este, también se tienen los objetivos y las reglas claras. Es debido a esto que los juegos son ideales para el surgimiento de ideas nuevas que después pueden ser extrapoladas a otras aplicaciones.

1.1. Descripción del problema

La mente humana ha sido capaz de crear una multitud de juegos, cada uno con acciones, estrategias, objetivos, ambientes y reglas propias. Elementos que distinguen a un juego del

resto otorgándole peculiaridades, que a la vez, dan la gracia al desarrollo de una partida. El afrontar el problema de plantear un modelo óptimo para un juego, con sus características particulares, supone todo un reto en cada ocasión, con algunos presentando mayor complejidad que otros, pero el mirar un juego análiticamente bajo un punto de vista de Aprendizaje Reforzado puede ayudar al surgimiento de ideas frescas en este campo. Hoy en día la pregunta que surge ya no es si la Inteligencia Artificial será capaz de desenvolverse en un juego dado y sobrepasar las expectativas humanas, la incognita a tomar en cuenta ahora es cómo debe ser planteado el modelo para dicho juego tal que le permita a la Inteligencia Artificial sobrepasar expectativas.

Actualmente, aún no se cuenta con un modelo capaz de alcanzar el nivel humano en juegos en los que el éxito depende de una interacción estrecha con los demás jugadores (Comercio, Carrera por obtener Territorio y/o Recursos, etc), además de la importante intervención de la suerte en el desarrollo de las partidas. Una investigación sobre este tema tiene el potencial de otorgar interesantes perspectivas sobre nuevas técnicas de implementación para un campo que no está muy deasrrollado. El problema que surge de esta falta de desarrollo es ¿qué modelado de Agente de Aprendizaje Reforzado es el más óptimo para juegos en los que el éxito depende de la suerte además de una interacción estrecha con los demás jugadores?

Dentro de este campo cae Monopolio [4], juego en el que la estrategia, el comercio y la aleatoriedad juegan un rol clave en el desarrollo de sus partidas. Monopolio es un juego con objetivos simples y claros, pero con un ambiente controlado por reglas complejas, las cuáles dan paso a desafíos que deben se rafrontadps al momento de diseñar un Agente Inteligente capaz de desenvolverse apropiadamente en el entorno.

Uno de estos deasfíos, culpables de la inexistencia del modelo, es la integración del comercio a la dinámica del juego. El agente debe de considerar los beneficios para sí mismo sin perder de vista los beneficios para sus oponentes en todo momento durante la partida. El comercio en general es poderoso, y bien utilizado garantiza una ventaja constante sobre los oponentes. El problema recae en cómo enseñar al agente a utilizarlo adecuadamente, contando además que en diferentes etapas de la partida se deberá de usar de forma distinta. Si el agente posee en este aspecto una *política de decisión*2.1.3 demasiado rígida puede comprometer la correcta toma de decisiones en el comercio. El reto es poder brindar al agente suficiente experiencia, teniendo cuidado de que no haya un BIAS[1] hacia ciertas situaciones, para brindarle mayor adaptabilidad a los cambios que surjan durante la partida.

[1]Prejuicio a favor o en contra de un objeto, persona o grupo comparado con otro, usualmente considerado injusto

Otra razón de la falta de un modelo óptimo es el ambiente cambiante, influenciado por la suerte, que afecta altamente al ya sensible agente inteligente, impiediendo el planteamiento de una estrategia general y estática. En Monopolio el comercio no es el único aspecto a tener en cuenta, hay decisiones clave a considerar que no necesariamente son partes de la estrategia a un inicio. Es decir, si la estrategia consiste en concentrarse en adquirir la propiedades del color rojo pero ya se tiene casi completo el color azul, puede llegar a ser más conveniente el centrar los recursos en completar el color azul primeramente, esto supone un cambio de estrategia que pudo ser influenciado por la naturaleza caótica de los dados, los cuales ayudaron a que el color favorecido sea el azul. Este comportamiento impredecible puede llegar a decidir el triunfo en la partida, razón por la que una adaptabilidad constante es necesaria. Pese a esto, el comercio debe de ser considerado activamente en el plan ganador, pero como una herramienta para obtener el bien mayor, la victoria.

El entorno, complejo a causa de las reglas del juego, también representa un problema. El agente inteligente debe de recibir información sobre el estado del entorno para tomar una acción en respuesta. Es decir, debe tener una forma de abstraer la información del estado entorno y tomar una acción favorecedora que el entorno permita para dicho estado. Para afrontar esto, métodos clásicos de aprendizaje reforzado como el Q-Learning [5] platean una tabla, con las columnas representando la acción y las filas representando los estados posibles en los que el ambiente se pueda encontrar. La cantidad de posibles estados en Monopolio es incontable, y eso representa un problema, pues ya no se puede aproximar una solución con simples métodos clásicos. Además, en primera instancia, se debe decidir que información puede recibir el agente (la cuál debería de ser la información a la que un jugador corriente tenga acceso) y en segunda instancia se debe de determinar si toda esa información va a ser útil para su aprendizaje. El segundo dilema es el identificar que acciones, para el estado recibido, están permitidas. Es decir que para un determinado estado el ambiente solo aceptará una acción o acciones pertenecientes a un cierto conjunto. Determinar, a base de las reglas, que acciones están disponibles para realizar puede sonar sencillo, pero es una decisión subjetiva, se puede sobre-especificar acciones en pos de modularizar lo más posible el problema para intentar facilitar la implmentación, pero afectando al mismo tiempo el funcionamiento del modelo en consecuencia. También puede generalizar demasiado teniendo como resultado una implementación muy compleja y decisiones ambiguas por parte del agente.

Otro desafío es el riesgo de integrar un BIAS en el agente. Un ejemplo es la *señal de recompensa*3.1.3, esta es crucial para el aprendizaje del Agente Inteligente, el porque a detalle se explicará más adelante, pero en pocas palabras determina a cuales decisiones tomadas

premiar, y en el plantemiento de esta señal un BIAS puede afectar enormemente al comportamiento del agente. El BIAS igualmente puede aparecer: al momento de determinar que información el agente debe recibir, en que acciones puede tomar, en el modelado de la estructura del agente, etc. Eliminar el BIAS por completo es imposible, pero para el desarrollo de un Agente Inteligente exitoso se debe de tratar de erradicarlo, o caso contrario se corre el riesgo de que el agente diverja del funcionamiento que se desea.

1.2. Hipótesis

Existen ciertos trabajos que plantearon una solución para el caso de estudio Monopolio, estos son: *Aprendiendo a jugar Monopolio: Un Acercamiento de Aprendizaje Reforzado* [6], *Toma de decisiones en Monopolio usando un acercamiento de Aprendizaje Reforzado Profundo Híbrido* [7]. Ambas aproximaciones presentan limitantes que no liberan todo el potencial contenido en el juego, los análisis de estas deficiencias se encuentran en las secciones 2.5 2.6 respectivamente. Para implementar una solución completa al problema se debe de plantear un Agente Inteligente que sea capaz de aprender a jugar Monopoly en su totalidad, sin poseer ningún tipo de BIAS y que logre superar a implementaciones anteriores. Para alcanzar esto se debe de identificar maneras de aproximar a los problemas que justamente impidieron una implementación completa en primer lugar. Ya habiendo planteado una solución para los desafíos se puede completar la proposición de implementación de los sectores que no fueron ya mencionados y así presentar el modelo propuesto en su totalidad.

El primer desafío identificado en la Descripción del Problema 1.1 es la integración del comercio. Para aproximar una solución es necesario analizar en qué situaciones el comercio entra en juego en la toma de decisiones de un jugador, dichas situacines son dos:

- Al decidir si realizar una oferta
- Al decidir si aceptar o rechazar la oferta

siendo la segunda situación es una consecuencia de la primera. En otras palabras, el comercio tiene dos caras, el ofertante y el interesado. Para que el agente aprenda a comerciar adecuadamente este debe tratar ambas caras de forma distinta, pues toman consideraciones distintas. Es decir que cuando un jugador se exponga a la primera situación fue por decisión propia y normalmente conlleva un menor riesgo, pues basa la oferta en pos de su conveniencia. Para tomar una decisión adecuada cuando se presenta la segunda situación se debe de tener más aspectos en cuenta, se tiene que determinar si el beneficio presente en dicha oferta

recibida equilibra la balanza con la ventaja que se otrogará al oponente al aceptar su oferta. Es así como se infiere que criterios distintos son necesarios para un razonamiento adecuado en ambos casos. Es por esto que se plantea que el Agente Inteligente base la toma de decisiones para el comercio partiendo de la perspectiva de la acción, en otras palabras dependiendo de a quién impacta la acción, razonar la respuesta de una manera acorde a dicha situación.

La sensibilidad en el entorno no es una buen propescto para un Agente de Aprendizaje Reforzado. Pequeños cambios en el ambiente afectan demasiado a las decisiones tomadas, cosa que puede resultar en acciones con una aparencia aleatoria durante la ejecución. Con el ambiente cambiante de una partida de Monopolio, está claro que si no se trata la sensibilidad del agente a los cambios el resultado siempre será malo, sin importar que se propongan soluciones a los demás problemas. Mayormente el causante de tal sensibilidad es la técnica de aprendizaje. Métodos tradicionales como SARSA o Q-learning [5], mencionado anteriomente, tienden a sobre-estimar los valores de acción. La integración de Redes Neuronales dan paso a modelos de toma de decisión más robustos como DQN (Deep Q-Network), que poseen mayor adaptabilidad y son capaces de manejar ambientes que tienen una cantidad inmensa de estados. Pese a esto, la integración de redes neuronales no soluciona por completo el problema de la sensibilidad a los cambios, el resultado no sería el mejor que se puede obtener. Además, en el aprendizaje del agente también influye a la estrategia utilizada por la técnica de aprendizaje. SARSA usa una estrategia on-policy, lo que significa que su política de toma de decisiones es la misma que dicta su comportamiento, Q-Learning en cambio es una técnica que maneja una estrategia off-policy, lo que significa que su comportamiento y sus decisiones son determinadas por políticas distintas. Para aproximar este importante problema se propone que el agente use la técnica de aprendizaje de Deep Double Q-Learning 2.3, explicado más a detalle en la sección 2.3, que como su nombre indica hace uso de Deep Q-Learning (DQN) pero soluciona el problema de sobre-estimación de valores de acción, disminuyendo así la sensibilidad del agente sobre los pequeños cambios en el ambiente. En este caso igualmente es más conveniente un acercamiento off-policy para controlar independientemente la exploración del ambiente y la explotación del conocimiento adquirido.

La aproximación de una solución al entorno complejo debe dividirse en dos partes:

- Abstracción del estado del entorno a información que el agente pueda procesar
- Determinaciones del conjunto de respuestas validas a dicho estado procesado

Ambas partes se encuentran fuertemente relacionadas, pero requieren de aproximaciones diferentes. Como el Agente Inteligente tomará el rol de jugador en una partida, para afrontar

la primera parte del problema es conveniente identificar primeramente qué información está a disposición de un jugador común. La respuesta es toda la información de los oponentes. Es decir que cada jugador sabe:

- Que propiedades poseen los demás jugadores
- Los detalles de dichas propiedades
- Si un jugador ya cuenta con un Monopolio sobre un color
- De cuantas casas contruidas un jugador es dueño
- Con cuanto dinero cuenta cada jugador

Si bien puede parecer que mientras más información reciba el agente más acertadas serán sus acciones, en muchas ocasiones, entre la información recibida existen datos que son irrelevantes para el objetivo que al que el agente apunta y el tomarlas en cuenta solo realentiza su entrenamiento sin realizar un aporte significativo, las desventajas superan los benficios. Feature Engineering es el campo que se encarga de estudiar la relevancia de los datos, combinaciones entre estos, jugando con las dimensiones de los datos para encontrar su forma óptima con la que el modelo de ML funcione mejor. En este caso se considera que un acercamiento .end to end learning"será más óptimo, es decir se propone que el agente tome en cuenta toda la información recibida, pues se considera que todo jugará un rol importante en su toma de decisiones.

La propuesta de solución para la segunda parte del problema es un poco más compleja, para determinar las acciones disponibles para un estado en específico es necesario estudiar las reglas del juego en busca de las limitantes que indiquen justamente esto. Monopolio por suerte posee de una manera bien establecida dichas limitantes en sus reglas oficiales [8], lo único por hacer es identificar todas las acciones que un jugador pueda tomar durante la partida y clasificarlas en función de las limitantes establecidas. Las clasificaciones son las siguientes:

- Por quién fue iniciada la acción
- A que tipo de propiedades está asociada la acción
- A que fase del juego pertenece la acción

De esta manera dependiendo de que limitantes se encuentren impuestas para una acción dado un estado, se podrá encontrar cuáles son las respuestas posibles para este mismo. Por último, para representar matemáticamente ambas partes de la solución con tal de que se pueda utilizar todo lo descrito anteriormente en una red neuronal, se propone hacer uso de un Proceso de Decisión de Markov 2.2, o MDP por sus siglas en inglés "Markov Decison Process". Un MDP permitirá abstraer la información del ambiente y las acciones decididas por el agente, permitiendo un engranaje entre la interacción del Agente Inteligente con su ambiente.

Para evitar el BIAS y además realizar la optimización de hiperparametros en el proceso, se propone hacer uso del algorítmo genético SEARL[2] 2.4. Las implementaciones anteriores hacen uso de Agente de Política de Acción Fija o FPA, por su nombre en inglés "Fixed Policy Agent", para realizar el entrenamiento y para la probar los resultados, esto no necesariamente es lo ideal, puesto a que lo que el agente inteligente en realidad aprenderá es a superar las estrategias invariables de los FPA. En otras palabras existirá un cierto BIAS hacia la estrategia utilizada en el diseño de los FPA. Una forma de eliminar esto con el algoritmo genético es agregar a la población inicial cierto número de los Agentes Inteligentes y en los restantes cupos de la población a los Agentes de Política de Acción Fija para que estos sirvan como guía inicial en el aprendizaje del Agente Inteligente. Eventualmente se verán superados por estos, desapareciendo y permitiendo así que los Agentes Inteligentes continuen mutando y desarrollandose entre ellos. La optmización de hiperparámetros tiene lugar en las mutaciones. En la etapa de selección los agentes inteligentes con los hiperparámetros más adecuados serán los que se verán victoriosos y avanzarán a la siguiente etapa.

1.3. Justificación

El potencial del aprendizaje reforzado mostrado por Alpha Go en 2016 fue determinante, pero contrario a lo que se puede llegar a entender, no es que se haya descubierto una forma definitiva de plantear un agente de aprendizaje reforzado tal que garantize su éxito en cualquier ambiente, ni siquiera en cualquier juego. La variedad de aplicaciones y por ende entornos en los que se puede aplicar un agente inteligente hace el generalizar su creación una tarea imposible. Es por esto que son necesarias múltiples herramientas, técnicas, métodos, para combinarlos con ideas propias aproximando de esa manera una solución a un problema que con seguridad será distinto en la mayoría de las ocasiones. Esta propiedad del campo del

[2]Aprendizaje Reforzado Automático con Muestreo Eficiente

aprendizaje reforzado hace que las puertas estén siempre abiertas a la innovación, el afrontar problemas nuevos brinda consigo nuevos métodos, nuevas técnicas y nuevas herramientas que serán recibidos con los brazos abiertos.

Ya se mencionó anteriormente sobre la importancia de los juegos en el aprendizaje reforzado como base para la obtención de beneficios, los cuáles pueden ser explícitos o implícitos. Los beneficios explícitos son los que se obtienen de manera directa del resultado de la investigación, como un modelo que sin ningún cambio puede funcionar como solución, o que al menos es el primer paso para un modelo más complejo. Los beneficios implícitos son las ideas aplicadas en el modelo que llevaron a que la solución final sea exitosa, en otras palabras están fuertemente arraigadas en el desarrollo del proyecto y son lo que diferencian la solución de las demás. Basandose en este motivo y en la aletoriedad de los problemas a enfrentar explicado anteriomente se puede concluir que de todo proyecto se obtendrá la solución (beneficio explícito) y al menos una idea innovadora (beneficio implícito). Open AI GYM es un buen ejemplo, juegos como Ant, Lunar-Lander o Cartpole pueden llegar a tener aplicaciones directas en el mundo real, como enseñar a un robot de cuatro patas a pararse, a un dron a aterrizar o a un brazo robótico a equilibrar un objeto. Claramente una extrapolación de cualquiera de este tipo de implementaciones a entornos mas complejos como el mundo real no es tan simple, se cuenta con muchas variables nuevas y complejas a tomar en cuenta, pero no se corre sin antes caminar y el ambiente presentado en un juego puede llegar a ayudar en los primeros pasos del proyecto de una forma u otra. Además, los beneficios implícitos pueden llegar a ser muy importantes en futuros proyectos, es muy probable que DeepMind haya tomado ideas de AlphaGo para reutilizarlas en AlphaFold, mirándolas desde otra perspectiva.

De el beneficio directo (la solución), que en este caso sería el agente de aprendizaje reforzado que aprendió a jugar Monopolio, se puede sacar aplicaciones bastante interesantes. Primeramente se encuentra la extrapolación del agente en juegos similares a Monopolio, como Catán, en los que no sería entrenado de cero, sino que desde el punto que se encuentra se le haría jugar y aprender de las partidas directamente. Esto reduce el tiempo total de entrnaiento por mucho aunque se tendría que adaptar el análisis del espacio de estado y de acción en la implementación del nuevo juego para que funcione adecuadamente con el funcionamiento del agente. Otra opción es extrapolar la idea del modelo del agente a una solución con un ambiente nada parecido a Monopolio, pero que se crea que la solución pueda funcionar óptimamente. Para este caso sí se tendría que replantear todo el estado de espacio y de acción, pero la idea troncal del modelo seguiría estando presente. Se tendría que entrenar

al nuevo planteamiento desde cero, pero ya con un porcentaje considerable de certeza de que funcionará óptimamente.

Existen versiones de Monopolio con reglas modificadas que son planteadas por sociólogos, unos ejemplos son los siguientes trabajos [9], [10], [11]. Estas versiones son usadas para el estudio social de la estratificación e inequidad en las sociedades. Es mayormente utilizado por profesores con objetivos puramente educacionales, el propósito detras de que el juego usado sea Monopolio es para provocar un giro en el pensamiento del estudiante sobre el juego, que con seguridad lo conocieron y jugaron en alguna ocasión. La implementación propuesta puede ser de mucha ayuda al momento de servir como herramienta para los docentes, pues un Agente de Aprendizaje Reforzado puede ser estudiado en su progreso de aprendizaje analizando los datos obtenidos durante su entrenamiento sobre esta versión modificada de Monopolio, para así otorgar una nueva perspectiva tanto a docentes como estudiantes. El algorítmo genético permitirá obtener datos sobre las diferentes generaciones de agentes, el porque de su selección y las mutaciones que se vieron favorecidas en el proceso, además de los cambios en el comportamiento que otorgaron estas. También se puede estudiar el planteamiento de nuevas funciones de recompensa en representación al comportamiento que deseen estudiar, para ver como se desenvuelve este agente modificado en el ambiente.

Otro aspecto en el que esta implementación puede ser de gran ayuda es en encontrar estrategias óptimas para el juego. Una ventaja de utilizar el algoritmo genético es que entre las diferentes generaciones se puede estudair las diferentes estrategias utilizadas por los agentes en la etapa de selección, ver qué características tenía la estrategia usada por el agente ganador y sobre que otra estrategia fue exitosa. También se puede hacer uso de los rewards para estudiar que estrategia es más exitosa.

1.4. Objetivos

Los objetivos de la investigación son los siguientes:

1.4.1. Objetivo General

Aplicando técnicas y conceptos de Aprendizaje Reforzado y Aprendizaje Profundo, esarrollar un Agente Inteligente que sea capaz de actuar como jugador de Monopolio en su totalidad, , con el fin de encontrar un modelo óptimo que sea capaz de superar a implementaciones anteriores.

1.4.2. Objetivos Específicos

- Plantear los estados de espacio y acción ideales para el Agente Inteligente haciendo uso de un Proceso de Decisión de Markov tal que le sea posible tomar el rol de un jugador de Monopolio.
- Diseñar la estructura interna del agente tal que le permita engranar correctamente con los espacios de estado y acción implementando redes neuronales que hagan uso de Double Deep Q-Learning.
- Implementar el juego de Monopolio haciendo uso del lenguaje de programación Python, tomando en consideración el manejo de los espacios de estado y acción, y utilizando conceptos y técnicas de Programación Orientada a Objetos.
- Integrar el Agente Inteligente diseñado en la implementación de Monopolio utilizando conceptos y técnicas de Programación Orientada a Objetos.
- Implementar Agentes de Política de Acción Fija (FPA) para ser usados en el entrenamiento del agente como oponentes iniciales.
- Encontrar la configuración óptima de hiperparámetros que le otorguen el éxito al Agente Inteligente haciendo uso del marco de trabajo SEARL.

Capítulo 2

Marco Teórico

En *Computing Machinery and Intelligence*[1] Turing menciona va a parecer que dado un estado inicial de una máquina y una entrada en concreto siempre va a ser posible predecir todos los futuros estados. Esta es una visión que se asemeja al del demonio de Laplace[1], sin embargo lo que las computadoras hacen actualmente es mejor categorizado como predecir los posibles estados futuros. En el universo descrito para el demonio de Laplace, pequeños errores en las condiciones iniciales pueden tener un gran efecto más adelante en el futuro cambiando el resultado final [1], las computadoras no cuentan con la exactitud con la que cuenta el demonio, sin embargo la precisión de las computadoras actuales nos permite obtener al menos una aproximación ínfima de este poder. El conocimiento detrás de los métodos que sustentan este trabajo representan ese pequeño poder, desarrollado por años de investigación y trabajo en conjunto. Para comprender el sustento de la propuesta es importante explicar la teoría sobre la que se fundamenta.

2.1. Aprendizaje Reforzado

El problema del aprendizaje reforzado esta diseñado para ser una copia directa del problema de aprender mediante la interacción con el propósito de lograr un objetivo. Quien aprenderá y tomará las decisiones es llamado agente. Con lo que interactua el agente, que comprende todo lo que es ageno a este es llamado el ambiente. Estos dos interactuan continuamente, el agente seleccionando acciones y el ambiente respondiendo a tales acciones presentando una nueva situación al agente. Este ambiente también da cabida a las recom-

[1]Demonio imaginado por Laplace, capaz de conocer la posición y la velocidad de todas las partículas del universo

pensas, valores numéricos especiales que el agente trata de maxmizar con el tiempo. Una completa especificación del ambiente, incluyendo cómo las recompensas son determinadas, define una tarea, una instancia del problema del aprendizaje reforzado.[12]

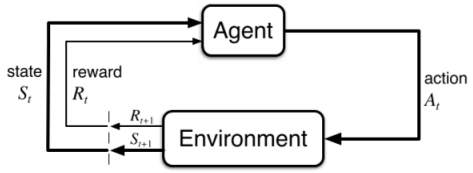

Figura 2.1: Bucle de Interacción entre Agente y Ambiente, Fuente [12]

Si bien ya se aproximó a la definición de Aprendizaje Reforzado en múltiples ocasiones anteriormente, es crucial explicar elementos clave de este modo de Aprendizaje de la Inteligencia Artificial para la mejor comprensión de ciertos temas tratados.

2.1.1. Tratamiento de los datos

Puede ser erroneamente pensado que la mayor diferencia entre los otros tipos de aprendizaje y el aprendizaje reforzado es que este último no trata con un conjunto de datos, sino directamente con el entorno, pero el esquema no es así. Los seres humanos poseen sentidos que le permiten recaudar información sobre su entorno, procesarlo y realizar una acción en función a la información recibida, es el mismo caso con un agente inteligente el cuál debe percibir el ambiente que lo rodea de alguna forma, de esta percepción es de donde se obtiene los datos sobre los cuales el agente deberá basarse para tomar una decisión, de esta decisión se estimará una recompensa para premiar al agente si el resultado fue bueno, o penalizarlo en caso de que el resultado haya sido malo. Como dicta el axioma peripatético *Nada está en el intelecto que no haya estado primero en los sentidos.*

Para dejar las diferencias claras es preferible indicar de manera directa las diferencias entre aprendizajes en su forma base. Empezando con el supervisado, la diferencia con el reforzado es que el supervisado aprende de un set de datos ya determinados como correctos o incorrectos. Cada ejemplo es una descripción de una situación en conjunto con una especificación sobre la acción correcta a tomar, esto es impráctico para los problemas que requieran interacción, en ambientes inexplorados es más conveniente aprender de la experiencia[12]. La estimación sobre la recompensa para un agente de aprendizaje reforzado es cálculada en tiempo real, además de ser más subjetiva, pues no es posible realizar una determinación binaria sobre si una acción fue completamente correcta o completamente errónea, es más

una estimación sobre cuán favorecedora fue la acción, cosa que le permitirá al agente una mejor adaptación a un ambiente inexplorado.

Entre el aprendizaje no supervisado y el reforzado la diferencia es más clara. Si bien ambos tratan con datos no clasificados, el aprendizaje reforzado trata de maximizar la recompensa en lugar de intentar encontrar una estructura oculta entre los datos[12].

2.1.2. Exploración vs Explotación

Otro aspecto importante a tomar en cuenta es la nivelación entre explorar el ambiente por nuevo conocimiento y explotar el conocimiento ya adquirido. Para maximizar las recompensas, un agente de aprendizaje reforzado debe preferir acciones que ya experimentó en el pasado y resultaron favorecedoras, pero para descubrir tales acciones el agente debe de probar acciones que no experimentó con anterioridad [12]. Cumplir con ambos requerimientos al mismo tiempo es imposible, por lo que el agente tiene que experimentar con una variedad de acciones al principio y progresivamente favorecer aquellas que aparenten ser las mejores. Para mayor seguridad es preferible que cada acción sea probada muchas veces para así asegurarse de cuál es la recompensa esperada para cada una de estas.

2.1.3. Política de toma de decisión

Un *policy* (política de toma de decisión) define la forma en la que le agente se comporta en un momento dado[12]. En otras palabras se puede decir que un *policy* es la estrategia del agente para afrontar un problema, haciendo una analogía a la inteligencia para seleccionar y perfeccionar esta estrategia en base a las metas que se le plantee al agente para cumplir. El *policy* es el núcleo del agente de aprendizaje reforzado en el sentido de que el policy solo es capaz de determinar el comportamiento[12], es por esto que el completo proposito del aprendizaje es encontrar el policy óptimo para el ambiente con el que se está tratando.

2.1.4. Señal de recompensa

La señal de recompensa define la meta del problema de reinforcement learning[12]. Con cada paso o cambio de estado que el ambiente sufre a causa de una acción del agente, este recibirá una señal que será alta para premiar su acción o baja para penalizarlo. Este aspecto influye directamente en la política de toma de decisiones, como se plantee la meta o cómo se estimará la señal de recompensa es lo que determinará como deberá ser la política óptima tal que esta señal de recompensa sea maximizada, es una dependencia mutua. Esta señal de

recompensa es estimada por una función que toma en cuenta el estado resultante de la acción del agente. Este solo puede alterar el resultado de la función de forma indirecta, es decir que únicamente mediante las acciones que tome, lo estimado de la función será diferente. El agente no debe ser capaz de cambiar la función que calcula la señal de recompensa pues esto implicaría una desviación en las metas planteadas.

2.1.5. Función de valor

La función de valor indica que es bueno a largo plazo, totalmente opuesto a la señal que recompensa que indica de manera inmediata si la acción fue buena o no. A grandes razgos, la función de valor de un estado es la recompensa total que el agente puede esperar acumular sobre el futuro empezando de este estado[12]. La importancia de esta función recae en otorgar una visión más amplia sobre las acciones a tomar para el agente, puesto a que una decisión puede tener bajo valor de recompensa inmediata, pero las posibles decisión consecuentes a esta pueden llegar a brindar mucho más recompensa.

Anteriormente se mencionó que el policy era el núcleo del agente de aprendizaje reforzado, pues la función de valor cumple una función que igualmente muy importante debido a que son los valores obtenidos de esta función los que dictan cuáles son las accciones óptimas justamente por el punto anetrior. Las recompensas son en un sentido primarias, donde las predicciones de estas recompensas (la función de valor) son secundarios, sin recompensas no habría valores a calcular para la función, y el único propósito de estimar valores es el obtener más recompensa. Aún así es el componente más importante de los algoritmos de aprendizaje reforzado es lograr obtener una función de valor que estime los valores eficientemente.

2.2. Proceso de Decisión de Markov

Un Proceso de Decision de Markov es un framework común, muy importante para el modelado de problemas de optimización dinámica. Son descritos por un set de estados S, un set de acciones A_s para cada estado $s \in S$, un kernel de transición $\mathbf{P}$ que marca las probabilidades de transición $\mathbf{P}_{sa} \in R_+^{|S|}$ para todo par estado-acción (s, a), una recompensa r_{sa} para cada par estado-acción (s, a) y un factor de descuento $\lambda \in (0, 1)$.

Una política π direcciona la historia de un estado-acción hasta el tiempo t $(s_0, a_0; s_1, a_1, ..., s_t)$ hacia una probabilidad de distribución sobre el set de acciones A_{s_t} para cada periodo $t \in N$. Una política es solo llamada *Markoviana* si depende únicamente de el estado actual y es llamada *Estacionaria* si no depende del tiempo[13].

Ya con lo descrito anteriormente se puede notar la similitud entre la definición del aprendizaje reforzado y la del Proceso de Decisión de Markov, esto es de esperar debido a que el problema del aprendizaje reforzado es dinámico, es decir que una representación *Markoviana* encaja perfectamente con el planteamiento de cualquier problema de aprendizaje reforzado. El rol que cumple el framework es el de otorgar una representación matemática a problemas en los que la toma de decisiones es dinámica. Es abstracto, flexible y puede ser aplicado a muchos problemas diferentes en múltiples formas diferentes.

La regla general a seguir al momento de la abstracción del problema a un MDP (Markov Decision Process) es que todo lo que no pueda ser cambiado arbitrariamente por el agente es considerado ageno a el, y por tanto parte de su ambiente, igualmente no se debe asumir que todo en el ambiente es desconocido por el agente [12]. Es importante determinar correctamente lo perteneciente al ambiente pues de esta es de donde se obtienen el set de estados S, llamado también *espacio de estados* y el set de acciones A, igualmente conocido como *espacio de acción*. Si el espacio de estado y de acción son finitos, entonces es llamado *Proceso de Decision de Markov finito (finite MDP)*.

Para abstraer un problema como un MDP primeramente se debe determinar si cumple con la propiedad de Markov, es decir que una señal de estado debe triunfar en retener toda la informació relevante para afirmar que posee la propiedad de Markov[12]. Matemáticamente esto se representar diciendo que la probabilidad de distribución 2.1 para todo r, s' y todos los posibles valores de los eventos pasados deben se iguales a $p(s', r|S_t, A_t)$, es decir que la respuesta del ambiente en $t+1$ depende únicamente del estado y acción en el tiempo t, como lo muestra la ecuación 2.2.

$$Pr\{S_{t+1} = s', R_{t+1} = r|S_0, A_0, R_1, ...S_{t-1}, A_{t-1}, R_t; S_t, A_t\} \tag{2.1}$$

$$p(s', r|s, a) = PR\{S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\} \tag{2.2}$$

Monopoly puede ser representado como un finite MDP, pese a que su espacio de estados es demasiado numeroso, sigue siendo teniendo un fin. Un finite MDP es importante para la teoría del aprendizaje reforzado, en particular es definido por el set de estados, el set de acciones y por la dinámica discreta de su ambiente. Con la ecuación 2.2 se pueden obtener las probabilidades para el próximo estado s', r y de esta misma ecuación derivan otras que

describen todo lo demás del ambiente, como las recompensas esperadas

$$r(s,a) = E[R_{t+1}|S_t = s, A_t = a] = \sum_{r\in R} r \sum_{s\in S} p(s', r|s, a) \tag{2.3}$$

o las probabilidades de transición a un estado

$$p(s'|s,a) = Pr\{S_{t+1} = s'|S_t = s, A_t = a\} = \sum_{r\in R} p(s', r|s, a) \tag{2.4}$$

2.3. Deep Double Q-Learning

Q-Learning es un algoritmo que provee al agente con la capacidad de aprender a actuar óptimamente en dominios Markovianos mediante la experiencia de las concecuencias de sus acciones[5]. Q-Learning es una forma primitiva de aprendizaje , pero puede operar como las base de técnicas más sofisticadas. La tarea que en enfrenta el agente que utilice Q-Learning es el de determinar un policy π^* sin saber inicialmente los valores de los rewards esperados ni de las probabilidades de tansición, es decir que no se tiene información para determinar los valores Q que determinan cuál es la mejor acción a tomar. Para determinar los valroes Q de un política π se usa la ecuación

$$Q^\pi(x,a) = R_x(a) + \gamma \sum_y P_{xy}[\pi(x)]V^*(y) \tag{2.5}$$

donde V^π de un valor dado es representado por la ecuación

$$V^*(x) = V^{\pi^*}(x) = \max_a \{R_x(a) + \gamma \sum_y P_{xy}[a]V^{\pi^*}(y)\} \tag{2.6}$$

de donde deriva

$$V^*(x) = \max_a Q^*(x,a) \tag{2.7}$$

entonces en conclusión se determina que

$$Q_n(x,a) = (1-\alpha)Q_{n-1}(x,a) + \alpha_n[r_n + \gamma V_{n-1}(y_n)] \tag{2.8}$$

donde para el episodio n

- x es el estado actual

- a_n es la acción seleccionada
- y_n es el estado subsecuente
- r_n es la recompensa adquirida
- α_n es el fator de aprendizaje

El valor Q es la recompensa descontada esperada por ejecutar la acción a en el estado x y siguiendo la política π. La condición más importante que el teorema de demostración de la convergencia de Q-Learning a una política óptima indica es que la secuencia de episodios que forma la base del aprendizaje debe de incluir un número infinito de episodios para converger al punto mínimo. Q-Learning es uno de los algoritmo de aprendizaje reforzado más populares, pero es conocido por aprender en ocasiones valores de acción irrealmente altos debido a que incluye un paso de maximización sobre los estimados valores de acción, lo que le da una tendecia a preferir valores sobre-estimados[14].

Deep Q-Learning (DQN) combina Q-Learning con una red neuronal profunda, la cuál le brinda una función de aproximación más flexible con el potencial de una baja aproximación de error asintótico, y le determinismo de los ambientes previene los effectos nocivos del ruido[14]. Pese a esto DQN aún tiende a sobre-estimar los valores de acción substancialmente. Un DQN, al ser una red neuronal multicapa, dado un estado s da como salida un vector que representa los valores de acción $Q(s;\theta)$ donde θ representa los parámetros de la red. Hay don ingredientes importantes para un DQN, una red *target* y otra *online*, la idea es que ambas inicien siendo iguales y que cada τ pasos los valores de la red *online* sean copiados a la red *target*, es decir $\theta_t^- = \theta_t$, a esto se llama *experience replay*. Entonces la ecuación usado para la red *target* es

$$Y_t^{DQN} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_t^-) \tag{2.9}$$

El operador *max* para Q-Learning normal o DQN usa el mismo valor para seleccionar y evaluar una acción, esto aumenta la probabilidad de sobre-estimar valores. Dobule Q-Learning soluciona esto desacoplando la selección de la evaluación en dos diferentes redes obteniendo θ y $\theta^{'}$, las experiencias se asignan a una de forma aleatoria. Es de esta manera que Double Q-Learning consigue mejorar considreablemente el problema de la sobre-estimación. Sus ecuaciones serían las siguientes

$$Y_t^Q = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta_t); \theta_t) \tag{2.10}$$

para calcular el valor Q,

$$Y_t^{DoubleQ} = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta_t); \theta_t') \tag{2.11}$$

para calcular el error.

Lo propuesto para usar en este proyecto es Double Deep Q-Learning, cuya ecuación es la siguiente

$$Y_t^{DoubleQ} = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta_t), \theta_t^-) \tag{2.12}$$

hace uso de la misma ecuación para DQN 2.9 debido a que su arquitectura provee un cadidato natural para la segunda función de valor sin la necesidad de adicionar una red nueva. Debido a esto la ecuación propone la elección de la acción usando la red *online* y la estimación del valor haciendo uso de la red *target*.

2.4. SEARL

Los algoritmos genéticos son un tipo de algoritmo de optimización, es decir son usados para encontrar soluciones óptimas a un determinado problema computacional que maximice o minimice una función particular. Representan una rama de un campo de estudio llamado computación evolutiva, en la que imitan procesos biológicos como la reproducción y la selección natural para encontrar la solución mas apta [15]. Al basarse en la evolución, los procesos de los algoritmos genéticos son aleatorios, aunque permite tener un nivel de aleatoriedad y otro de control, debido a esto los algoritmos genéticos son mucho más poderosos y eficientes que los de busqueda exhaustiva y no requieren de información extra sobre el problema.

Sus componentes son los siguientes:

- Una función de ajuste (*fitness function*) que es la que el aogritmo trata de optimizar.
- Una población
- Un proceso de selección
- Un proceso para producir a la siguiente generación (*crossover*)
- La mutación aleatoria de los miembros de la población

Sample Efficient Automated Reinforcement Learning o *SEARL* es un framework de meta-optimización basado en un algoritmo evolutivo o genético que actúa en los hiperparamtros y el entrenamiento basado en gradiente, haciendo uso de un *experience replay* compartido [16], todo para agentes *off-policy*. Este framework está diseñado para afrontar tres problemas:

- Eficiente optimmización de los hiperparametros
- Configuración dinámica
- Arquitectura dinámica de la red neuronal

SEARL preseva los beneficios de la configuración dinámica para habilitar la optimización de los hiperparámetros y descubrir una agenda apropiada en lugar de una confiuración estática [16]. Las redes neuronales evolutivas, que se plantean, preservan parámetros ya entrenados a la vez que adaptan sus arquitecturas. En SEARL cada individuo de la población representa un agente de aprendizaje reforzado profundo, formado por su política, su red neuronal de valor y los hyperparámetros de entrenamiento entre los que se incluye la arquitectura de la red neuronal. El entrenamiento y meta-optimización de los individuos toma lugar en bucle que consite de cinco fases:

- **Inicialización:** Inicializar la población representada como:

  $pop_g = (\{A_1, \theta_1\}_g, \{A_2, \theta_2\}_g, ..., \{A_N, \theta_N\}_g)$

  donde θ representa la configuración de los hiperparámetros.

- **Evalaución:** Despues de la Inicialización y de cada fase de Entrenamiento, se evalua a cada individuo de la población por al menos un episodio o un número mínimo de pasos en el ambiente. La función fitness, definida en esta fase, es la media de las recompensas obtenidas por el agente.

- **Selección:** Se realiza la selección mediante un torneo, el ganador reserva un puesto en la siguiente generación. El individuo con el valor de *fitness* más elevado de la fase anterior es pre-seleccionado para la siguiente generación.

- **Mutación:** Como su nombre indica, muta los hiperparámetros de cada individuo en busca de la configuración óptima.

- **Entrenamiento:** Se obtiene un porcentaje de los datos guardados en la memoria compartida *experience replay* y son usados para entrenar al agente.

Una mejor representación sobre el flujo de funcionamiento de SEARL puede ser observado en la figura 2.2.

Este framework tambié abre las puertas a experimentar con distintos tipos de agente, y estudiar los resultados finales, los datos de sus mutaciones y comportamientos a lo largo de la ejecución del algoritmo genético. SEARL es un poderoso y flexible framework, incluso capaz de poder usaurse en agentes *on-policy*.

2.5. Aprendiendo a Jugar Monopolio: Un Acercamiento de Aperndizaje Reforzado

Este es uno de los trabajo previos sobre el tema que se trata en este proyecto, pero posee una implementación limitada debido a que no toma en cuenta el comercio en su planteamiento, aspecto que es clave dentro del desarrollo del juego. Se plantea el estado de espacio como un vector tri-dimensional de objetos que contienen información sobre el *áream posiciónm y finanzas* del actual estado del juego en el tiempo t [6].

- El objeto *área* contiene información sobre las propiedades del juego, es decir las propiedades que posee el jgador y las de sus oponentes en el tiempo t. Es una matriz $10 * 2$ donde la primera columna representa las propiedades del agente, la segunda el resto de sus oponentes. Cada fila corresponde a un grupo de propeidades, las cuáles están clasificadas por color si son avenidas, los ferrocariles y las utilidades. Cada elemnto de la matriz es una tupla (x, y) que especifica el porcentaje que el jugador y posee sobre el grupo x.

- La variable posición determina la posición actual del jugador en la tabla en relación con le grupo de color, es decir, que si se encuentra en el grupo 4 el valor de posición es 0,4.

- El vector de finanzas consiste de 2 valores, el número de propiedades del jugador en comparación con sus oponentes, al igual que el cálculo de su cantidad de dinero poseido.

Al inicio de la descripción del espacio de estado el autor del trabajo [6] indica que se le suministrará al agente toda la información disponible para un jugador, cosa que es erronea. Con el objeto *área* del espacio de estado se indica que lo que contendra la matriz será en realidad representaciones de los grupos de propiedades, del jugador y del resto de sus

oponentes, es decir que no hay una descripción específica para sobre que propiedades posee específicamente cada jugador. Igualmente en el objeto *posición* se omite las casillas que no pertenezcan a un grupo de propiedades, con esto se demuestra que en realidad lo que recibe el agente no es el panorama completo y esto puede llegar a afectar severamente el aprendizaje del agente, pues la información omitida es fundamental para una estrategia ganadora.

El espacio de acción está dividido en solo tres:

- *Spend*: Toda posible acción que haga disminuir la cantidad de dinero de un jugador.
- *Sell*: Toda posible acción que permita al jugador ganar dinero.
- *Do Nothing*: No tomar ninguna acción para un grupo específico de propiedades.

Bajo este planteamiento de espacio de acción, el integrar el comercio sería muy comlicado. La clasificación de una acción de comercio resultaría en ambiguedad si se usa el planteamiento de este trabajo, pues el comercio puede ser clasificado tanto como una acción *Spend* o *Sell*. Además esta sobre-simplificación de las decisiones posibles es perjudicial para el entrenamiento del agente. Al revisar el programa, se puede comprobar que el agente planteado por Bailis [6] hace uso de únicamente una red neuronal y fue diseñado con una única salida, 1 para *Spend*, 0 para *Do Nothing* y -1 para *Sell*. La razón detras de que un espacio de acción tan pequeño puede se perjudicial para el agente es que a base de la misma entrada el agente tiene que inferir diferentes salidas, como ejemplo se puede poner las acciones mejorar o deshipotecar una propiedad, siguiendo las normas del espacio de estado planteado anteriomente, es claro que si se trata de la misma propiedad, la entrada es igual para ambos casos, los cuáles se encuentran generalizados dentro del mismo grupo de acción, *Spend*. El programa puede saber que respuesta corresponde a cada acción, pero el agente no es capaz de identificar que recompensa pertenece a cada acción, pues por la información que el posee, que son únicamente sus hiperparámetros y la entrada, ambos casos son iguales lo que puede desembocar en una situación en la que puede resultar premiado y penalizado por tomar la misma decisión, pese a la equidad de los casos desde su punto de vista.

Lo último por tratar es la función de recompensa, la cuál es representada de la siguiente manera:

$$r = \frac{\frac{v}{p} * c}{1 + |\frac{v}{p} * c|} + \frac{1}{p} * m \tag{2.13}$$

Lo que cada elemento representa es lo siguiente:

- p es el número de jugadores.
- c es un valor de suavizado.
- v es la cantidad que representa el valor total de las pertenencias del jugador y es calculado sumando todas las propiedades en poseción del jugador menos la cantidad de todos sus oponentes.
- m es la cantidad que representa las finanzas del jugador y es igual al porcentaje del dinero del jugador sobre la de sus oponentes.

El planteamiento de la función de recompensa no es óptima debido a que en la mayoría de las ocasiones va a calcular un valor negativo, penalizando al agente sin motivo aparente. Esto es debido a el valor de v, pues a menos de que en cada partida el agente sea dueño de al menos 14 propiedades (son 28 propiedades en total) continuará recibiendo recompensas negativas. Además no premia al agente cuando este completa un color o construye una casa.

2.6. Toma de decisiones en Monopolio usando un acercamiento de Aprendizaje Reforzado Profundo Híbrido

Este trabajo previo si presenta una mejor aproximación al problema. Presenta un agente híbrido, es decir el agente toma ciertas decisiones haciendo uso de una política resultante del entrenamiento y otras decisiones a partir de políticas fijas (Fixed Policy). El argumento planteado como sustento para esto es que algunas acciones ocurren mucho má rfecuentemente que otras, resultando en una distribución desbalanceada (*skewed distribution*), entonces es más conveniente usar los *fixed policy* para decisiones poco comunes pero simples y aprendizaje reforzado profundo para las decisiones frecuentes y complejas. En este caso, se usa un *fixed policy* para las acciones de *buy property* y *accept trade offer*, una descripción más detallada de cada acción puede verse en la tabla 2.1. El problema de las *skewed actions* se resuelve en el agente planteado en la hipótesis con el algorítmo SEARL, en la memoria compartida se pueden almacenar igualmente datos sobre los comercios realizados por toda la pobalción, lo que permite realizar un entrenamiento óptimo en la fase *Training*.

La red usada en el trabajo [7] es igualmente una Double Deep Q-Network, misma propuesta en la hipótesis, la diferencia recae en la rigidez de la estructura y configuación, la

propuesta a usar en la hipótesis vería cambios incluso en su arquitectura, lo que puede resultar en una red neuronal mucho más óptima.

El estado de espacio que plantea el trabajo [7] es una combinación de la representación de las propiedades y jugador. Para la representación del jugador se considera

- La posición actual
- La cantidad de dinero
- Un indicador que denote si el jugador está en la carcel
- Un indicador que denote si el jugador posee la carta de salir de la carcel gratis

Para la representación de las propiedades se inluye las posiciones dentro del tablero de las 28 disponibles, Estas contienen 22 avenidas, cuatro ferrocarriles y dos propiedades de utilidad. Esta representación consiste de

- Una representación de dueño que es un vector de cuatro dimensiones, un índice para cada jugador, el banco es representado con puros ceros.
- Un indicador que denote si la propiedad es parte de un Monopolio
- Un indicar que denote si la propiedad fue hipotecada
- La fracción de las casas construidas sobre el número de máximas permitidas

En total el espacio de estado es un vector $v \in R^{240}$, con 16 dimesiones para las representaciones de los jugadores y las restantes 224 para las representaciones de las propiedades. Este planteamiento si presenta toda la información disponible para un jugador al agente, a diferencia del anterior trabajo [6].

Para el espacio de acción el autor de [7] no toma en cuenta acciones compulsivas del jugador, como el tener que pagar renta o avanzar en el tablero. El resto de acciones están clasificados entre tres grupos

- Asociadas a las 28 propiedades
- Asociadas a los grupos de colores
- No está asociada a ninguna propiedad

Action type	Associated Properties	Game Phase	Dimensions
Make Trade Offer (Exchange)	All	Pre-roll, out-of-turn	2268
Make Trade Offer (Sell)	All	Pre-roll, out-of-turn	252
Make Trade Offer (Buy)	All	Pre-roll, out-of-turn	252
Improve Property	Color-group	Pre-roll, out-of-turn	44
Sell House or Hotel	Color-group	Pre-roll, post-roll, out-of-turn	44
Sell Property	All	Pre-roll, post-roll, out-of-turn	28
Mortgage Property	All	Pre-roll, post-roll, out-of-turn	28
Free Mortgage	All	Pre-roll, post-roll, out-of-turn	28
Skip Turn	None	Pre-roll, post-roll, out-of-turn	1
Conclude Actions	None	Pre-roll, post-roll, out-of-turn	1
Use get out of jail card	None	Pre-roll	1
Pay Jail Fine	None	Pre-roll	1
Accept Trade Offer	None	Pre-roll, out-of-turn	1
Buy Property	All	Post-roll	1

Tabla 2.1: Tabla de acciones posibles por un jugador en Monopoly descritas por el trabajo [7]

La siguiente tabla muestra la clasificación

Existe ciertas incoherencias dentro de la tabla, pues en el trabajo se indica que se siguen las reglas oficiales marcadas por hasbro, en estas no indica que un jugador pueda saltar de turno, también la acción *Sell Property* en realidad estaría contenida dentro del comercio. También en el trabajo se simplifica la operación de comercio en tres acciones diferentes

- *Exchange* es para intercambiar propiedades y dinero entre ambos jugadores participes del comercio, la salida representa 28 propiedades para el oponente $\times$ 27 propiedades para el jugador $\times$ 3 métricas de dinero, dando el resultado de 2268.
- *Sell* usado para intercambiar una propiedad por dinero, su salida representa 3 jugadores $\times$ 28 para la representación de propiedades $\times$ 3 métricas de dinero.
- *Buy* se usa para intercambiar dinero por una propiedad, su salida es similar al anterior punto, 3 jugadores $\times$ 28 para la representación de propiedades $\times$ 3 métricas de dinero.

La integración del comercio al juego puede ser mucho más corta, aunque aumenta un poco la complejidad, permite simplificar el estado de acción y el funcionamiento del juego de Monopoly, por lo que no es una forma óptima de afrontar el problema del comercio.

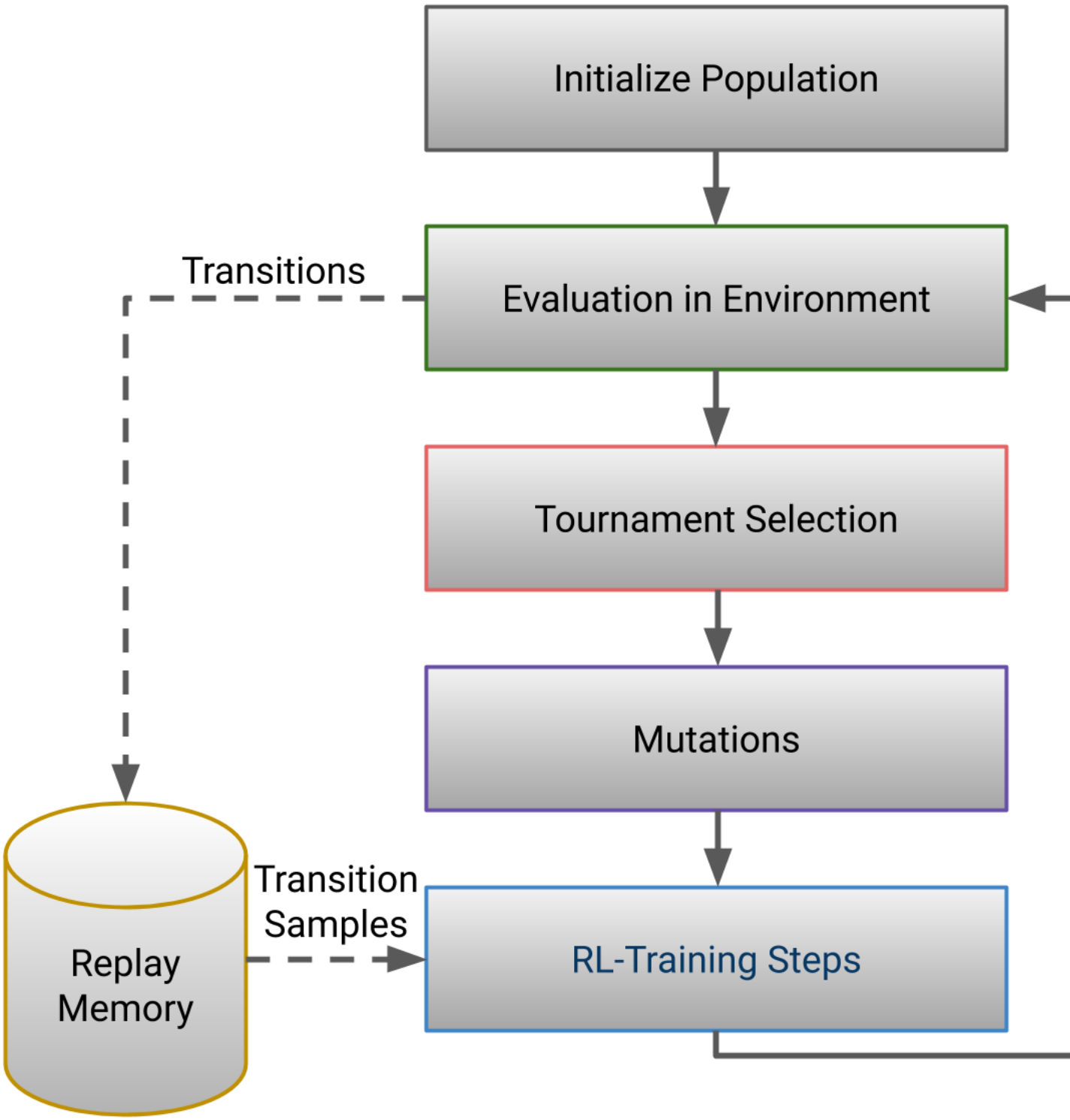

Figura 2.2: Algorítmo SEARL

Capítulo 3

Metodología

En su artículo *Hacia una Metodología de Investigación para la IA: Tres Casos de Estudio en Evaluación* [17] Paul R. Cohen y Adele E.Howe mencionan una pregunta importante que uno debe realizarse al momento de diseñar lo que ellos llaman el método, o dicho en otras palabras los algoritmos o procesos necesarios para alcanzar la tarea obtenida al analizar el tópico; dicha pregunta era la siguiente: ¿cuál es alcance que se desea tener con el método a desarrollar?. Para este proyecto tal cuestión es contestada en la definición del objetivo general 1.4.1 donde se describe que se desea que el agente inteligente debe ser capaz de actuar como jugador de Monopoly en su totalidad. Pero por cuestiones de conveniencia, en el diseño del Agente Inteligente, se determina que el número máximo de jugadores en una partida son 4.

Al ser un campo relativamente reciente de nacimiento, la Inteligencia Artificial no cuenta aún con una metodología global de investigación definida que se pueda usar para alcanzar tal objetivo. Si bien sigue los pasos de las demás ciencias en la metodología general de investigación, debido a sus características propias en ocasiones si se hace evidente la falta de estandarización. Paul R. Cohen y Adele E.Howe, en su artículo [17] de 1889, hacen un primer acercamiento hacia la resolución de este problema, planteando una metodología sencilla para la Inteligencia Artificial. Esta fue la que se siguió en este proyecto.

Cabe resaltar que los pasos indicados en el artículo hacen referencia a todo el desarrollo del proyecto, sin especificaciones sobre lo que debe llevar el documento de investigación. Sin embargo hay algo que si se deja bastante claro, y es que lo importante para la comunidad científica es el método usado [17], el valor científico que tiene por aportar el proyecto. A causa de esto se determinó que el desarrollo de los objetivos, desde el punto de vista de componentes del proyecto, son los que deben ser explicados a profundidad en esta sección.

3.1. Espacios de Estado y Acción

El diseño de los espacios de Estado y Acción es el primer paso en el proceso de creación de un Agente Inteligente. El set de Estados S y el set de Acciones A_s para cada estado $s \in S$ son elementos propios de un Proceso de Decisión de Markov 2.2, y son la representación matemática para los espacios de Estado y Acción respectivamente. La importancia de esta representación recae en la capacidad de abstraer la información tal que pueda ser usada por los modelos estadísticos de Inteligencia Artificial.

La figura 2.1, mostrada en la sección de aprendizaje reforzado, representa el proceso de interacción entre el agente y el ambiente. Algo que quizá no se muestre en la figura es que el ambiente es quién siempre inicia la interacción en bucle. Tal iniciación no siempre es de una manera activa, es decir hay situaciones en las que el ambiente exige activamente una respuesta del agente y otras situaciones en las que el ambiente es pasivo e indiferente a si el agente decide realizar una acción o no, tal indiferencia no indica que no sea vea afectado por las acciones del agente, sino que no espera a que el agente tome una decisión. De este análisis se puede determinar que la información del ambiente, además de ser la causa principal de la toma de decisiones, es información que siempre estará presente, y lo que en realidad variará es la información determinante en la toma de decisiones. Debido a esta razón se determinó que lo ideal es primeramente diseñar el Espacio de Acción para así clarificar cuál es la información determinante en base a la toma de decisiones del agente y formar el Espacio de Estado en consecuencia.

3.1.1. Diseño del Espacio de Acción

Como se mencinó al inicio de la sección, el Agente Inteligente debe cumplir el rol de un jugador, por lo que de manera sencilla se puede inferir que sus posibles acciones deben ser las mismas que las de un jugador regular de Monopolio. Algo importante que se debe tomar en cuenta es que un jugador no tiene el poder de decisión sobre todas las acciones posibles dentro del juego, existen acciones obligatorias. En el segundo trabajo analizado, *Toma de decisiones en Monopolio usando un acercamiento de Aprendizaje Reforzado Profundo Híbrido* 2.6, se califica a tales acciones como impulsivas. Se considera que este término es incorrecto debido a que la impulsividad es una característica del comportamiento, el cuál en el agente está definido por la política de toma de decisión 2.1.3, que a su vez depende de el conjunto de acciones válidas para un estado dado, sobre las que el agente pueda decidir, y las acciones obligatorias no entran dentro de este conjunto. Algunas de las acciones obligatorias

en Monopolio son:

- Lanzar los dados
- Pagar la renta
- Pagar los impuestos
- Avanzar en el tablero, cuando no se encuentre en la cárcel
- Sacar una carta de Comunidad o de Chance
- Cobrar la renta
- Cobrar el salario al pasar por la casilla Go

Debido a lo anterior, cabe recalcar que las acciones que pertenecen al espacio de Acción únicamente son aquellas sobre las que se posea el poder de la decisión. En el trabajo *Toma de decisiones en Monopolio usando un acercamiento de Aprendizaje Reforzado Profundo Híbrido* [7] se toma un buen acercamiento inicial; su planteamiento del espacio de Acción está presentado en la tabla 2.1, pero como se plantea en el análisis del trabajo realizado en la sección 2.6, este sufre limitaciones al no integrar de manera completa el comercio y al sobre-especificar ciertas acciones que en realidad son implíctias . Debido a esto, despues de analizar cuidadosamente las reglas [8], se plantea que el espacio de Acción óptimo es el presentado en la tabla 3.1

Tipo de Acción	Iniciada por	Fase del Juego
Realizar Oferta	Jugador	Pre-turno, Fuera-de-turno
Continuar en Subasta	Banco	Subasta
Construir casa	Jugador	Pre-turno, Fuera-de-turno
Vender casa	Jugador	Pre-turno, Post-turno, Fuera-de-turno
Hipotecar Propiedad	Jugador	Pre-turno, Post-turno, Fuera-de-turno
Pagar Hipoteca	Jugador	Pre-turno, Post-turno, Fuera-de-turno
Concluir Acciones	Jugador	Pre-turno, Post-turno, Fuera-de-turno
(Carta) Salir de la Cárcel	Jugador	Pre-turno
Pagar Fianza	Jugador	Pre-turno
Aceptar Oferta	Banco, Oponente	Pre-turno, Fuera-de-turno
Comprar Propiedad	Jugador	Post-turno

Tabla 3.1: Tabla del espacio de Acción propuesto para el agente

Se resaltan tres cambios realizados a la tabla inicial 2.1. Primeramente las tres acciones relacionadas con el comercio en la primera tabla se unieron a una sola en la tabla propuesta 3.1, esto reduce la complejidad de la acción en sí y permite una representación matemática más simple la cuál se explicará más adelante. Segundo, se quitaron acciones que no estaban permitidas en las reglas [8], como la de salto de turno o la de vender la propiedad, simplificando aún más el espacio de Acción. Igualmente se agregó una nueva acción *Continuar en Subasta*. Tercero, se implementó una nueva clasificación de acción, la columna *Iniciada por* que representa a la entidad que empezó con el proceso de la acción. Pese a que la mayoría de las estas son iniciadas por el jugador, es importante tener este campo debido a que para las acciones que no son iniciadas por el jugador toman un contexto distinto a las demás; las acciones iniciadas por el jugador tienen un contexto propio (no dependen de las decisiones de otro ente solo del estado del juego), en cambio el contexto de las acciones iniciadas por otro ente involucra tanto a quién inició la acción(ente) como a quién tiene que responder a ella (jugador). Como ejemplo, para aceptar una oferta, al jugador no le servirá contar con información sobre si los demás jugadores ajenos al proceso están en la cárcel, sino únicamente le interesará el involucrado.

Para la representación matemática del espacio de acción propuesto 3.1 se toma en cuenta con que tipo de propiedad esta relacionado la salida de cada acción, es decir que tipo de propiedad se debe representar. Para el caso de las acciones relacionadas únicamente con un grupo de color la representación sería un vector $v \in R^{22}$ debido a que son únicamente 22 propiedades que son parte del conjunto de grupo de color. La representación para las acciones que toman en cuenta todas las propiedades sería similar, solo que las dimensiones pasarían a ser 28 ya que se toma en cuenta los cuatro Ferrocarriles y las dos Utilidades, el vector resultante sería $v \in R^{28}$. Para el resto, se da el caso de ser acciones cuya respuesta wes binaria y no están relacionadas con ninguna propiedad, por lo que su representación sería: $[0, 1]$. Las únicas acciones compuestas son: *Realizar una oferta* y *Continuar en Subasta*, para estos casos resulta más conveniente describrir sus estructura con una tabla.

Campo	Dimensiones
Propiedades ofrecidas	28
Propiedades deseadas	28
Dinero ofrecido	1
Dinero deseado	1
Jugador Objetivo	3

Tabla 3.2: Representación matemática de la acción Realizar una Oferta

El jugador objetivo es de 3 dimensiones y no cuatro debido a que el agente, que es un jugador, tendrá como máximo tres oponentes. *Continuar en Subasta* tendría una estructura más simple.

Campo	Dimensiones
Continuar	1
Cantidad a ofertar	1

Tabla 3.3: Representación matemática de la acción Continuar en Subasta

Incluyendo la clasificación de las propiedades relacionadas, la tabla de representación matemática es la siguiente.

Tipo de Acción	Propiedades asociadas	Dimensiones
Realizar Oferta	Todas	61
Continuar en Subasta	Ninguna	2
Construir casa	Grupo-de-color	22
Vender casa	Grupo-de-color	22
Hipotecar Propiedad	Todas	28
Pagar Hipoteca	Todas	28
Concluir Acciones	Ninguna	1
(Carta) Salir de la Cárcel	Ninguna	1
Pagar Fianza	Ninguna	1
Aceptar Oferta	Ninguna	1
Comprar Propiedad	Ninguna	1

Tabla 3.4: Tabla del espacio de Acción propuesto para el agente

El vector del Espacio de Acción Completo sería $v \in R^{168}$, considerablemente menor al Espacio de Acción propuesto en *Toma de decisiones en Monopolio usando un acercamiento de Aprendizaje Reforzado Profundo Híbrido* en el que se presenta el vector como $v \in R^{2922}$.

3.1.2. Diseño del Espacio de Estado

Como el Agente Inteligente va a tomar el lugar de un jugador, este debe tener acceso a la información común que un jugador regular de Monopolio tiene acceso. Si se analizan las reglas [8] se puede observar que en Monopolio, exceptuando el orden de las cartas de Comunidad y de Chance, no hay información oculta a los jugadores. Cada participante de una partida puede observar que propiedades están compradas y cuales no, la cantidad de casas que se

tiene construidas en una propiedad en concreto, el precio y renta, quién es el dueño, incluso cuanto dinero posee este, que otras propiedades posee, etc. En sí, la información a la que un jugador regular de Monopolio tiene acceso es el estado global de la partida, para ser más exactos dicha información esta compuesta por instancias múltipes de dos componetes principales, las características de un jugador y las características de una propiedad.

Empezando por las características de una propiedad, existen dos tipos de propiedades en Monopolio, en un lado están las Avenidas y en el otro el conjunto de Ferrocarriles y l Utilidades. No existe en sí algo que las diferencie, en realidad las Avenidas cuentan con más carácterísticas distintivas además de compartir todas las que poseen los Ferrocarriles y Utilidades. Generalizando, la información de una propiedad sería la siguiente:

- Propietario
- Estado de Hipoteca
- Renta

En caso de ser una Avenida se llegaría a contar con la siguientes características igualmente:

- Color de la Avenida
- Casas construidas (El hotel puede ser visto como 5 casas construidas)
- Set de color completo

En cuanto al set de color completo, más que una característica es simplemente información que se puede inferir viendo si las demás Avenidas, partes del mismo set de color, pertenecen al mismo dueño, aún así es información que afecta a características intrínsecas de una Avenida como lo son: su renta y las casas construidas. Debido a tales razones se considera que es información relevante y debe ser considerada. En sí, una representación de una propiedad sería el conjunto de ambas listas, simplemente teniendo en cuenta de que en el caso de que la propiedad no sea una Avenida las características propias a esta tendrían un valor nulo.

En cuanto a su representación matemática, esta se ve explicada en la tabla 3.5

Para la representación del propietario, en el vector se estaría marcado con un 1 el identificador del jugador propietario. Los significados de las variables presentadas serían los siguientes:

Característica	Representación
Propietario	$v \in R^4$
Estado de Hipoteca	0|1
Renta	$renta/\mu_{rentas}$
Color de la Avenida	$idx_{color}/cantidad_{colores}$
Casas construidas	num_{casas}/max_{casas}
Set de Color completo	0|1

Tabla 3.5: Características de una propiedad

- μ_{rentas}: media de las rentas
- idx_{color}: índice del color, si cada color estuviera representado por un índice
- $cantidad_{colores}$: cuantos colores de Avenida están presentes en el juego
- num_{casas}: cantidad de casas construidas
- max_{casas}: número máximo de casas posibles para una Avenida

Las características de un jugador son simples, definen su estado en la partida. Tales caraceterísticas son las siguientes:

- Posición en el tablero
- Cantidad de dinero que posee
- Si posee la carta de salir gratis de la cárcel
- Cantidad de sets completados

Las propiedades que posee el jugador igualmente es una característica, pero esto ya se encuentra represetnado en las propiedades, por la característica de propietario. La representación matemática de las características del jugador es la presentada en la tabla

Respecto a la posición, se la divide por 40 debido a que son 40 casillas en el tablero, la lógica es similar con la cantidad de sets.

Ya con los componentes principales definidos, se puede pasar a analizar otra cuestión fundamental, ¿qué información es usada para tomar una decisión?. Las posibles decisiones de un jugador serán analizadas en el diseño del espacio de acción, a lo que se pretende llegar con la cuestionante anterior es el analizar que información se considera útil para un jugador

Característica	Representación
Posición	$pos/40$
Dinero	$dinero/\mu_{total}$
Carta	0|1
Cantidad de sets completados	$num_{sets}/8$

Tabla 3.6: Características de un jugador

y en que situaciones. Como se mencionó anteriormente, solo hay un tipo de decisiones que se encuentran involucradas en un conexto distinto a las demás, las que el proceso de la acción fue iniciada por otro ente aparte del jugador. Analizando más detenidamente se puede llegar a la conclusión de que tales acciones van relacionadas con el comercio. Debido a esto se determinó que la información relevante para tales acciones es la siguiente:

- Propiedades ofrecidas
- Propiedades pedidas a cambio
- Dinero ofrecido
- Dinero pedido a cambio
- Propiedades del jugador ofertante
- Propiedades del jugador objetivo

Ya que las reglas de Monopolio prohiben comerciar con propiedades que tengan construcciones, sea quien sea el ente ofertador, no habría motivo para contar con más información más que un indicador de cuál propiedad es cuál. Una representación binaria para las propiedades resulta perfecta, marcando un 1 como indicador. En cuanto al dinero, para otorgarle al agente una mejor métrica, se debe dividir las dos cantidades presentadas por el dinero del jugador objetivo.

3.1.3. Función de recompensa

Este es otro componente que va correlacionado con los espacios de Estado y Acción. Con su rol ya descrito en la sección , la función de recompensa debe representar el obejtivo al que el Agente Inteligente debe apuntar. Se puede cometer el error de asociar este objetivo únicamente con ganar, este no es el caso, pues el ganar una partida puede ser descompuesto

en otros objetivos relevantes que si son cumplidos garantizan la victorio. Como ejemplo, no es posible ganar una partida sin ser dueño de una propiedad.

Hay dos factores determinantes que definen el desenlace de un juego de Monopolio, las propiedades y el dinero. De ambos, el factor de las propiedades es más determinante pues la probabilidad de que los oponentes pierdan dinero aumenta mientras más propiedades se posea. También se debe tomar en cuenta las casas construidas, la renta de una Avenida sin contrucciones es menos de la mitad de la renta que tendría si tuviera solo una casa.

No es necesario que la función trabaje solo con un factor, puede trabajar con todo lo mencionado anteriormente. La función de recompensa que se plantea es la ecuación

$$r(x) = \frac{x}{1 + |x|} \tag{3.1}$$

$$x = asset_{factor} + finance_{factor} \tag{3.2}$$

$$asset_{factor} = \begin{cases} j - \mu_p \ ; \ si \ c < 1 \\ (j - \mu_p) + (\frac{h}{5*c}) \ ; \ si \ c \geq 1 \end{cases} \tag{3.3}$$

$$finance_{factor} = d_j - \mu_{op} \tag{3.4}$$

Donde cada componente significa:

- j: número de propiedades del jugador
- μ_p: media del número de propiedades de los oponentes
- c: número de sets de colores completados
- h: número de casas construidas
- d_j: dinero del jugador
- μ_{op}: media del dinero de los oponentes

La función planteada pretende guiar al agente a priorizar estar siempre por encima de la media, tanto en lo que respecta a propiedades como dinero. En cuanto a las casas construidas, la función suma valor si se tiene más de una casa construida, igualmente toma en cuenta los sets completados para calcular tal valor disminuyendo si se tiene muchos sets pero pocas casas, incinuando así al agente a construir casas.

3.2. Estructura interna del Agente

Monopolio posee una cantidad de posibles estados abismal, métodos clásicos como *Q-Learning* [5] se vuelven directamente imposibles aún contando con las máquinas actuales. Es debido a tal desafío que las redes neuronales se hacen presentes para la toma de decisiones. Las redes neuronales permiten afrontar la toma de decisiones en ambientes continuos, Monopolio se asemeja a tal caso.

Con el espacio de Estados ya definido se puede definir facilmente el tamaño de la capa de entrada, y con la información del espacio de Acción se define el tamaño de la capa de salida. Pero como se mencionó ya en el diseño del espacio de acción, las acciones iniciadas por otro ente están involucradas en un contexto diferente a las demás acciones, es debido a este que resulta conveniente separar a las acciones de acuerdo a esta categoría. El espacio de Estado para esta separación también fue presentado ya en el diseño del espacio de Estado. La esrtuctura del agente resultante es la mostrada en la figura 3.1

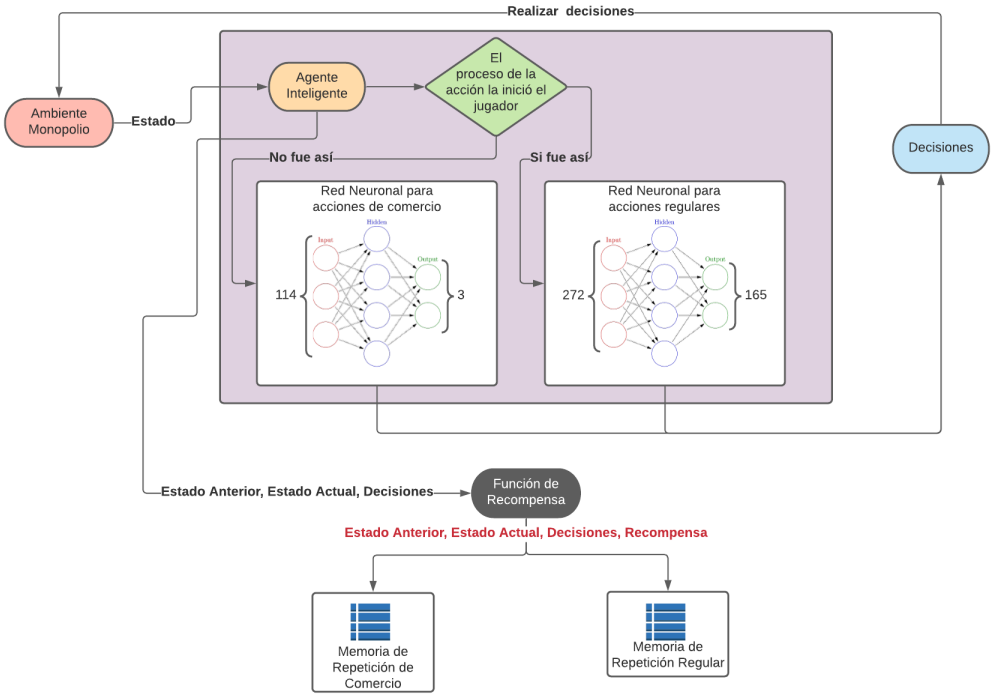

Figura 3.1: Estructura del Agente Inteligente

El tipo de red neuronal que se plantea usar es la secuencial (conversión lineal), no se requiere trabajar con redes convolucionales, ni con redes con memoria a corto y largo plazo

(LSTM). Esto simplifica su funcionamiento, otorgando mayor rapidez en la toma de decisiones y en el etrenamiento, el cuál estará basado en *Deep Double Q-Learning* [14] para evitar la sobre-estimación de parámetros, frecuente en *Deep Q-Learning*. El implementar *Deep Double Q-Learning* implica que se debe trabajar con dos redes neuronales para poder realizar el cálculo de la actualización de los valores Q. Para este caso en particular, cada Agente Inteligente contaría con cuatro redes, dos para la toma de decisiones, y para cada una de estas otra red objetivo (target) para el entrenamiento.

Igualmente se contará con Memorias de Repetición, o en otras palabras buffers donde almacenar el estado, la acción, la recompensa a la acción y el estado resutante. En métodos clásicos se tendía a entrenar al agente con cada interacción, pero resutla más óptimo ir guardando la experiencia obtenida para después entrenar al agente de golpe, sacando un lote de esta información almacenada. Cada red destinada a un tipo de acción tendrá su propia memoria de repetición.

Cabe resaltar que debido a este planteamiento, el agente está diseñado para siempre intentar realizar una acción, sin importar que esta pueda resultar inválida, ya que no se está penalizando a ninguna red por la toma de decisiones inválidas, esto podría realizarse mediante la función de recompensa o en el guardado de los valores en la memoria de repetición marcando los valores como negativos de alguna forma; aunque se desconoce si esto vaya a tener un efecto positivo. La razón por la que se determinó no penalizar las acciones inválidas es para maximizar la cantidad de experiencia recolectada, a mayor cantidad de intentos el agente explora más su entorno y consecuentemente aprende la mejor manera de interactuar con él.

3.3. Implementación de Monopolio

Para la implementación del juego de Monopolio se hizo uso del lenguaje de programación Python, esto pensando más que nada en la implementación del Agente Inteligente 3.4, siendo Python el lenguaje predilecto para el Aprendizaje de Máquina por todas las librerías y documentación disponibles. Python no es el lenguaje más eficiente, pero posee varias bondades como una fácil legibilidad y uso debido a los principios *Zen* en los que fue basado. Aún siendo un lenguaje interpretado con una estructura similar a los *lenguajes de scripting*[1], Python también puede ser usado para proyectos de Programación Orienteada a Objetos (POO). Para la implementación de Monopolio se hizo uso de fundamentos de POO tal como patrones

[1]Lenguajes de programación que son utilizados para la rápida satisfacción de exigencias comunes.

de diseño[2] a la vez de conceptos de herencia y polimorfismo. El diagrama de clases de la implementación de Monopolio sería el de la figura 3.2

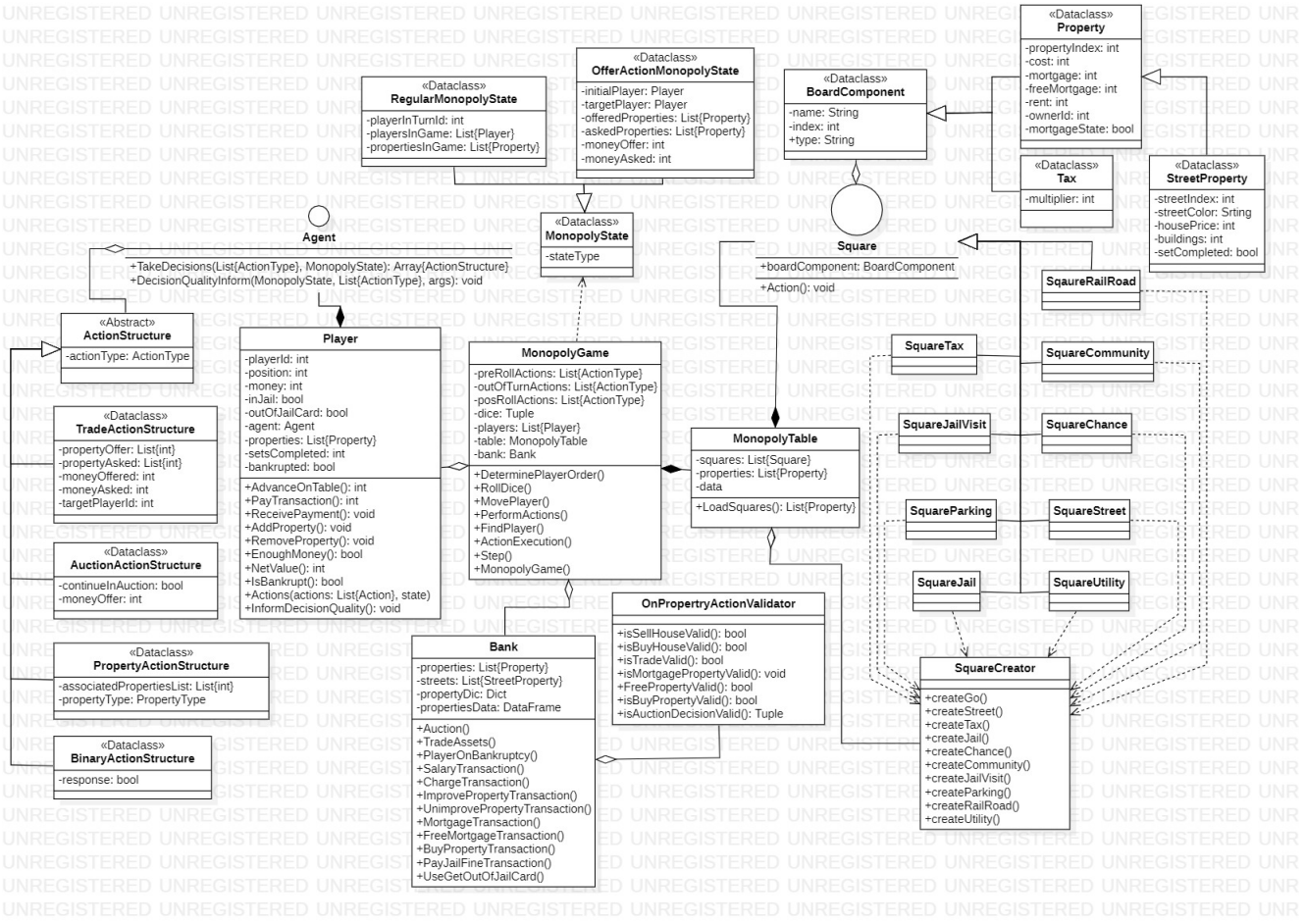

Figura 3.2: Diargama de clases Implementación Monopolio

El único patrón de diseño usado es el *Método Fábrica (Factory Method en inglés)* el cuál es de la categoría creacional y brinda la facilidad de crear elementos en tiempo de ejecución. La razón de uso de este patrón es que cada casilla de Monopolio tiene una acción diferente la cual depende del tipo de casilla y de la iformación contenida en esta misma; como ejemplos se puede mencionar a la acción de la casilla de Go que es el de otorgar 200$ al jugador que pase sobre ella lo cuál es una acción propia del tipo de la casilla, igualmente la acción de cobrar renta es propia de las casillas de tipo propiedad, pero la cantidad de la renta

[2]Un patrón de diseño describe un problema recurrente en el diseño de software orientado a objetos y describe le núcleo de una solución para el problema.

depende de información contenida dentro de la casilla, cuantas casas se tiene construida en la propiedad, si se tiene todo el conjunto completo, a que color pertence, etc. Toda esta información ensuciaría el código si es configurada manualmente, es debido a esto que se hace uso de un archivo *.csv* y se crea las casillas al momento de inicializar el tablero contenido en *MonopolyTable* con la clase *SquareCreator*.

El polimorfismo y herencia puede ser apreciado múltiples veces en el diseño del programa, siendo el del Agente el más importante. El agente es el encargado de la toma de decisiones del jugador a quién fue asignado, el polimorfismo permite que pueda haber diferentes tipos de agente que hereden de la interfaz y que tomen las decisiones con procedimientos diferentes sin que se tenga que cambiar nada en el resto del programa. Esto fue esencial para las implementaciones del Agente Inteligente y de los Agentes Fijos.

Para trabajar con clases abstractas e interfaces se hizo uso de la librería *ABC*, para las *clases de datos*[3] se hizo uso de la librería *Dataclasses* y también se utilizó *Pandas*[4] para el manejo del archivo *.csv*, no se requirió el uso de ninguna otra librería para la programación del juego de Monopolio.

3.4. Implementación del Agente Inteligente

Para la implementación del Agente Inteligente y su estructura interna definida en la sección 3.2 se hizo uso de *PyTorch*[5]. Dicha biblioteca tiene más versatilidad de configuración a diferencia de *Keras* y es más comprensible a diferencia de *Tensorflow*, ambos siendo otras bibliotecas de Aprendizaje Reforzado.

De PyTorch se usaron diferentes módulos para obtener los componentes necesarios para la la construcción de la red y su entrenamiento. Del módulo *optim* de obtuvo el optimizador *Adam*, del módulo *nn* se obtuvo el contructor de las capas *nn.Linear* al igual que el constructor de una red neuronal secuencial *nn.Linear*, además de la clase padre *nn.Module* sobre el cuál la clase hija que implemente la red neuronal debe heredar; igualmente se obtuvo la función de pérdida usada para el entrenamiento de la red.

De las librerias propias de Python se hizo uso de *collections* para poder hacer uso de *OrderedDict* en pos de poder inicializar la red neuronal en base a un diccionario ordenado para poseer así mayor versatilidad sobre su contrucción. Esto fue útil más que nada para la

[3]Set de atributos y sus valores, es decir una clase cuya función principal es almacenar información.

[4]Herramienta de anális y manipulación de datos contruido sobre Python.

[5]Una biblioteca de Aprendizaje Automático basada en la biblioteca Torch y desarrollado por el laboratorio de Inteligencia Artificial de Facebook FIAR.

mutación de arquitectura y de función de activación, ya haciendo uso del marco de trabajo *SEARL* con la configuación descrita en la sección 3.6.

Por último, la librería *collections* también es usada para la memoria de repetición o memoria compartida. De *collections* se obtiene *deque* para poder inicializar un buffer donde almacenar los diferentes elementos a tomar en cuenta para el entrenamiento del agente, en este caso la información a almacenar sería la siguiente:

- Estado anterior
- Decisiones tomadas
- Recompensa a las decisiones tomadas
- Estado resutlante

Haciendo uso de la ecuación de *Deep Double Q-Learning* 2.3 y de la función de pérdida, a base de la información anterior, se puede realizar el ajuste en los pesos de la red; es decir se puede entrenar al agente. Después de realizar esta operación para todas las muestras obtenidas del *buffer* se copiará los pesos de la red principal a la red obejtivo (target).

3.5. Implementación de Agentes Fijos

Los Agentes fijos que se implementaron siguen las mismas reglas usadas por los Agentes Fijos que priorizan cierto tipo de propiedad usados en el trabajo *Toma de decisiones en Monopolio usando un acercamiento de Aprendizaje Reforzado Profundo Híbrido* 2.6. En total son dos.

Estos se basan principalmente en la estrategia de evitar algunas propiedades, en este caso a las *utilidades*, al mismo tiempo de priorizar otras, el primero dará preferencia a los *ferrocarriles* y a las avenidas de color *azul oscuro*, el segundo dará prefernicia a las avenidas de color *celeste* y *anaranjado*. Son agentes capaces de comerciar con propiedades pero sin dinero de por medio, tratando siempre de completar un set y deshaciendose de las propiedades sueltas que se tenga.

3.6. Planteamiento de SEARL para el entrenamiento

La población inicial de agentes será de un tamaño de 12 de los cuales 10 serán Agentes Inteligentes y tan solo dos serán Agentes Fijos, los mencionados en la sección 3.5. Los Agentes

Fijos no recibirán ningún tipo de mutación ni de entrenamiento, estos son incluidos dentro de la población inicial para ofrecer una guía a los otros agentes inteligentes en el principio de su evolución.

La razón de no especificar los hiper-parámetros y la arquitectura interna de las redes en el punto 3.2 es debido a que justamente el marco de trabajo encargado de encontrar tal configuración es SEARL. Los hiperparámetros con los que el agente dispone, junto con sus valores iniciales, son los siguientes:

- α : 0,003: Frecuencia de aprendizaje (learning rate)
- γ : 0,80: Usado para calcular la périda al momento del entrenamiento
- τ : 0,02: Usado para la copia de los pesos de la red de decisiones a la red objetivo.
- ϵ : 0,5: Es lo característico del algoritmo *off-policy*, este hiper-parámetro es usado para balancear la exploración de la explotación.

La arquitectura inicial de una red neuronal esta definida por dos capas ocultas de 64 nodos de profundidad, ambas con una activación Relu que devuelve 0 en todos los valores negativos, y retorna positivos. La función de activación de la capa de salida es sigmoide, esto para facilitar la conversión de lo numérico al formato del ambiente de Monopolio.

Como se puede observar en la imagen 2.2, durante el periodo de evaluación es cuando se recolectará experiencia en la memoria comaprtida. Esto no implica un cambio en lo explicado en la estructura interna del agente. La razón detrás de la eficiencia de SEARL es justamente esta memoria global que permite el compartir experiencia entre miembros de la población de agentes. Como se mencionó en el punto 3.2, el agente requiere de dos memorias de repetición una para cada tipo de red, y en lugar de crearlas individualemente para cada agente, serán dos memorias de repetición globales compartidas por todos los Agentes Inteligentes.

Igualmente, es en el periodo de evaluación donde se calculará el *valor de ajuste* de cada agente, usado para el periodo de selección. Todos los agentes jugarán una partida de Monopolio contra otros 3, es decir todos tendrán solo una oportunidad de obtener el máximo *valor de ajuste* posible. Este valor es la media de la recompensa obtenida a lo largo de toda la evaluación.

La selección por torneo otorga la habilidad de filtrar a los miembros de la población, al ser una selección elitista priorizará al agente que mejor desempeñe. El desempeño es medido por el *valor de ajuste*, el agente con el mejor valor es guardado como élite en la siguiente generación sin borrarlo de la actual, ya después el torneo realizado es de tamaño $k = 4$ ya

que el tamaño de la población es múltiplo de 4. Se pasa a la siguiente generación al hijo del agente con mejor *valor de ajuste* entre los cuatro seleccionados.

Las posibles mutaciones son las mismas definidas en el trabajo original 2.4:

- Mutación en los pesos de la red neuronal haciendo uso de ruido Gaussiano.
- Cambio en la función de activación en las capas ocultas.
- Cambio en la arquietctura interna de la red, ya sea agregando un nodo o una capa oculta entera.
- Cambio en los hiperparámetros del agente.
- Ninguna mutación.

Las mutaciones permitirán explorar distintas opciones en busca de la configuración óptima para el agente.

Por último, para el entrenamiento se obtendrá una cierta cantidad de muestras de la memoria compartida. El tamaño de este lote para el muestreo no está dentro de las mutaciones de los hiper-parámetros, pues la idea es que a medida que se tenga más experiencia guardada el agente sea capaz de entrenar y aprender más. Debido a esto el tamaño del lote siempre será el 25 % del total de experiencia almacenada dentro de la memoria de repetición o memoria compartida.

Capítulo 4

Resultados y Discusión

La experimentación determinada para este proyecto consta de obtener al *agente* élite de la última generación, a un segundo agente seleccionado de manera aleatoria igualmente de la última generación; juntarlos con los dos agentes fijos planteados en 3.5 para formar una nueva población. Posteriormente, ejecutar únicamente la etapa de evaluación 100 veces para recaudar información y así analizar los valores de ajuste de cada individuo, además de las victorias y las acciones realizadas; igualmente se tomarán en cuenta los datos obtenidos de las diferentes generaciones de agentes.

Empezando por las acciones, primeramente se obtuvo un conteo de cuantas de ellas fueron válidas/inválidas a lo largo de las generaciones, para tener un punto de referencia al analizar lo obtenido en los experimentos. Los resultados, mostrados en la tabla 4.1, eran de esperarse, de 1,479,458 acciones realizadas tan solo el 11 % fue válido. Esto es así debido a que como se mencionó en la sección 3.2, el Agente Inteligente está diseñado para siempre intentar una acción, sin importar si esta será válida o no.

Un aspecto interesante que es arrojado por los resultados de las últimas dos generaciones 4.2, es que el Agente Inteligente nunca aprendió a comerciar. Se puede observar 16,878 intentos fallidos y 0 intentos exitosos, cosa que indica que todas las ofertas válidas realizadas provienen de los *agentes fijos*, los cuales se extinguieron en la cuarta generación. Algo positivo es que durante las primeras dos generaciones; de 103,496 acciones, aún contando con agentes fijos, tan solo el 7 % de las acciones fueron válidas; en cambio en las últimas dos generaciones de 99,002 acciones, el 12 % fueron válidas, esto es una muestra de mejora por parte del *Agente Inteligente*.

Se pasó a analizar la historia de evolución del *agente élite*, comparándolo con el otro agente perteneciente a la última generación, el *agente aleatorio*, llegando a la conclusión

Tipo de Acción	Validas	Invalidas
Realizar Oferta	1108	243874
Continuar en Subasta	7381	3392
Construir casa	33318	211910
Vender casa	42959	278360
Hipotecar Propiedad	35427	285892
Pagar Hipoteca	37433	283918
(Carta) Salir de la Cárcel	3671	2146
Pagar Fianza	5682	135
Aceptar Oferta	1108	0
Comprar Propiedad	1720	24

Tabla 4.1: Tabla de las Acciones Válidas/Inválidas realizadas por las diferentes generaciones de Agentes

Tipo de Acción	Validas	Invalidas
Realizar Oferta	0	16878
Continuar en Subasta	144	287
Construir casa	2612	14266
Vender casa	3523	17836
Hipotecar Propiedad	2935	18424
Pagar Hipoteca	2960	18399
(Carta) Salir de la Cárcel	180	106
Pagar Fianza	281	5
Aceptar Oferta	0	0
Comprar Propiedad	165	1

Tabla 4.2: Tabla de las Acciones Válidas/Inválidas realizadas por las dos últimas generaciones de Agentes

de que pese a la ineficiencia en la toma de decisiones, se pueden apreciar valores de ajuste bastante altos a través de las diferentes generaciones. Los resultados mostrados en la imagen 4.1 indican que ambos son descendientes de los mismos antecesores, debido a esto las líneas de ambos coinciden separándose únicamente en las última generación, en la que el *agente élite* obtuvo un repique mucho más pronunciado.

Con la segunda imagen 4.2 se puede concluir que la familia del *agente élite* y del *agente aleatorio* siempre fueron sobresalientes. Exceptuando un punto bajo en la sexta generación, siempre se mantuvieron igual o por encima de la media. Además de esto, obtuvieron el 47 % de las victorias disputadas sobre las 20 generaciones.

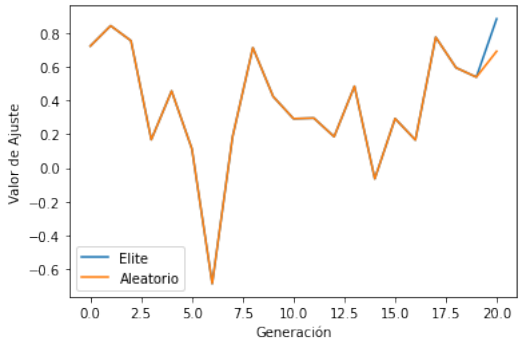

Figura 4.1: Historia a través de las 20 generaciones: Agente élite vs Agente Aleatorio

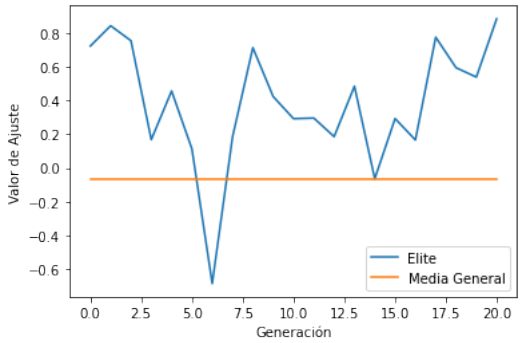

Figura 4.2: Historia a través de las 20 generaciones: Agente élite vs Agente Aleatorio

Aún siendo individuos sobresalientes, al momento de las 100 evaluaciones junto con los agentes fijos, tanto el *agente élite* como el *agente aleatorio* se vieron ampliamente superados. Fueron 54 victorias para el agente fijo que prioriza los ferrocarriles y las avenidas de color azul, este está bautizado con el seudónimo de *Fijo 0*; le sigue con 43 victorias el agente fijo que prioriza las avenidas de color celeste y anaranjadas, bautizado con el seudónimo *Fijo 1*; con 2 victorias se encuentra el *agente aleatorio* y como último, con tan solo 1, se encuentra el *agente élite*.

Aunque parezca una derrota apabullante se debe recordar que el objetivo del Agente Inteligente es el de maximizar el valor de la recompensa, el cuál va muy relacionado con el *valor de ajuste*. La media del *valor de ajuste* del *agente élite* es de 0.4611 y la del *Fijo 0* es de 0.4979. El *agente élite* es quién más cercano está a los valores del *Fijo 0* como se puede apreciar en la imagen 4.3.

La media del *agente aleatorio* es de 0.4424 y la del *Fijo 1* es de 0.4458, es decir el *agente élite* supera a ambos por aproximadamente $\frac{2}{3}$ de la diferencia que posee con el *Fijo 0*, pese

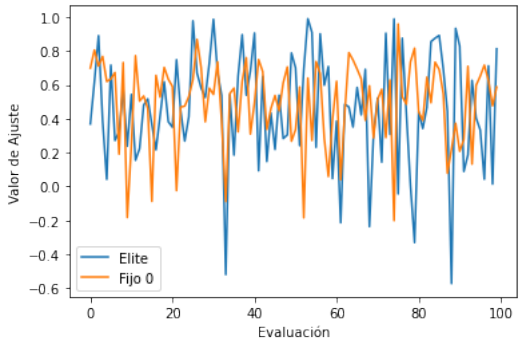

Figura 4.3: Experimentos: Agente élite vs Fijo 0

a posicionarse como último en el número de victorias. Aún así, las diferencias que presentan entre sí en los valores de ajuste no es muy alta, analizando el *agente élite*, el error cuadrático con los demás no es muy elevado siendo los valores:

- Aleatorio: 0.2516
- Fijo 0: 0.1715
- Fijo 1: 0.1837

Esto brinda una mejor visión sobre la diferencia real entre el rendimiento del *agente élite* con respecto a los demás, los valores muestran que este tiene un error menor con los agentes fijos. En la figura 4.4 se puede apreciar que que el *agente aleatorio* presenta picos más bajos y no tiene picos tan altos como los del *agente élite*, justificando así el error obtenido.

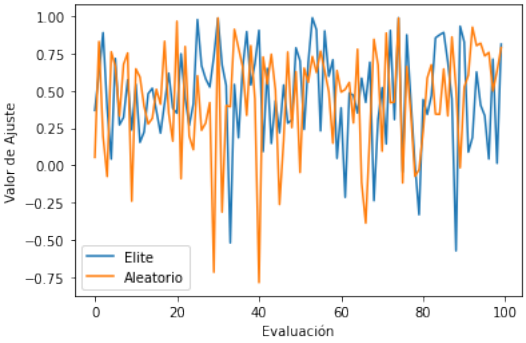

Figura 4.4: Experimentos: Agente élite vs Aleatorio

Se tiene una situación distinta con el *Fijo 1*, el cuál presenta un gráfico más parecido al del *Fijo 0*, donde si bien no alcanza valores tan altos como el *agente élite*, sus picos bajos no

son tan pronunciados, lo que lo hace más consistente y es la causa del valor menor del error. Dicho gráfico es el 4.5.

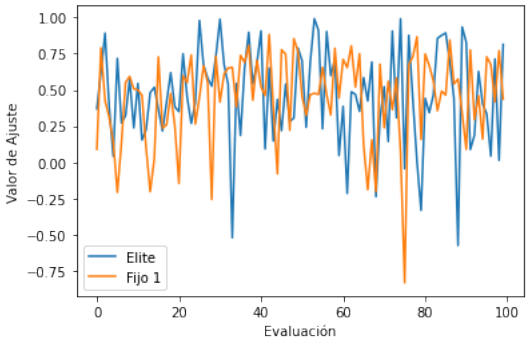

Figura 4.5: Experimentos: Agente élite vs Fijo 1

Aún con el buen rendimiento con respecto al *valor de ajuste*, el propósito principal de todo el entrenamiento era el poder encontrar un Agente Inteligente que sobrepasara al planteado en el trabajo *Toma de decisiones en Monopolio usando un acercamiento de Aprendizaje Reforzado Profundo Híbrido* [7]. En dicho trabajo, su planteamiento híbrido alcanza un 76.91 % de victorias sobre 2000 partidas jugadas, si bien son 20 veces más que las evaluaciones realizadas en este proyecto, cabe resaltar que el planteamiento híbrido aprende contínuamente sobre las 2000 partidas, en cambio el planteamiento de este proyecto pretendía optimizar eso mediante las memorias compartidas. Igualmente no se entrena a la población durante los experimentos, únicamente durante el proceso de desarrollo de las generaciones.

Incluso el agente estándar de *Toma de decisiones en Monopolio usando un acercamiento de Aprendizaje Reforzado Profundo Híbrido* [7] presenta un procentaje de victoria mucho más alto, 47.41 %.

Capítulo 5

Conclusiones

La falta de victorias pese al alto *valor de ajuste* durante los experimentos indica una contradicción. Durante el desarrollo de las generaciones sí se tenía una relación coherente, donde el agente ganador es normalmente aquel que tiene valores altos, prueba de esto es el mismo desarrollo de la familia del *agente élite*. Es durante los experimentos que sale a relucir que el *valor de ajuste* no es un buen indicador para el Agente Inteligente. El *valor de ajuste* deriva de la función de recompensa, por lo que es esta quién no plasma correctamente el objetivo que se desea, ganar.

Es probable que si el Agente Inteligente hubiera sido capaz de comerciar, su cantidad de victorias se hubiera disparado en contra de los Agentes Fijos. Para validar esto se debe de aproximar el problema desde un punto que priorice el aprendizaje de comercio. Antes se mencionó la posibilidad de penalizar al agente por una acción inválida, debido a que ambas redes están diseñadas para tomar múltiples decisiones una sola función de recompensa general es incapaz de indicar al agente cuando una acción en específico fue inválida. Un planteamiento que use funciones de recompensa específicas por acción puede resultar más óptimo para guiar al agente.

Igualmente, puede que el modelo óptimo se encuentre en un planteamiento híbrido, pero diseñado de una manera distinta a *Toma de decisiones en Monopolio usando un acercamiento de Aprendizaje Reforzado Profundo Híbrido*, en el sentido de que no se reemplace la toma de decisión completamente, sino que sirva más que nada como una ayuda para el agente. Un ejemplo puede ser que el agente decida si continuar en la subasta, pero la oferta de dinero es determinada por una función.

La división de la toma de decisiones en dos redes neuronales, en la medida de lo posible, dió buenos resultados. Es verdad que los Agentes Inteligentes solo tuvieron la oportunidad

de aceptar ofertas de comercio cuando contaban con la presencia de agentes fijos, pero esta red también era usada durante las subastas y, en las tablas 4.1, 4.2, se puede observar un buen uso de esta acción.

En resumen, se pudo alcanzar los objetivos específicos, pero la falla de alcanzar el objetivo general indica que este planteamiento se descarta como el más óptimo. La razón de esto puede estar en lo descrito anteriormente, o incluso en la falta de objetivos específicos por cumplir, en todo caso el proyecto si pudo brindar valor científico al plantear nuevos espacios de estado y acción simplificados, exceptuando la función de recompensa, pues se considera que el problema se encuentra en el uso y planteamiento de esta.

Capítulo 6

Recomendaciones

Para estudios posteriores sobre este tema se recomienda explorar con distintas posibilidades de diseño, no solo con una única. Como ejemplo se puede poner el uso de LSTM[1] para la red usada para comerciar, así toma en cuenta ofertas anteriores en la toma de decisiones futura. O el planteamiento de distintas mutaciones para el Agente tal que ayuden en su proceso de aprendizaje.

Como se mencionó en el anterior punto, también se recomienda hacer uso de funciones de recompensa específicas para cada acción en lugar de una única función de recompensa general. Se considera que la función de recompensa es necesaria pues cada acción puede ser calificada como favorecedora tomando en cuenta distintos parámetros. De esa manera se puede prestar una mejor guía al Agente Inteligente durante el periodo de aprendizaje.

Para ayudar al Agente Inteligente a aprender a comerciar se puede almacenar las decisiones de comercio de los Agentes Fijos, al menos por una determinada cantidad de partidas, para que el agente tenga una buena fuente de experiencia sobre la cuál dar sus primeros pasos. Esto no necesariamente implica que el Agente Inteligente se vaya a enfrascar en el criterio sobre el cuál un Agente Fijo toma la decisión, pues la diferencia de preferencias evita esto, además si se está haciendo uso de SEARL el agente de por sí cambiará su comportamiento a medida que sufra mutaciones y las generaciones avancen.

Se plantea como última recomendación brindarle más prioridad al diseño del Agente Inteligente, antes de cualquier implementación. El correcto funcionamiento de un Agente Inteligente se encuentra en su configuración de hiper-parámetros, en la arquitectura y pesos de su red, etc; pero también está presente en las bases teóricas sobre las cuales está construido el núcleo de su funcionamiento. En este caso se hizo uso de redes neuronales secuenciales, las

[1] Redes neuronales de memoria a corto y largo plazo.

más simples, pero como se menciona en una recomendación anterior, se pueden implementar otro tipo de redes y para saber cuál implementar se debe de tener buenas bases de conocimiento sobre su funcionamiento. Lo mismo aplica para el entrenamiento, experimentación, etc.

Bibliografía

1. TURING, Alan M. Computing machinery and intelligence. En: *Parsing the turing test*. Springer, 2009, págs. 23-65.
2. ABBASS, H. Editorial: What is Artificial Intelligence? *IEEE Transactions on Artificial Intelligence*. 2021, vol. 2, n.º 02, págs. 94-95. Disp. desde DOI: 10.1109/TAI.2021.3096243.
3. *AlphaGo* [https://deepmind.com/research/case-studies/alphago-the-story-so-far]. [s.f.]. Accessed: 2021-11-04.
4. ENCYCLOPAEDIA BRITANNICA, The Editors of. *Monopoly* [Available at https://www.britannica.com/sports/Monopoly-board-game (4/05/2021)]. [s.f.].
5. WATKINS, Christopher JCH y DAYAN, Peter. Q-learning. *Machine learning*. 1992, vol. 8, n.º 3-4, págs. 279-292.
6. BAILIS, Panagiotis; FACHANTIDIS, Anestis y VLAHAVAS, Ioannis. Learning to play monopoly: A reinforcement learning approach. En: *Proceedings of the 50th Anniversary Convention of The Society for the Study of Artificial Intelligence and Simulation of Behaviour. AISB*. 2014.
7. HALIEM, Marina; BONJOUR, Trevor; ALSALEM, Aala; THOMAS, Shilpa; LI, Hongyu; AGGARWAL, Vaneet; BHARGAVA, Bharat y KEJRIWAL, Mayank. *Decision Making in Monopoly using a Hybrid Deep Reinforcement Learning Approach*. 2021. Disp. desde arXiv: 2103.00683 [cs.LG].
8. *Monopoly* [https://www.hasbro.com/common/instruct/monins.pdf]. [s.f.]. Accessed: 2021-11-05.
9. DARR, Benjamin J y COHEN, Alexander H. The rules of the game: Experiencing global capitalism on a monopoly board. *Journal of Political Science Education*. 2016, vol. 12, n.º 3, págs. 268-281.

10. ANSOMS, An y GEENEN, Sara. Development Monopoly: A simulation game on poverty and inequality. *Simulation & Gaming*. 2012, vol. 43, n.º 6, págs. 853-862.

11. ENDER, Morton. Modified monopoly: Experiencing social class inequality. *Academic Exchange Quarterly*. 2004, vol. 8, n.º 2, págs. 249-253.

12. SUTTON, Richard S y BARTO, Andrew G. *Reinforcement learning: An introduction*. MIT press, 2018.

13. GOYAL, Vineet y GRAND-CLÉMENT, Julien. *Robust Markov Decision Process: Beyond Rectangularity*. 2021. Disp. desde arXiv: 1811.00215 [math.OC].

14. VAN HASSELT, Hado; GUEZ, Arthur y SILVER, David. Deep reinforcement learning with double q-learning. En: *Proceedings of the AAAI conference on artificial intelligence*. 2016, vol. 30. N.º 1.

15. CARR, Jenna. An introduction to genetic algorithms. *Senior Project*. 2014, vol. 1, n.º 40, pág. 7.

16. FRANKE, Jörg K.H.; KOEHLER, Gregor; BIEDENKAPP, André y HUTTER, Frank. Sample-Efficient Automated Deep Reinforcement Learning. En: *International Conference on Learning Representations*. 2021. **urlalso**: https://openreview.net/forum?id=hSjxQ3B7GWq.

17. COHEN, P.R. y HOWE, A.E. Toward AI research methodology: three case studies in evaluation. *IEEE Transactions on Systems, Man, and Cybernetics*. 1989, vol. 19, n.º 3, págs. 634-646. Disp. desde DOI: 10.1109/21.31069.