UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Optimización del comportamiento de bots aplicado a teoría de juegos

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Sanchez Gregori, Edgar

Tutor/a: Herrero Debón, Alicia

Cotutor/a: Ardid Ramírez, Joan Salvador

CURSO ACADÉMICO: 2023/2024

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

OPTIMIZACIÓN DEL COMPORTAMIENTO DE BOTS APLICADO A TEORIA DE JUEGOS

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Edgar Sánchez Gregori

TUTORIZADO POR

Alicia Herrero Debón

Joan Salvador Ardid Ramírez

CURSO ACADÉMICO: 2023/2024

Resumen

La flexibilidad, rapidez y robustez en el aprendizaje y, en definitiva, en la toma de decisiones son aspectos clave a la hora de desarrollar robótica humanoide. En este TFG se analizará el comportamiento de bots en juegos competitivos de estrategia mixta, tales como el juego piedra-papel-tijeras o el "matching pennies", que es una simplificación que únicamente considera 2 alternativas. En el TFG, los bots serán operados mediante técnicas de aprendizaje por refuerzo ("reinforcement learning", RL), dotadas éstas de la capacidad de aprendizaje continuado y/o de estructuras de meta-aprendizaje. En este TFG nos planteamos un doble objetivo, por un lado, se pretende optimizar el comportamiento de los agentes de RL compitiendo contra otros algoritmos computacionales (véase https://www.kaggle.com/c/rock-paper-scissors). Por otro lado, se pretende entender mejor nuestra toma de decisiones en este tipo de juegos: los primates, incluyéndonos a nosotros humanos, somos incapaces de seleccionar opciones de forma aleatoria y, por tanto, nos desviamos del comportamiento óptimo en este tipo de juegos. Esta incapacidad la suplantamos mediante cambios de estrategia (por ejemplo, win-stay/lose-switch) a partir de mecanismos de memoria e inferencia basada en la teoría de la mente. El uso del aprendizaje por refuerzo para el estudio del comportamiento humano (y animal) nos permite entender cómo se producen las decisiones en situaciones complejas.

Resum

La flexibilitat, rapidesa i robustesa en l'aprenentatge i, en definitiva, en la presa de decisions són aspectes clau a l'hora de desenvolupar robòtica humanoide. En aquest TFG s'analitzarà el comportament de bots en jocs competitius d'estratègia mixta, com ara el joc pedra-paper-tisores o el "matching pennies" que és una simplificació que únicament considera 2 alternatives. En el TFG, els bots seran operats mitjançant tècniques d'aprenentatge per reforç ("reinforcement learning", RL), dotades aquestes de la capacitat d'aprenentatge continuat i/o d'estructures de meta-aprenentatge. En aquest TFG ens plantegem un doble objectiu, d'una banda es pretén optimitzar el comportament dels agents de RL competint contra altres algorismes computacionals (vegeu https://www.kaggle.com/c/rock-paper-scissors). D'altra banda es pretén entendre millor la nostra presa de decisions en aquest tipus de jocs: els primats, incloent-nos a nosaltres humans, som incapaços de seleccionar opcions de manera aleatòria i, per tant, ens desviem del comportament òptim en aquest tipus de jocs. Aquesta incapacitat la suplantem mitjançant canvis d'estratègia (per exemple, win-stay/lose-switch) a partir de mecanismes de memòria i inferència basada en teoria de la ment. L'ús de l'aprenentatge per reforç per a l'estudi del comportament humà (i animal) ens permet entendre com es produeixen les decisions en situacions complexes.

Abstract

Flexibility, speed and robustness in learning and, ultimately, in decision-making are key aspects when developing humanoid robotics. In this TFG the behavior of bots in competitive mixed strategy games will be analyzed, such as the rock-paper-scissors game or the "matching pennies" which is a simplification that only considers 2 alternatives. In the TFG, the bots will be operated using reinforcement learning techniques ("reinforcement learning", RL), endowed with the capacity for continuous learning and / or meta-learning structures. In this TFG we have a double aim, on the one hand it is intended to optimize the behavior of RL agents competing against other computational algorithms (see https://www.kaggle.com/c/rock-paper-scissors). On the other hand, it is intended to better understand our decision-making in these types of games: primates, including us humans, are unable to select options randomly and, therefore, we deviate from the optimal behavior in these types of games. We supplant this inability through changes in strategy (for example, win-stay / lose-switch) based on memory mechanisms and inference based on theory of mind. Using reinforcement learning to study human (and animal) behavior allows us to understand how decisions are produced in complex situations.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

I. MEMORIA

1. Introducción

1.1 Objetivo del proyecto

Actualmente los robots son utilizados en múltiples ámbitos, entre ellos en la industria moderna donde se encargan de realizar trabajos repetitivos y precisos en un entorno controlado en el que realizan tareas sobre las que se conocen de antemano los datos necesarios para resolver el trabajo.

Uno de los campos de investigación que más se está desarrollando dentro de la robótica es el de la inteligencia artificial, cuyo objetivo es que los robots puedan realizar tareas de manera independiente con un comportamiento "inteligente", es decir, que sean capaces de tomar decisiones sobre cómo actuar en un entorno bajo unos parámetros preestablecidos. Para conseguirlo, dentro del ámbito de la inteligencia artificial se encuentra el aprendizaje por refuerzo (*reinforcement learning*), del que se distinguen tres tipos de aprendizaje: supervisado, no supervisado y por refuerzo.
En el aprendizaje supervisado los algoritmos aprenden a través de un conjunto de datos etiquetados donde se específica como debe ser categorizada esa información, es decir, le indicamos lo que queremos que aprenda. Necesita de la intervención humana para proporcionar retroalimentación.
El aprendizaje no supervisado no requiere de la intervención humana. No se dispone de datos etiquetados para indicar como debe ser categorizada la nueva información, los algoritmos deben encontrar la manera de explotarla ellos mismos.
En el aprendizaje por refuerzo los algoritmos aprenden a través de la experiencia. El sistema aprende en base al ensayo-error retroalimentándose de la información que obtiene del mundo exterior como respuesta a sus acciones, como la obtención de un "refuerzo positivo" al acertar en su ejecución [1].

Estos algoritmos tratan de acercarse a la capacidad de tomar decisiones de un humano. Los pensadores de la Antigua Grecia consideraban que estas se producían a partir de un profundo análisis y estaban basadas en el razonamiento. Este pensamiento sigue vigente hoy en día, donde desde el origen del ser humano éste se ha visto en la necesidad de tomar decisiones, desde qué comer, hasta qué hacer con cada aspecto y circunstancia de la vida. En la actualidad las investigaciones relacionadas con las toma de decisiones tiene diversas vertientes, una que se centra en la forma en que las personas evalúan la probabilidad de recibir ganancia y evitar pérdidas de las diferentes alternativas y donde a partir de la información tratan de realizar un cálculo basado en la expectativa para tomar la decisión, y otra basada en "la emoción" donde la respuesta emocional afecta a la toma de decisiones y viene dada por la experiencia o el aprendizaje a partir de situaciones anteriores o parecidas [2].

Sobre la toma de decisiones se han realizado varios estudios analizando su comportamiento, tanto en primates como humanos, como por ejemplo el publicado en la revista científica "Nature" por los profesores Tianming Yang y Michael Shadlen, de la universidad de Washington, en EE. UU. En este trabajo demostraron que los monos, al igual que los humanos, son capaces de tomar decisiones calculando cuál de estas creen que obtendrán mayor recompensa [3]. También se ha estudiado la toma de decisiones a través de la teoría de juegos, donde a través de un juego o interacción social con unas reglas establecidas, se analiza las diferentes estrategias óptimas así como el comportamiento del resto de los sujetos para elegir la opción que otorgue más ganancia. Entre estos juegos, se han realizado estudios de reinforcement learning para el juego de "matching pennies" [4] y de "Piedra, Papel, Tijeras" [5] con monos donde se ha estudiado el patrón de comportamiento de estos al realizar elecciones.

Uno de los objetivos de este proyecto será analizar el comportamiento que tienen los humanos en la toma de decisiones para comprender mejor que factores son más determinantes a la hora de tomar una decisión. Para ello, trataremos de ver cuál de los diferentes algoritmos, en los que cada uno sigue una estrategia diferente, se asemeja más al comportamiento de los participantes viendo cuál tiene más probabilidad de realizar las mismas elecciones que los sujetos. Este comportamiento se analizará a través del juego competitivo "matching pennies", en el que además también optimizaremos el comportamiento de los algoritmos y estudiaremos qué modelo es capaz de ganar más veces en este juego [6]. Los datos de los sujetos los hemos obtenido gracias a la colaboración con investigadores de la escuela de medicina de la Universidad de Yale en New Haven, Connecticut, que nos ha cedido los resultados de su estudio sobre la toma de decisiones en los humanos.

Finalmente, mediante una competición entre distintos bots en el juego de piedra-papel-tijeras realizado a través de la plataforma Kaggle, abordaremos nuestro segundo objetivo. Para ello, analizaremos la optimización del comportamiento de los agentes de RL a través de la competición contra otros algoritmos computacionales viendo qué modelo algorítmico compite mejor en el juego.

La optimización de los algoritmos de aprendizaje por refuerzo nos permitirá dotar a las computadoras de una inteligencia artificial en la que puedan aprender a tomar la acción más indicada en función de la situación en la que se encuentren.

1.2 Estado del arte

En este apartado realizaremos una introducción al aprendizaje por refuerzo y explicaremos los modelos utilizados para el desarrollo de nuestro estudio.

1.2.1 Aprendizaje por refuerzo (RL, reinforcement learning)

El aprendizaje por refuerzo (*Reinforcement Learning*) se inspiró en gran parte en la investigación sobre el comportamiento y psicología animal donde en las primeras investigaciones se demostró que, a través de prueba y error, los animales eran capaces de aprender a ejecutar un comportamiento que pudiera conducirles a algún resultado previsiblemente satisfactorio. Los algoritmos de RL utilizan mecanismos de aprendizaje muy similares al de los animales. El aprendizaje por refuerzo contribuye en la investigación en psicología y neurociencia y es, además, uno de los campos del aprendizaje automático cuya disciplina pertenece al área de la Inteligencia Artificial [7]. Se está aplicando, por ejemplo, para la creación de vehículos autónomos, el tratamiento de datos, donde estos pueden aplicarse para reducir el consumo energético o para generar políticas de recomendación de productos a clientes con lo que se consiga maximizar las ventas mediante las recomendaciones, la robótica y automatización de tareas en la industria, entre otros.

Consiste en un conjunto de algoritmos computacionales cuyo objetivo es aprender a tomar ciertas decisiones a través de la experiencia.
Es una técnica basada en la prueba y error en la que se recompensa la toma de decisiones correctas optimizando así el comportamiento del sistema.

El agente aprende a través de los mecanismos de exploración-explotación que utiliza para aumentar su eficacia ante la incertidumbre de saber qué opción debe escogerse para maximizar la recompensa. Esta metodología basada en el aprendizaje permite manejar la incertidumbre asociadas a los sistemas estocásticos o dinámicos de forma más eficiente.

Para ello, durante el proceso iterativo, en la fase de exploración, se irán explorando las diferentes opciones de ejecución ante los estados proporcionados por el entorno, en los que los valores de selección irán evolucionando a medida que va aprendiendo el agente.
En la fase de explotación el agente debe seleccionar aquella opción que considera óptima en el momento en el que se encuentra. El dilema de tener que lidiar entre la ejecución de la fase de explotación o de exploración puede hacer que esta alternancia llegue a ser compleja. Al optimizar la recompensa también lo hará el rendimiento del agente durante las fases de explotación y exploración de las diferentes opciones.

El funcionamiento del aprendizaje por refuerzo sería el siguiente: El agente interactúa con el entorno en una secuencia discreta de ensayos t= 0, 1, 2, 3..., en la que para cada una de las secuencias recibirá una representación del entorno, que llamaremos estado, y en función de esta el agente realizará una acción por cada uno de los ensayos. En cada una de estas secuencias el agente recibirá una señal de recompensa que evaluará la acción previa. Así pues, a lo largo de los distintos ensayos, el agente realizará una acción en la que acertará o errará. Si la acción realizada ha sido la correcta obtendrá recompensa, en caso contrario se le penaliza, por ejemplo, sin recompensa. El objetivo del agente es maximizar la recompensa total obtenida al final para conseguir que el agente actúe con el comportamiento esperado.

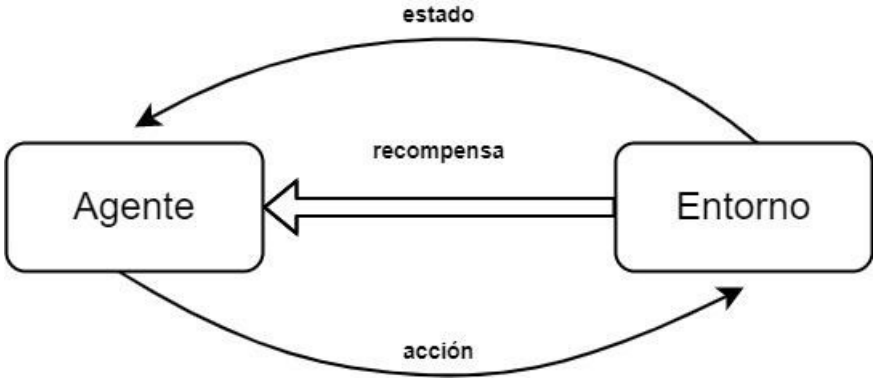

Figura 1. *Esquema de la interacción entre el agente y el entorno. El agente realiza una acción sobre el entorno y esto provoca su respuesta generando un estado y una recompensa.*

Por tanto, el agente es el algoritmo que aprende con el objetivo de emprender la mejor solución posible ante una situación dada. La toma de decisiones la realiza a partir de una función con unas entradas numéricas y uno o varios valores de retorno en el espacio de acciones posible. Podría tratarse, por ejemplo, de algo complejo como un robot con varios sensores con los que pudiera recoger información del entorno.
El entorno se trataría del ecosistema donde el agente realiza sus acciones y con el cual interactúa.

En este proyecto vamos a aplicar diferentes modelos de aprendizaje por refuerzo, mediante algoritmos, para comprobar cómo se comportan y cuál de ellos es capaz de obtener la máxima recompensa posible o reproducir de la mejor manera la respuesta de los sujetos (agentes biológicos como los humanos o animales).
Para el inicio de este proceso el agente requerirá de unos parámetros iniciales que se aplicarán al modelo escogido.
Cada uno de los modelos tiene diferentes opciones, donde para “matching-pennies” tenemos dos opciones (par e impar) y para “piedra, papel, tijeras” tendremos tres. Cada una de estas opciones presenta unas características en las que la correlación entre cada una de las características con la obtención (o no) de recompensa da lugar a un valor. Para cada ensayo se actualiza este valor, y si el agente ha obtenido recompensa, aumenta el valor de la opción escogida y disminuye el valor del resto de opciones. Si, por el contrario, no ha obtenido recompensa, disminuye el valor de la opción realizada y aumenta el valor del resto.

La elección de cada una de las opciones disponibles se realiza en base a la probabilidad, esta se obtiene a partir de una función lineal (exponencial normalizada, softmax) de la acumulación del valor en base a las características de cada una de las opciones. Esto evita el determinismo de escoger siempre la opción con el valor máximo, por lo que favorece la exploración sobre todo cuando la diferencia de valores no es grande, y tiene un fuerte fundamento con el comportamiento probabilístico de los agentes biológicos.
Una vez escogida una de las opciones se actualiza el valor. Este nuevo valor se origina en base a la respuesta obtenida tras elegir una de las opciones, es decir si hay recompensa o no, y proporcionalmente a la tasa de aprendizaje, que es un término que mide como de rápido se actualizan los valores en base a la respuesta obtenida (recompensa o no).
Finalmente, tras realizar todos los ensayos estimados obtendremos los resultados, ya sea, medias de probabilidades o recompensa del modelo

En el siguiente punto vamos a detallar los parámetros iniciales a aplicar al comienzo del proceso.

1.2.2 Parámetros Optimizables

A continuación, definimos las condiciones que debe cumplir nuestro sistema. Nuestro sistema se basa en un modelo de aprendizaje por refuerzo en el que los parámetros irán ajustándose según sean mejores o peores predictores de los resultados deseados. Para el cálculo de los algoritmos disponemos de unos parámetros optimizables como variables de entrada que irán cambiando a lo largo de la ejecución, estas se irán ajustando a través de un proceso de optimización basado en el algoritmo de optimización Bayesiano (Bayesian Adaptative Direct Search, BADS).

Los modelos de reinforcement learning se ajustan a los casos específicos donde se aplican en base a unos parámetros libres que se tienen que optimizar. Existen dos parámetros fundamentales: el *"learning rate"* (η) y la temperatura (T), y otros que dependen según los mecanismos que incluya el modelo, como en nuestro caso, el parámetro que determina el cambio de ritmo de *"learning rate"* (μ) y los parámetros que determinan el decaimento del valor (wdec y ewdec).

A continuación, describimos los parámetros utilizados:

η_0: Parámetro inicial del *"learning rate"* o tasa de aprendizaje.

μ: Parámetro que determina el cambio de ritmo del *"learning rate"*. Su inversa es como una constante de tiempo, cuya unidad es el trial, que separa las correlaciones locales de las sistemáticas. En los extremos, cuando μ es igual a 1, todas las correlaciones se consideran sistemáticas, para cada trial se sobrescribe el *"learning rate"* ya sea creciente o decreciente. En cambio, cuando μ es igual a cero, ninguna correlación es suficientemente sistemática y en este caso el *"learning rate"* no se actualiza.
Este parámetro se tiene que ajustar en función de la volatilidad del entorno.

T: Este parámetro se llama temperatura, en alusión a la distribución de Boltzmann. Define el grado de aleatoriedad con el que el modelo ejecuta una predicción. Para valores altos de T habrá mayor arbitrariedad y más equiprobables serán las posibilidades, mientras que para valores bajos los valores de asociación mayores serán más probables que el resto.
Podemos ver su funcionalidad cuando usamos la función softmax. Se trata de una función para pasar de valores a probabilidades y se aplica a todas las opciones posibles. Se utiliza para tomar una elección entre posibles opciones en función de su probabilidad de recompensa asociada.

Recordemos que las propiedades de las probabilidades son tales que tienen un valor positivo y menor o igual a 1 y la suma total de las probabilidades es igual 1.

La probabilidad P se obtiene a través de la formula:

$$P_i = \frac{e^{V_i/T}}{\sum_{j=1}^{N} e^{V_j/T}} \quad (1)$$

donde j representa cada una de las características del valor V que van asociadas a una opción i y T es el parámetro de temperatura. En función de la T la diferencia de valor entre las diferentes opciones se amplifica o se hace más pequeña.
Si tenemos dos V_J con magnitudes distintas, por ejemplo $V_J = 0.2$ y $V_J = 0.8$, para un alto T (cercano a 1), tendremos unas probabilidades más equiprobables. Así, si T= 1, tenemos:

$$P_1 = \frac{e^{0.2/1}}{e^{0.2/1} + e^{0.8/1}} = 0.35 \qquad P_2 = \frac{e^{0.8/1}}{e^{0.2/1} + e^{0.8/1}} = 0.64$$

Por otro lado, para un valor bajo de T (cercano a 0), tendremos unos valores más desiguales donde el valor más alto tendrá más probabilidad. Así, por ejemplo si T= 0.1, tenemos:

$$P_1 = \frac{e^{0.2/0.1}}{e^{0.2/0.1} + e^{0.8/0.1}} = 0.00247 \qquad P_2 = \frac{e^{0.8/0.1}}{e^{0.2/0.1} + e^{0.8/0.1}} = 0.997$$

La elección de la opción a escoger es estocástica en base a estas probabilidades.

wdec: Es un parámetro de decaimiento del valor, cuando éste no está asociado con la opción escogida y se ha obtenido recompensa.

ewdec: Es un parámetro de decaimiento del valor, cuando éste está asociado con la opción escogida y no se ha obtenido recompensa.

1.2.3 Modelos de condicionamiento clásico

El condicionamiento clásico es un tipo de aprendizaje por el cual un estímulo nuevo, que no provoca respuesta, llega a poder provocarla gracias a la conexión asociativa entre este estímulo nuevo con un reflejo ya existente que sí la provoca. Nace como consecuencia de los estudios de Pavlov, el más conocido de ellos es el reflejo de salivación en perros, donde empezó a aplicar estímulos auditivos y visuales antes de servirle la comida al perro, y tras varios intentos, observó que el animal salivaba ya que había asociado estos estímulos con la comida [8].

Los diferentes estímulos se clasifican en estímulos incondicionales y condicionales.
Un estímulo incondicional (EI) es el que genera una respuesta incondicionada, en este caso sería la comida ante la cual se produce salivación.
Un estímulo condicional (EC) es aquel que por sí solo no genera respuesta, pero mediante la asociación con un estímulo incondicional puede llegar a generarla por sí solo. En el caso de los perros de Pavlov, los estímulos auditivos representarían un estímulo condicional que al presentarse varias veces con la comida (estímulo incondicional) consigue que el animal asocie el sonido con la comida, y que solo al escuchar el sonido se provoque una respuesta (salivación).

Posteriormente y a raíz de la teoría de bloqueo de Kamin, comenzó a ponerse en tela de juicio la importancia de contigüidad entre estímulos como condición indispensable para que se produjese el aprendizaje de una asociación pavloviana. Esta teoría consistía en que si en un ensayo, donde ya se ha realizado la asociación con el estímulo, le añadimos un nuevo estímulo en una fase posterior, este nuevo estímulo no aporta ya ningún tipo de

información novedosa y no se está realizando ningún tipo de aprendizaje sobre este nuevo estímulo. Según Kamin para que se realice un aprendizaje la aparición del estímulo debe ser sorpresiva [8].

A partir de estos conceptos es de donde parten los diferentes modelos de condicionamiento clásico sobre los que hemos basado los algoritmos usados en el estudio.

1.2.3.1 Modelo No Adaptativo (Rescorla-Wagner)

Se trata del modelo de condicionamiento clásico más conocido y a través del cual son explicados la mayoría de los fenómenos y procesos básicos del condicionamiento clásico.

Este modelo se basa en la idea de la sorpresa desarrollada por Kamin, en la que los sujetos aprenden sobre un estímulo condicionado cuando el estímulo incondicional es inesperado [9]. El sujeto aprende de las discrepancias entre lo que espera que suceda y lo que sucede realmente y considera que el aprendizaje se detiene cuando deja de ser sorpresivo el estímulo incondicional.

Partiendo de la teoría del bloqueo de Kamin, Rescorla-Wagner propuso que los cambios del valor asociado a un estímulo concreto dependían del total del valor que ya poseía el estímulo en el ensayo. Estos valores asociados a un estímulo u opción pueden referirse por ejemplo, en la elección entre dos objetos, que uno esté a la izquierda y el otro a la derecha, uno sea circular y el otro triangular, uno tenga color rojo y el otro azul… Todas estas características pueden tener un valor asociado en el contexto de una tarea que determine finalmente que objeto es mejor escoger para obtener una recompensa.

La ecuación básica del valor asociado a un estímulo para un ensayo es:

$$\Delta V^n = \eta\,(R - V^{n-1}) \quad (\mathbf{2})$$

donde ΔV es el incremento del valor de una opción en el ensayo n. V^{n-1} es el valor de la opción en el ensayo anterior. R es la asíntota del valor, representa el estado de realidad, de manera que si se ha presentado el estímulo incondicional, R vale 1, y si no se ha presentado, R vale 0. La diferencia (R - V) representa la diferencia entre realidad y expectativa (sorpresa), es decir, entre lo que podemos llegar a aprender y lo que hemos aprendido, η es la tasa de aprendizaje al que nos referiremos como *"learning rate"*.

Al principio del aprendizaje V estará cercano a 0, ya que aún no se ha producido la asociación del estímulo condicionado con el estímulo incondicional, mientras que la diferencia (R - V) será grande ya que habrá sorpresa y por tanto el incremento del valor asociado al estímulo ΔV será mayor. Según vayamos realizando ensayos V irá creciendo ya que aumentará su asociabilidad con el EI, mientras que (R - V) será cada vez menor al ir perdiendo la sorpresa y su ΔV también disminuirá.

Este modelo lo denotaremos en el contexto de meta-aprendizaje con "*learning rates"* dinámicos como "No adaptativo" puesto que el "*learning rate*" en este caso es fijo.

En nuestro caso hemos adaptado la fórmula de la siguiente manera:

$$\Delta V_i^n = V_i^{n-1} + R\,\eta_i\left(Opción_{elegida}(R - V_i^{n-1}) - wdec \cdot Opción_{noelegida} \cdot V_i^{n-1}\right) + (1-R)\,\eta_i\,(Opción_{noelegida} \cdot \left((1-R) - V_i^{n-1}\right) - ewdec \cdot\ Opción_{elegida} \cdot V_i^{n-1}) \quad (\mathbf{3})$$

Cada uno de los estímulos i tendrá su propio valor asociado siendo ΔV^n = $[\Delta V_{i1}^n, \Delta V_{i2}^n, \ldots]$.

Por tanto, el incremento del valor asociado, ΔV^n, será diferente para cada uno de los estímulos i. Para la variable $Opción_{elegida}$ el valor de la opción que hemos escogido será 1 y para el resto 0, y al contrario para la variable $Opción_{noelegida}$. Por ejemplo si tenemos dos estímulos y hemos escogido la primera opción (estímulo i1) el valor de la variable $Opción_{elegida}$ será [1, 0] y el de $Opción_{noelegida}$ será [0, 1]. Como hemos indicado anteriormente la variable R valdrá 1 en los casos en los que se obtiene el estímulo incondicional, a esto lo llamaremos recompensa. Cuando no obtiene este estímulo, el valor de R será 0 y diremos que no ha obtenido recompensa. De esta forma, para los estímulos i que obtienen recompensa se producirá un aumento de ΔV^n.

Si obtiene recompensa el estímulo elegido:

$$\Delta V_i^n = V_i^{n-1} + R\,\eta_i\left(R - V_i^{n-1}\right) \qquad (\mathbf{4})$$

donde R = 1 resultando:

$$\Delta V_i^n = V_i^{n-1} + \eta_i\left(1 - V_i^{n-1}\right) \qquad (\mathbf{5})$$

En este caso se produce un aumento de $1 - V_i^{n-1}$, es decir, de la diferencia entre lo obtenido (señal de recompensa) y lo esperado (valor asociado del estímulo i en el ensayo anterior n-1) por su "*learning rate*" η_i. A mayor η, mayor será la velocidad de aprendizaje y el incremento de ΔV_i^n será mayor.

Si no obtenemos recompensa para los estímulos no elegidos, R = 0, se producirá también el mismo aumento de ΔV_i^n:

$$\Delta V_i^n = V_i^{n-1} + (1 - R)\,\eta_i\left((1 - R) - V_i^{n-1}\right) \qquad (\mathbf{6})$$

Como R = 0, resulta:

$$\Delta V_i^n = V_i^{n-1} + \eta_i\left(1 - V_i^{n-1}\right)$$

obteniendo de nuevo la misma fórmula (**5**) que en el caso anterior.

En los estímulos i donde no se obtiene la recompensa, la fórmula resultante si hemos elegido el estímulo que no obtiene la recompensa a partir de (3) es:

$$\Delta V_i^n = V_i^{n-1} + (1 - R)\,\eta_i(-ewdec\ V_i^{n-1}) \qquad (\mathbf{7})$$

Aplicando R = 0:

$$\Delta V_i^n = V_i^{n-1} - \eta_i\,(ewdec\ V_i^{n-1}) \qquad (\mathbf{8})$$

Si hemos elegido el estímulo con recompensa, R = 1, para los estímulos no elegidos la formula resultante será:

$$\Delta V_i^n = V_i^{n-1} - \eta_i\,(wdec\ V_i^{n-1}) \qquad (\mathbf{9})$$

Por tanto, vemos que para los estímulos que no se obtiene recompensa se disminuye el valor asociado. Por los parámetros ewdec y wdec asignamos una penalización a estos estímulos que no han obtenido recompensa y ajustamos ΔV_i^n.

En este modelo el valor de η es constante y no se adapta al aprendizaje del sujeto.

A continuación, ilustramos con un ejemplo, cómo se calcula el valor asociado a un estímulo y como este depende de la recompensa. Estas se calculan a partir de las elecciones y recompensas que ha tenido el sujeto, lo cual hace que el modelo se adecúe a su comportamiento. Los valores de entrada del modelo serán los valores iniciales de los parámetros η_0, μ, T, wdec y ewdec previamente optimizados por BADS.

Algoritmo: Cálculo del valor en el modelo "No adaptativo" (Rescorla-Wagner) .Extracto de la maximización de la probabilidad de realizar la misma selección que el sujeto.

1. Inicio.
2. Inicializar $\eta = \eta_0$ para ambos estímulos.
3. Inicializar V = 0.5 para ambos estímulos.
4. Hacer ntrials = 200.
5. Desde ind_trials = 1 hasta ind_trials = ntrials
6. Selección basada en el valor: valOpt = Valor inicial o valor ronda anterior (V).
7. Calculamos las probabilidades de escoger cada una de las opciones:
   pOpt = exp(valOpt/T).
   pOpt = pOpt/sum(pOpt).
8. Guardamos en opción_elegida la elección del sujeto guardada en la variable choice: opción_elegida = choice(ind_trials).
9. Guardamos en opción_no_elegida la opción no elegida del sujeto: opción_no_elegida = ~opción_elegida.
10. Almacenamos en Outcome el valor de la recompensa del sujeto reward. (1 si gana, 0 si pierde): outcome = reward(ind_trials).
11. Calculamos el valor de asociación en función del valor obtenido en la ronda anterior y la recompensa:
    V= V + R η(opción_elegida(R -V) – wdec opción_no_elegida V) + (1- R) η(opción_no_elegida((1- R)- V) - ewdec· opción_elegida ·V).
12. Fin del bucle Desde.

Este modelo presenta una serie de limitaciones ya que no incluye conceptos como el de la inhibición latente o retraso en el aprendizaje debido a estímulos neutros preexpuestos para el sujeto. Por eso se desarrollaron nuevos modelos más completos.

1.2.3.2 Modelo en función de la Relevancia (Mackintosh)

El modelo de Mackintosh [10] nace como un modelo que incorpora el concepto de inhibición latente no incluido en el modelo anterior de Rescorla-Wagner.
La inhibición latente explica que el aprendizaje en el que un estímulo neutro se convierte en condicional tarda más en adquirirse para un sujeto en el que ese estimulo neutro es previamente expuesto en solitario y, por tanto, esté ya familiarizado con él, frente a un sujeto en el que ese estímulo neutro se presente por primera vez acompañado de un estímulo incondicional. Por ejemplo, si en una primera fase, a un grupo A mostráramos un pitido y, en una segunda fase, para el grupo A y B, al pitido lo acompañáramos de comida, el grupo B asociaría el pitido a la comida más rápidamente que el grupo A [10].

Este concepto no era considerado por Rescorla-Wagner, sin embargo, una de las teorías de Mackintosh es que la asociabilidad de un estímulo no viene determinada solo por sus características físicas, como su intensidad o modalidad, sino que puede variar a través de la experiencia del sujeto con tal estímulo. Esta experiencia puede haber ocurrido antes del experimento o durante. Pero el caso más interesante, es la consecuencia de la correlación entre el estímulo y el refuerzo. Si las variaciones de un estímulo se correlacionan con cambios en el refuerzo, el valor de la asociabilidad puede aumentar, mientras que si no se correlacionan con el refuerzo esta puede disminuir.
En nuestro estudio más que por las variaciones del estímulo, la tasa de aprendizaje η

vendrá determinada por la toma de decisiones y su dinámica, si ha obtenido o no recompensa y varía en función de la asociabilidad.

Mackintosh afirma que, si un estímulo predice el reforzador mejor que el resto de los estímulos presentes, su tasa de aprendizaje (η) aumentará. Esto no quiere decir que la tasa de aprendizaje de los demás estímulos tenga que disminuir, ya que, el valor de η es específico e independiente de cada estímulo. Este valor de η es determinado por las características físicas del estímulo y por la percepción del sujeto que lo recibe, y puede cambiar con la experiencia. Si el estímulo A se correlaciona con cambios en el refuerzo η_A aumentará, y si el estímulo B no se correlaciona con el refuerzo η_B disminuirá.

Mackintosh también decía que los sujetos aprenden a atender e ignorar los estímulos en la medida en que estos predicen con éxito el resultado. Basándonos en esta teoría, implementamos el modelo de "Relevancia", donde calculamos el incremento de la asociabilidad a partir de (**3**), pero en este modelo η es dinámica y debemos calcularla en función de la relevancia del estímulo, así η aumentará para los estímulos que sean buenos predictores. Calculamos la relevancia a partir de:

$$Relevancia = R\ (1 - (R - V_i^{n-1})) \quad (\mathbf{10})$$

Aquellos valores que no predigan el resultado y, por tanto, no obtendrán recompensa (R = 0), su relevancia será cero. En cambio, en aquellos que han predicho el resultado (R = 1), su relevancia será igual al total de los valores asociados al estímulo i hasta el ensayo n-1. A partir de la relevancia actualizaremos el valor de la tasa de aprendizaje η:

$$\eta_i^n = \eta_i^{n-1} + \mu\ (\ Relevancia - \ \eta_i^{n-1}) \ (\mathbf{11})$$

donde μ es un parámetro, previamente calculado por BADS, que determina el cambio de ritmo de η y η_i^{n-1} es el "*learning rate*" hasta el ensayo n-1 del estímulo i. Para aquellos estímulos que son buenos predictores η aumenta en función de la discrepancia entre la Relevancia y η_i^{n-1} .Para aquellos estímulos que no tienen Relevancia, μ disminuye.

A continuación, vemos un ejemplo de cómo implementamos este modelo:

Algoritmo: Cálculo del valor en el modelo "Relevancia" (Mackintosh). Extracto de la maximización de la recompensa contra un agente dinámico.

---

1. Inicio. Ponemos valores iniciales a los parámetros η_0, μ, T, wdec, ewdec previamente optimizados.
2. Inicializar η = η_0 para ambos estímulos.
3. Inicializar V = 0.5 para ambos estímulos.
4. Hacer ntrials = 200.
5. Desde ind_trials = 1 hasta ind_trials = ntrials
6. Llamamos a la función matching_pennies que emulará a un competidor y en función de la anterior elección del sujeto y su recompensa devolverá el parámetro computerChoice para minimizar la recompensa de su adversario: [computerChoice, pComputerRight, biasInfo] = matching_pennies(data, 4, 0.05);
7. Selección basada en el valor: valOpt = Valor inicial o valor ronda anterior (V);
8. Calculamos las probabilidades de escoger cada una de las opciones: pOpt = exp(valOpt/T). pOpt = pOpt/sum(pOpt).
9. Realizamos la elección con la función SOFTMAX que maximiza la probabilidad de escoger la acción de mayor recompensa: opt_chosen(ind_trials) = find(cumsum(pOpt) >= rand, 1).
10. Guardamos en opción_elegida la elección del sujeto: opción_elegida = opt_chosen(ind_trials).
11. Guardamos en opción_no_elegida la opción no elegida del sujeto:

opción_no_elegida = ~opción_elegida.

12. Guardamos en outcome la elección del sujeto:
    outcome=(opt_chosen(ind_trials).
13. Si outcome == computerChoice entonces
        R = 1
    Sino
        R = 0
    Fin-Si
14. Calculamos el valor de asociación respecto de la recompensa:
    V= V + R η(opción_elegida(R -V) - wdec·opción_no_elegida·V) + (1- R) η(opción_no_elegida ((1- R)- V) - ewdec·opción_elegida·V).
15. Calculamos en la variable Relevancia la diferencia entre el valor y la recompensa. Para Mackintosh será: Relevancia = R(1 - abs(R -V)).
16. Calculamos la tasa de aprendizaje:
    η = η + μ(Relevancia - η).
17. En el caso de este modelo el "*learning rate*" aumenta para los estímulos elegidos que consiguen obtener recompensa (ganan). Para aquellos estímulos que no consiguen recompensa (pierden), la variable Relevancia será 0 y eta disminuirá.
18. Fin del bucle Desde.

1.2.3.3 Modelo en función de la Sorpresa (Pierce-Hall)

Pierce y Hall presentan un modelo similar al de Mackintosh, pero mientras que para Mackintosh la asociabilidad de un estímulo y su velocidad de aprendizaje aumentaban si eran buenos predictores, en el modelo Pearce-Hall, en cambio, su velocidad de aprendizaje será mayor cuanto mayor sea la sorpresa causada al presentarse el estímulo. Lo explicaron mediante los resultados obtenidos en un experimento en el que se presentaba un tono asociado con un choque eléctrico de baja intensidad y poca duración y en una siguiente fase se presentaba ese mismo tono asociado con un choque de intensidad y duración alta, se observó que en la segunda fase el grupo experimental mostraba un retraso en el condicionamiento y que a medida que el estímulo era predicho por el sujeto, era menos sorprendente y disminuía su tasa de aprendizaje [10].

El modelo de Pearce-Hall lo hemos implantado bajo el nombre de "Sorpresa", puede plantearse formalmente por la ecuación:

$$Sorpresa = \left| R^{n-1} - V_i^{n-1} \right| \qquad (12)$$

donde "Sorpresa" representa la diferencia entre la intensidad del estímulo incondicional (R^{n-1}) y el valor del estímulo i (V_i^{n-1}) en el ensayo previo.
La tasa de aprendizaje, η_i^n, de un estímulo i en un ensayo n se calcula igual que en (11), pero esta vez en función de la sorpresa.

$$\eta_i^n = \eta_i^{n-1} + \mu\,(Sorpresa - \eta_i^{n-1}) \qquad (13)$$

Esta fórmula muestra que, cuando el valor ($R - V_i$) es elevado, el sujeto se sorprende y presta más atención al estímulo condicional, y por tanto la tasa de aprendizaje, que va en función de esta "sorpresa", aumenta. Si la sorpresa es pequeña la tasa de aprendizaje aumentará poco o disminuirá.

A continuación, mostramos como se calcula el algoritmo en detalle:

Algoritmo: Cálculo del valor en el modelo "Sorpresa" (Pearce-Hall)

---

1. Calculamos el valor de asociación respecto de la recompensa:

V= V + $R\,\eta$ (opción_elegida (R -V) - wdec·opción_no_elegida·V) + (1- R) η(opción_no_elegida((1- R)- V) - ewdec·opción_elegida·V).

2. Calculamos en la variable Sorpresa la diferencia entre el valor y la recompensa. Para Pearce-Hall será: Sorpresa = | $R - \mathrm{V}$ |.
3. Calculamos la tasa de aprendizaje:
   η = η + μ (Sorpresa - η).

1.2.3.4 Modelo en función de la Información

El modelo "Información" es un modelo creado por el investigador y profesor Salvador Ardid en la *Universitat Politècnica de València* en el marco del grupo de investigación *Lab of Natural and Designed Intelligence* [11].
Este modelo prioriza la información que se recibe de las variables, de ahí su nombre.
En este caso, la tasa de aprendizaje aumenta cuando la magnitud de la correlación aumenta, independientemente de su signo, ya que, tan informativo es predecir la recompensa como la ausencia de esta.

Los valores asociados a un estímulo varían entre 0 y 1. Cuando los valores se acercan a 0.5 no son significativos estadísticamente, esto genera mayor incertidumbre e implica una disminución de la tasa de aprendizaje. Hay un mapeo 1:1 cuando se hace el escalado de [0, 1] a términos de correlación a los valores [-1, +1], donde el punto medio 0.5 pasa a ser 0 (no tiene correlación).

A continuación, mostramos como se calcula el algoritmo, donde la variable "Información" se introduce para reducir la incertidumbre que se produce cuando no hay una correlación sistemática entre un valor y el hecho de que haya finalmente recompensa:

$$Informacion = \left| 2\,(V_i^{n-1} - 0.5) \right| \qquad (\mathbf{14})$$

Cuando los valores se alejan de 0.5, implica obtener más información. Si los valores son altos, se acercan a 1, quiere decir que la correlación entre lo que se esperaba y lo que se ha obtenido es positiva (+1). Si los valores son bajos, se acercan a 0, la correlación es negativa, ha habido sorpresa porque no ha obtenido lo que esperaba (-1). En ambos casos, la tasa de aprendizaje aumenta cuando los valores se alejan de 0.5.

Si no hay una correlación sistemática se penaliza la tasa de aprendizaje (15) ya que ese valor no es relevante para la decisión entre lo que se desea y se correlaciona positivamente ni para aquella decisión que tratamos de evitar y se correlaciona negativamente.

$$\eta_i^n = \eta_i^{n-1} + \mu\,(Información - \eta_i^{n-1}) \qquad (\mathbf{15})$$

Así priorizamos aquellas características que son más predictivas y aportan información relevante para la toma de decisión y reducimos aquello que no es sistemático y no aporta información relevante (ruido).

Algoritmo: Cálculo del valor en el modelo "Información"

1. Calculamos el valor de asociación respecto de la recompensa:
   V= V + $R\,\eta$(opción_elegida(R -V) - wdec·opción_no_elegida·V) + (1- R) η(opción_no_elegida((1- R)- V) - ewdec·opción_elegida·V).
2. Calculamos la variable Información: Información = | $2\,(V - 0.5)$ |
3. Calculamos la tasa de aprendizaje:
   η = η + μ (Información - η).

Utilizaremos la variable “Información” para actualizar la tasa de aprendizaje. Si los valores se alejan de 0.5, esto implica más información, por tanto, la variable aumenta y lo hace también el *“learning rate”*. Si los valores se acercan a 0.5, esto implica mayor incertidumbre, y por ello, “Información” disminuye y lo hace también el *“learning rate”*.

1.2.3.5 Modelo en función de la Incertidumbre

Este modelo llamado “Incertidumbre” se trata de un modelo para probar como es el comportamiento en un escenario diferente del modelo “Información”. Mientras que en el modelo “Información” sube el *“learning rate”* cuando los valores se alejan de 0.5, en este modelo aumentamos el *“learning rate”* cuando los valores fluctúan alrededor de 0.5 para poder capturar los cambios de manera más rápida.
En este modelo se busca el componente sorpresivo. Si “Información” era una generalización del modelo de “Relevancia” por tener en cuenta, no solo la relevancia, si no todo aquello que aporta información, en este modelo lo invertimos de manera que el modelo acelere el aprendizaje al máximo cuando la incertidumbre es más grande, bajo esta concepción lo que estamos haciendo es generalizar el modelo “Sorpresa”. A este modelo lo hemos llamado ”Incertidumbre” y empleamos la siguiente fórmula para calcular su variable:

$$Incertidumbre = \ 1 - \left| 2\ (V_i^{n-1} - 0.5) \right| \qquad (16)$$

A continuación, mostramos como se calcula en el algoritmo:

Algoritmo: Cálculo del valor en el modelo “Incertidumbre”

1. Calculamos el valor de asociación respecto de la recompensa:
   V= V + $R\ \eta$(opción_elegida(R -V) - wdec·opción_no_elegida·V) + (1- R) η(opción_no_elegida((1- R)- V) - ewdec·opción_elegida·V).
2. Calculamos la variable Incertidumbre: Incertidumbre = $1 - |\ 2\ (V - 0.5)\ |$
3. Calculamos la tasa de aprendizaje:
   $\eta = \eta + \mu$ (Incertidumbre - η).

Si los valores se acercan a 0.5, su incertidumbre será cercana a 1 y la tasa de aprendizaje aumentará. En este caso si el *“learning rate”* es bajo el aumento será mayor.
Si los valores se alejan de 0.5, su incertidumbre será cercana a 0 y la tasa de aprendizaje disminuirá. Si el *“learning rate”* es alto disminuirá mucho y si es bajo su disminución será muy poca.

1.3 Teoría de juegos

La teoría de juegos nace a partir del libro de Von Neumann y Morgenstern *The Theory of Games and Economic Behaviour,* publicado en 1944, utilizado inicialmente para comprender el comportamiento de la economía. Se trata de un área de la matemática aplicada que se utiliza para estudiar las relaciones humanas a través de las interacciones y tomas de decisiones llevadas a cabo en estructuras formalizadas de incentivos, lo que conocemos formalmente como “juegos” [12].

La teoría de juegos analiza las situaciones de las relaciones estratégicas. Consiste en tratar de predecir la decisión o estrategia que vaya a seguir el oponente y en función a esto ajustar nuestra propia estrategia. Un ejemplo de esto podría ser el caso siguiente. Imaginemos que tres amigos llamados Lucía, Pablo y José quieren hacer una escapada y no tienen decidido si quieren disfrutar de un fin de semana en el campo o ir a la playa, por

otra parte, no están seguros de hacerlo esa semana o posponerlo. En primera instancia votarán para saber si el destino del viaje será el campo o la playa y en una segunda votación elegirán si se realiza una de estas actividades o se pospone para otra semana. Por tanto, existirán tres posibles resultados, ir al campo, a la playa o posponer la salida [12].

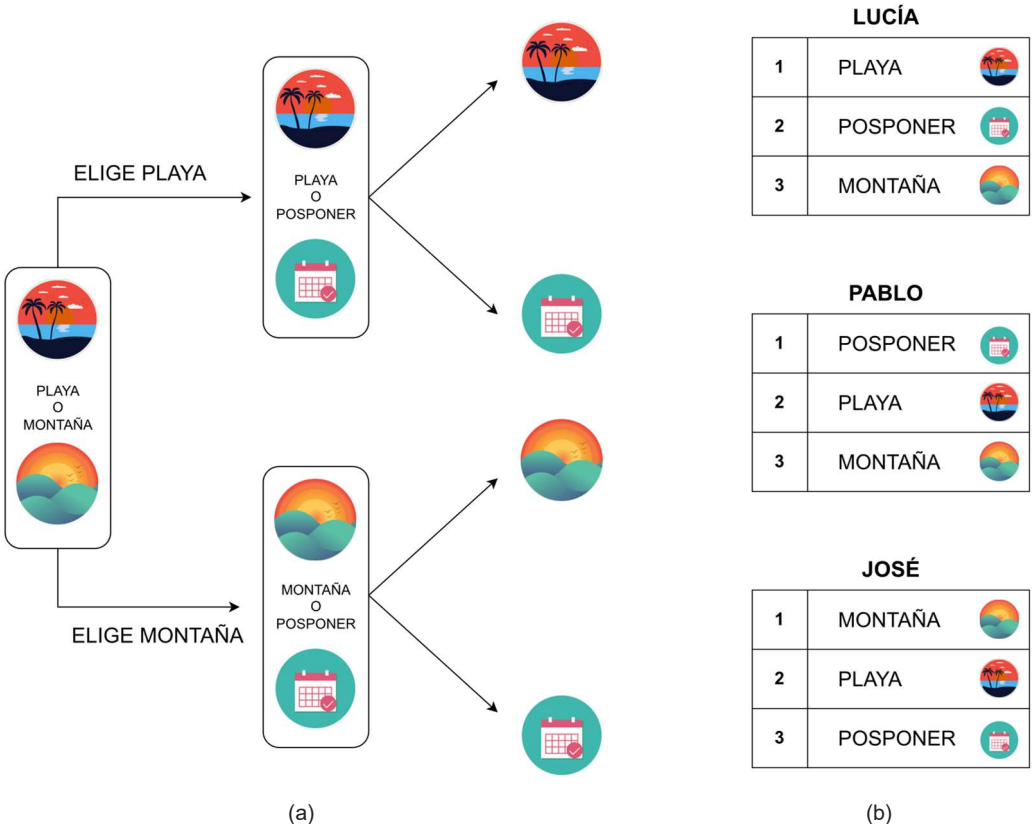

Figura 2. *Votaciones estratégicas. La figura (a) se muestra el diagrama que representa el orden de votación. En la figura (b) tenemos las tablas con la intención de voto de cada uno de los sujetos [12].*

En la figura 2(a) podemos ver el orden en el que se realizarán las votaciones, mientras que en la figura 2(b) muestra las preferencias de selección de cada uno de los amigos.
Si las votaciones se realizaran según el orden de preferencia ganaría la opción de ir a la playa, ya que, en la primera votación entre playa o montaña Lucía elegiría playa, mientras que José elegiría montaña, y Pablo puesto que su primera opción no está disponible, pasaría a su segunda preferencia que es playa. En una segunda votación, entre ir a la playa o posponerlo, ganaría ir a la playa, ya que esta es la opción preferida de José entre las dos elecciones disponibles. Por tanto, ganaría la preferencia de Lucía. Sin embargo, si Pablo pensara estratégicamente, vería que le conviene votar a favor de la montaña, ya que en la segunda opción ganaría que se pospusiera el viaje. Por tanto, en la primera votación debería votar montaña que es la opción que menos le gusta de todas.
Si al final se elige posponer el viaje, más tarde, se volvería a votar desde el inicio.

Como podemos ver a la hora de tomar una decisión se tiene en cuenta la estrategia que pueden tomar el resto de las personas para decidir que opción escoger.

En un entorno social la toma de decisiones tiene dos características distintivas. Por una parte, los seres humanos y otros animales modifican su comportamiento en base a los cambios que se producen en el entorno, estas elecciones son difíciles de predecir debido a los múltiples factores que pueden influir en el resultado de la decisión. Por otro lado, esta decisión puede venir condicionada con el objetivo de provocar una consecuencia positiva o negativa para otros individuos. A partir de la teoría de juegos se han realizado diversos estudios neurobiológicos para investigar la base neuronal de la toma de

decisiones, los cuales sugieren que estas podrían surgir a partir de las zonas del cerebro encargadas de la evaluación de recompensas y el aprendizaje por refuerzo [13]. En este trabajo se propone realizar la optimización de algoritmos de refuerzo computacional aplicado a dos juegos de estrategia enmarcados en la teoría de juegos.

1.3.1 Matching Pennies (Centavos a juego)

Es un juego que sirve como ejemplo básico en la teoría de juegos. Se compone de dos jugadores en los que uno de ellos debe tratar de elegir la misma opción que el contrario y el otro debe elegir la opción distinta. Originalmente se jugaba con centavos, donde ambos jugadores lanzaban la moneda y si ambas salían iguales el jugador A se llevaba la moneda del jugador B, por el contrario, si salían diferentes el jugador A perdía su moneda dándosela al jugador B. Era un juego en el que había que maximizar la ganancia.

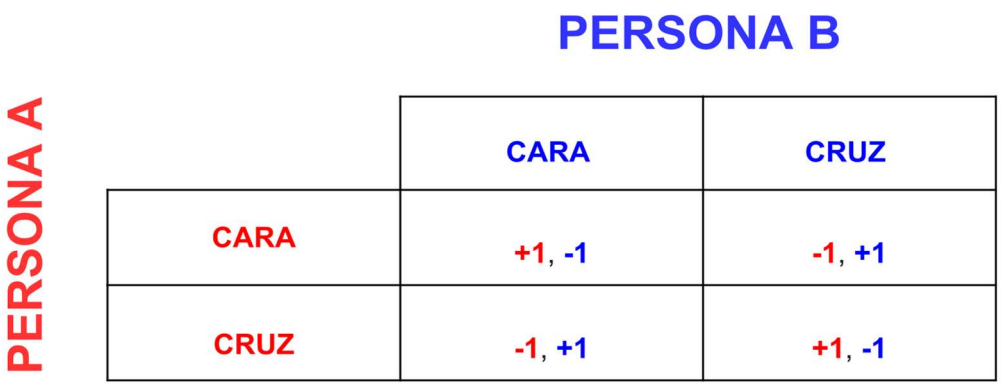

Figura 3. *Representación del juego Matching Pennies. Dos jugadores lanzan una moneda cada uno, si ambas coinciden, cara o cruz, la persona A gana y pierde la persona B. Por el contrario, si las monedas no coinciden ganará la persona B.*

En este primer estudio trataremos de acercarnos a la capacidad de tomar decisiones que tienen los seres humanos en el juego de "matching pennies" a través de la implementación de distintos algoritmos.

1.3.2 Rock-Paper-Scissors (Piedra, Papel, Tijeras)

Se trata de un juego de manos popular en el que los participantes muestran a la vez uno de los tres elementos posibles: piedra, papel o tijeras. La piedra se representa con el puño, el papel con la palma de la mano extendida y las tijeras haciendo una "V" con los dedos. El objetivo es vencer al oponente mostrando el elemento que gane al del contrario. Las reglas son las siguientes:

- El papel gana a la piedra porque la lanza.
- La piedra gana a la tijera porque la rompe.
- La tijera gana a el papel porque lo corta.

En caso de sacar el mismo elemento se considera empate y se vuelve a jugar de nuevo.

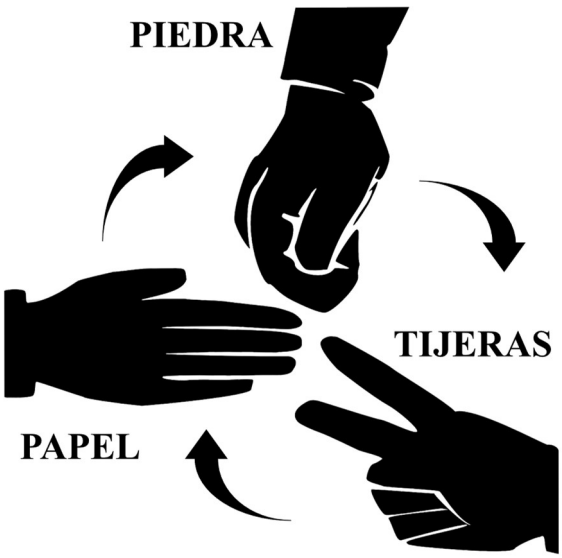

Figura 4. *Representación del juego Piedra, Papel, Tijeras. El papel gana a la piedra, la piedra a las tijeras y las tijeras gana al papel.[14]*

2. Estudio del comportamiento humano a través del juego Matching Pennies

A continuación, abordaremos las cuestiones referidas al estudio de la toma de decisiones de los seres humanos mediante refuerzo computacional a través del juego “matching pennies”.

2.1 Estudio de necesidades

Como hemos explicado, la intención original del TFG es doble, por una parte, optimizaremos el comportamiento de los agentes de RL compitiendo contra otros algoritmos computacionales y por otro lado se pretende mediante estos algoritmos entender mejor la toma de decisiones que tenemos los seres humanos en este tipo de juegos.

2.1.1 Consideraciones

Para poder analizar el comportamiento humano en un juego de “matching pennies” hemos obtenido, a través de una colaboración, los datos de un estudio sobre el aprendizaje en la toma de decisiones realizado en la escuela de medicina de la Universidad de Yale en New Haven, Connecticut. En este estudio se reclutaron 30 participantes entre 18 y 40 años que tenían que enfrentarse a un agente computacional en la tarea “matching pennies”.

El estudio tenía 45 minutos de duración y consistía en una prueba de 200 ensayos, dividida en dos bloques de 100 ensayos con una pausa en medio para descansar. En cada ensayo debían elegir uno de los dos rectángulos que aparecían en la pantalla (izquierda o derecha) y si se elegía el mismo que la computadora ganaban puntos, en caso contrario no obtenían recompensa. Tenían tan solo un segundo de tiempo para escoger su opción, tras cada elección obtenían la información de si habían obtenido recompensa o no.

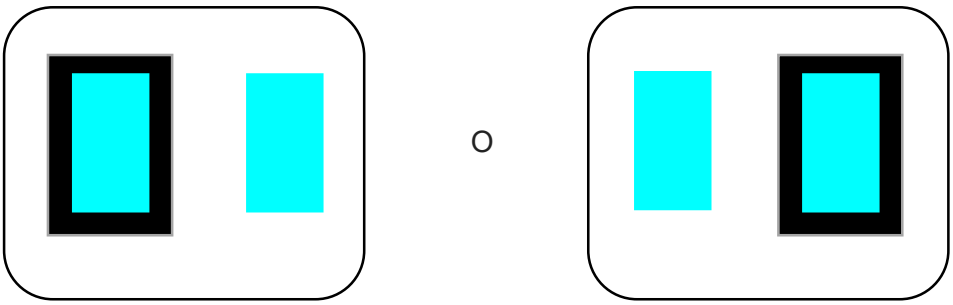

Figura 5. *Diapositiva utilizada en la explicación de la prueba con las dos posibles opciones a elegir.*

Por acudir al estudio ganaban 10$ de recompensa base y según la recompensa obtenida en el ensayo podían llegar a ganar entre 1 y 36$ adicionales.

2.2 Planteamientos de alternativas

A continuación, vamos a describir los lenguajes de programación más habituales en el desarrollo de proyectos de "machine learning" y por cual nos hemos decantado para nuestros estudios.

2.2.1 Lenguajes de programación

Matlab

Es una plataforma de programación que destaca por sus prestaciones de cálculo numérico, representación de datos y funciones, implementación de algoritmos y análisis de datos. Utiliza un lenguaje propio de programación, llamado M, que puede ejecutarse tanto desde el propio entorno como a través de archivos script con la extensión *.m. Las funciones de MATLAB se pueden ampliar mediante paquetes adicionales llamados toolboxes (caja de herramientas) orientadas a una actividad específica, como, por ejemplo, de aplicación matemática general, procesamiento de señales, inteligencia artificial o estadística. Se trata de un software privativo perteneciente a la empresa Mathworks.

GNU Octave

Es un lenguaje de alto nivel que se utiliza para realizar cálculos numéricos. Se distribuye como software libre y es de código abierto. Su lenguaje es en su mayoría compatible con MATLAB aunque su intérprete es más lento.

Python

Es un lenguaje de programación utilizado para cualquier tipo de aplicaciones. Su lenguaje de alto nivel presenta un código claro, sencillo de leer y flexible, debido a estas características se trata del lenguaje más utilizado para el desarrollo de proyectos de aprendizaje automático o inteligencia artificial. Además, es compatible con todas las plataformas, gratuito y de código abierto, lo que ha permitido que contenga una gran cantidad de librerías.

R

Se trata de un lenguaje de programación orientado principalmente al análisis estadístico utilizado para minería de datos, matemáticas financieras o aprendizaje automático entre otros. Es multiplataforma, fácil de aprender y de código abierto, es por ello por lo que contiene una gran variedad de paquetes desarrollados por usuarios que amplían las capacidades de este lenguaje.

Javascript

Es un lenguaje de programación interpretado, sin necesidad de compilación, que funciona en los navegadores de forma nativa, orientado a objetos, dinámico y de aprendizaje sencillo, que se utiliza principalmente como complemento de HTML y CSS para crear páginas webs. Existen paquetes y librerías enfocadas a ciencia de datos, entre ellos TensorFlow.js, orientada al aprendizaje automático que permite el entrenamiento e implementación de modelos en navegadores y en el entorno de ejecución de JavaScript Node.js

C++

Es un lenguaje de programación de alto nivel orientado a objetos. Originalmente se trataba de una extensión del lenguaje C y su sintaxis principal proviene de este. Entre sus ventajas cabe destacar su alto rendimiento, es multiplataforma y posee un lenguaje actualizado y muy extendido que está presente en gran cantidad de programas o sistemas.

Analizando las diversas opciones optamos por descartar C++ debido a que el manejo de librerías es más complicado que en otros lenguajes y su curva de aprendizaje es alta, y JavaScript lo descartamos porque aunque disponga de algunas librerías orientadas al "machine learning" no está principalmente enfocada a esta función.
Finalmente, nos decantamos por utilizar MATLAB debido a que sus funciones y diversas librerías que contiene acerca de la gestión de análisis de datos y optimización hacen de MATLAB un perfecto entorno para aplicar técnicas y algoritmos de aprendizaje automático. Además, también se trataba del entorno más familiar y con el cual no tendríamos que preocuparnos por la curva de aprendizaje y donde teníamos implementado el proceso de optimización BADS.
Aunque nos hayamos decantado por MATLAB también utilizaremos Python para implementar los bots en el concurso de Kaggle de Piedra-Papel-Tijeras ya que se trabajaba con este lenguaje.

Respecto a los modelos utilizados no tuvimos un planteamiento de alternativas, ya que, se trata de un estudio en el que probamos los diferentes modelos para analizar cuál de ellos produce mejores resultados.

2.3 Descripción detallada de la solución adoptada

En este apartado detallaremos el procedimiento que se ha llevado a cabo para la realización del estudio. Primero explicaremos en qué consiste el programa y los procesos utilizados para su cálculo y posteriormente explicaremos los diferentes procesos de refuerzo computacional que se han realizado. Hemos utilizado la versión Matlab R2021a para programar y ejecutar el proceso.

2.3.1 Obtención de datos

En primer lugar, para poder analizar el comportamiento humano, hemos adquirido una

muestra de datos sobre su comportamiento en el juego de “matching pennies”. Como hemos comentado anteriormente, hemos recibido los datos de un estudio sobre la toma de decisiones realizado por la escuela de medicina de Yale. Guardamos en el fichero “rl_subjects.mat” los datos de elección y recompensa a lo largo de 200 ensayos de los 30 participantes del estudio. Los datos de elección vienen representados por 0 o 1 en la columna “rl_subjects.choice”. Los datos de la recompensa vienen representados por 1 si obtuvieron recompensa o 0 en caso contrario. Se almacena en la columna “rl_subjects.reward”.

A partir de estos datos, tratamos de ver mediante los algoritmos cómo se comporta el ser humano en la toma de decisiones, viendo cuál de ellos es capaz de reproducir mejor la respuesta del ser humano, y comprobamos cuáles tienen un mejor funcionamiento y son capaces de obtener mayor recompensa.

2.3.2 Modelos

Una vez tenemos los datos de entrada lo siguiente ha sido implementar los cinco modelos de aprendizaje por refuerzo que vamos a utilizar: “No adaptativo”, “Relevancia”, “Sorpresa”, “Información” e “Incertidumbre”.

Los modelos han recibido unos parámetros iniciales a partir de los cuales empezar el proceso. A cada ensayo los agentes se han ido entrenando para poder predecir mejor el resultado y maximizar sus resultados. Los pasos que han realizado los modelos a lo largo de 200 *trials* son los siguientes:

- Primero hemos calculado la probabilidad de realizar las selecciones basadas en el valor de asociación.
- A continuación, el agente ha escogido una de las dos opciones. Como consecuencia de esto hemos obtenido una recompensa o no.
- Hemos seguido con la actualización del valor a partir de los valores previos. A partir de la siguiente ecuación (3):

$$V_i^n = V_i^{n-1} + R\,\eta_i\left(Opción_{elegida}\left(R - V_i^{n-1}\right) - wdec \cdot Opción_{noelegida} \cdot V_i^{n-1}\right) + (1 - R)\,\eta_i(Opción_{noelegida}\left((1 - R) - V_i^{n-1}\right) - ewdec \cdot Opción_{elegida} \cdot V_i^{n-1}) \quad (3)$$

Donde recordamos que:

$\boldsymbol{V_i^n}$**:** Representa el total de la variación del valor asociado a un estímulo en la ronda.
$\boldsymbol{V_i^{n-1}}$**:** Es el valor asociado a un estímulo en la ronda anterior. Tras el cálculo de la ecuación este valor se actualizará.
$\boldsymbol{R}$**:** Es el valor de la señal de salida. Si hay recompensa será 1 y si no será 0.
$\boldsymbol{\eta}$**:** Es la tasa de aprendizaje o *“learning rate”*. Es un hiperparámetro que controla cuánto cambio experimenta el modelo en respuesta al error estimado cada vez que se modifican los pesos del modelo, metafóricamente hablando representa la velocidad de aprendizaje del modelo.
Opción_elegida: La elección escogida. Si elegimos la primera opción será [1 0], si elegimos la segunda opción tendrá el valor [0 1].
Opción_no_elegida: Se trata del valor de la opción que no hemos escogido. [0 1] si elegimos la primera opción y [1 0] si elegimos la segunda.
Wdec y ewdec: Son valores relativos a la tasa de aprendizaje que vienen dado por los valores iniciales.

El valor se actualiza a partir del valor de la ronda anterior y de la salida (si ha obtenido recompensa). V presenta dos valores [V_1, V_2], uno para cada una de las opciones, en

función de si obtenemos recompensa y el valor escogido, aumentan o disminuyen cada uno de los valores. Si escogemos el primer valor y obtenemos recompensa, V_1 aumenta mientras que V_2 disminuye. Cuanto mayor sea el valor V mayor probabilidad tiene de ser la opción escogida.

- Una vez tengamos actualizado el valor procederemos al cálculo de la tasa de aprendizaje. Este cálculo varía dependiendo del modelo que estemos aplicando:

En el apartado 1.2.3 explicamos con detalle en que consiste cada modelo, no obstante, vamos a definir muy brevemente cada uno de los modelos:

Información: Modelo basado en la información. La tasa de aprendizaje aumentará para aquellas variables que sean capaces de predecir la recompensa y penalizar aquellas que no den información. Los valores se encuentran entre 0 y 1. En los extremos existe correlación, si se acerca a 1 tendremos una correlación positiva entre lo que se esperaba y se ha obtenido y si se acerca a 0 la correlación será negativa, ya que, no se ha obtenido lo esperado. Cuando haya correlación aumentará la tasa de aprendizaje. Cuando los valores se acercan al punto medio, 0.5, no hay correlación y los valores no son significativos. Esto crea mayor incertidumbre e implica una disminución de la tasa de aprendizaje.

Incertidumbre: Modelo que funciona al contrario de "Información". La tasa de aprendizaje aumenta cuando los valores están alrededor de 0.5. Este modelo permite que el agente se adapte más rápidamente a cambios de sistematicidad.

Relevancia: La tasa de aprendizaje presentará un aumento para los estímulos elegidos si predice el resultado (son relevantes). Para aquellos estímulos que no han predicho el resultado, y por tanto no son relevantes, la tasa de aprendizaje disminuirá.

Sorpresa: La tasa de aprendizaje aumentará cuanto mayor sea la sorpresa, es decir, aumente la discrepancia entre lo que se espera obtener y lo que se produce $(R - V)$.

No adaptativo: Se basa en el modelo de Rescorla-Wagner donde la tasa de aprendizaje es constante.

- El proceso seguirá actualizando los valores y la tasa de aprendizaje al largo de los 200 *trials*. Finalmente obtendremos los resultados, ya sean de las probabilidades o recompensa, obtenidos por el modelo.

A continuación, vamos a explicar los distintos procesos realizados.

2.3.3 Comportamiento humano. Maximización de la probabilidad.

Para entender el comportamiento humano nuestro proceso tratará de maximizar las probabilidades de elegir las mismas elecciones que realizaron los sujetos.
Gracias a los datos obtenidos tenemos la información de cada una de las elecciones que realizaron a lo largo de los 200 *trials*, de esta manera, podemos calcular los valores y probabilidades que hay de realizar la misma elección que el sujeto.
Para ello lo compararemos con los diferentes modelos implantados para ver cuál de ellos se asemeja más al comportamiento humano.

Este proceso está implementado en la función "mp_rl_agent_prop.m" cuya llamada sería la siguiente:

function [optMetric, optimization_data] = mp_rl_agent_prop(x, task_data)

Donde pasándole las variables de inicialización nos devolverá la métrica y el resto de los elementos de estudio.

Sus valores de entrada son:

x: Contiene los valores de los parámetros de inicialización η_0, μ, T, wdec, ewdec que han sido previamente optimizados en BADS.

task_data: Campo de tipo estructura que contiene los datos de elección y recompensa de los sujetos, así como el id asociado al sujeto del que se toman los datos.

Sus valores de salida son:

optMetric: Devuelve la métrica a evaluar, ya sea la probabilidad de elegir la misma respuesta que el sujeto o la media de la recompensa obtenida.

optimization_data: Estructura que devuelve los campos de salida del algoritmo como la elección del algoritmo (opt_chosen), *"learning rate"*, recompensa, etc.

A continuación, exponemos como maximizamos la probabilidad.

Probabilidad de escoger la misma elección que el sujeto de estudio

1. Iniciamos con los valores de entrada de los parámetros η_0, μ, T, wdec y ewdec previamente optimizados por BADS. Y con los valores de elección y recompensa obtenidos por los sujetos y el modelo a implementar.
2. Asignamos a η el valor de η_0 y a V como 0.5 en su primera ronda.
3. Iniciamos bucle. A lo largo de 200 iteraciones realizaremos lo siguiente:
   Selección basada en el valor: valOpt = V.
4. Calculamos las probabilidades de escoger cada una de las opciones:
   pOpt = exp(valOpt/T).
   pOpt = pOpt/sum(pOpt);
5. Seleccionamos la misma respuesta que ha elegido el sujeto:
   opción_elegida = choice(ind_trials).
   opción_no_elegida = ~opción_elegida.
6. La salida R guardará la recompensa que tuvo el sujeto:
   R = reward(ind_trials).
7. Actualizamos el valor a partir de los valores previos y en función de la salida y opción elegida:
   V= V + R η(opción_elegida(R -V) - wdec·opción_no_elegida·V) + (1- R) η(opción_no_elegida((1- R)- V) - ewdec·opción_elegida·V).
8. Aplicamos el cálculo de la variable (información, relevancia, sorpresa, incertidumbre o no adaptativo) en función del modelo que se esté ejecutando.
9. Actualizamos el valor del *"learning rate"* o tasa de aprendizaje:
   η = η + μ (Variable_modelo - η).
10. Guardamos la probabilidad de selección del modelo, esta será la probabilidad que tiene el modelo de escoger la misma opción que el sujeto:
    ModelSelectedProb(ind_trials) = pOpt(opt_chosen(ind_trials)).
11. Fin del bucle. Una vez se han recorrido los 200 *trials* sacamos la media de las probabilidades de selección:
    optMetric = ModelSelectedProb.
12. Devolvemos los valores de salida.

A lo largo de las iteraciones se irán actualizando los valores e irá variando la tasa de

aprendizaje en función de lo acontecido en anteriores rondas, el modelo irá aprendiendo de las elecciones pasadas y esto afectará a un aumento o disminución de las probabilidades de cada una de las opciones a escoger. Aquel modelo cuyos valores de "*optMetric*" se acerquen más a 1 indicará que el modelo tiene una mayor probabilidad de reproducir la misma respuesta que los sujetos.

2.3.4 Maximización de la recompensa frente a un modelo computacional

En este proceso comprobaremos cuál de los modelos funciona de manera más eficiente frente a un modelo computacional.
Durante el estudio que realizó la escuela de medicina de Yale, los participantes se enfrentaron a un modelo computacional contra el que tenían que competir eligiendo una de dos opciones disponibles. Lo que hacemos es competir contra el perfil de respuestas resultante del enfrentamiento contra este agente virtual para cada participante, de manera que podemos comprobar si los modelos que hemos implantado se asemejan o mejoran el comportamiento humano en la toma de decisiones. Para ello, maximizaremos la recompensa de nuestros modelos, y, al igual que en el caso anterior, iremos actualizando el valor a partir del valor de la ronda anterior y en función de su elección, recompensa y de la tasa de aprendizaje. Para maximizar la recompensa elegiremos la elección en función de la probabilidad con la función SOFTMAX (Apartado 1.2.2).
Tras la elección de la respuesta miraremos si obtuvimos o no recompensa.
Finalmente ajustaremos el valor de V y μ dependiendo del modelo como hemos explicado en el apartado (1.2.3)

Método de selección y recompensa
1. Selección basada en el valor: valOpt = V.

2. Calculamos las probabilidades de escoger cada una de las opciones:
pOpt = exp(valOpt/T);
pOpt = pOpt/sum(pOpt);

4. Realizamos la elección con la función SOFTMAX que maximiza la probabilidad de escoger la acción de mayor recompensa: opt_chosen(ind_trials) = find(cumsum(pOpt) >= rand, 1).
5. Ajustaremos los valores V y μ dependiendo del modelo como hemos visto en apartados anteriores.

En este caso la variable de salida "optMetric" devolverá la media de la recompensa obtenida por el modelo, que es lo que estamos midiendo en este proceso.
Este proceso está implementado en la función "mp_rl_agent_reward.m".

Al enfrentamiento contra esta serie de respuestas lo consideramos un agente estático porque las respuestas resultantes de los participantes contra el modelo computacional, contra las que compiten nuestros agentes, siempre serán las mismas cada vez que lo enfrentemos contra uno de los nuestros modelos.
Mientras que, para un modelo dinámico, se enfrentarán a un agente cuyas respuestas se escogen al momento respecto a las elecciones de los modelos RL, y, por tanto, cada enfrentamiento contra este agente dinámico presentará una serie de respuestas diferentes cada vez. Este modelo dinámico lo veremos a continuación.

2.3.5 Maximización de la recompensa frente a un modelo dinámico

Por último, maximizaremos la recompensa frente a un modelo dinámico para ver cuál de ellos compite mejor. En el caso anterior habíamos maximizado la recompensa en un entorno del cual ya estaban predeterminadas las elecciones y sus recompensas. Esta vez

los modelos se enfrentarán a un agente cuya elección es cambiante y que tratará de escoger la mejor selección posible teniendo en cuenta las elecciones y recompensas del pasado al igual que nuestro modelo. Este otro agente esta implementado en la función "matching_pennies.m"
Nuestros modelos ganarán si eligen la misma respuesta que el agente adversario, en caso contrario no obtendrán recompensa.

Método de selección y recompensa en su enfrentamiento a un modelo dinámico

1. Con el proceso matching_pennies le damos un valor de selección a computerChoice. Este proceso trata de escoger la mejor selección posible teniendo en cuenta las elecciones y recompensas del pasado:
   [computerChoice, pComputerRight, biasInfo] = matching_pennies(data, 4, 0.05).
2. Selección basada en el valor: valOpt = V.
3. Calculamos las probabilidades de escoger cada una de las opciones:
   pOpt = exp(valOpt/T)
   pOpt = pOpt/sum(pOpt)
4. Realizamos la elección con la función SOFTMAX que maximiza la probabilidad de escoger la acción de mayor recompensa:
   opt_chosen(ind_trials) = find(cumsum(pOpt) >= rand, 1)
5. Si elegimos lo mismo que computerChoice Entonces
   la recompensa será 1
   Sino
   La recompensa será 0
   Fin-Si

Al igual que el caso anterior utilizaremos la función SOFTMAX para realizar la selección por parte del modelo.
La variable de salida "optMetric" devolverá la media de la recompensa obtenida por el modelo.
Este proceso nos servirá para comprobar que modelo compite mejor. **Nos servirá también para poder enfrentar a los modelos entre ellos mismos.**
La función encargada de realizar este proceso es "mp_rl_agent_dyn_reward.m".

2.3.6 Optimización de parámetros

Para que los valores iniciales de los modelos sean óptimos utilizaremos el optimizador BADS (Bayesian Adaptive Direct Search for nonlinear function minimization).
Este se encarga de realizar optimización bayesiana para la resolución de problemas de optimización complejos. Es ideal para el ajuste de modelos computacionales de un número reducido de parámetros libres.

Ejemplo de llamada:

```
[X, fval ,exitflag] = bads(fun, x0, LB,UB,LB'+realmin,UB'-realmin, Options);
```

Para cada uno de los participantes ejecutamos la función BADS. A partir de los valores iniciales de entrada se obtiene un mínimo local X para la función de la variable "*fun*". Esta función, que es donde calculamos el *"learning rate"*, obtiene X y devuelve un valor de la función escalar evaluado en X.
Esta función nos devuelve los parámetros iniciales η_0, μ, T, wdec y ewdec que se irán ajustando a través del proceso de optimización de esta función.
Estos parámetros se mantendrán a lo largo de la ronda para un mismo sujeto, y volverán a optimizarse cuando cambiemos de participante.

Sus valores de entrada son:

fun: Llamada de la función que realiza el cálculo de la tasa de aprendizaje.

x0: Valores iniciales de los parámetros para cada una de las iteraciones.

LB: Determina los límites del valor inferior de los parámetros.

UB: Límites superiores de los parámetros.

OPTIONS: Devuelve una estructura de opciones predeterminada.

Sus valores de salida son:

X: Valores finales de los parámetros para obtener un mínimo local de la función "*fun*".

FVAL: Devuelve el valor de la función "*fun*".

EXITFLAG: Describe la condición de salida. Estos son los valores que adopta:
0: Número máximo de evaluaciones de funciones o iteraciones alcanzadas.
1: La magnitud del valor es menor que la tolerancia especificada.
2: El cambio del valor de la función estimado es menor que la tolerancia especificada.

2.3.7 Problemáticas y solución

En un primer momento realizamos la optimización de la recompensa de los agentes frente a las respuestas proporcionadas por los sujetos. Esto nos planteó un problema, ya que, al enfrentarse a una serie de respuestas estáticas, la optimización de los mecanismos de los modelos RL resultaba ser artificiosa debido a que el proceso de BADS obtiene ventaja al enfrentar siempre el mismo patrón de respuestas.

Para poder tener unos resultados más justos y ajustados a la realidad debíamos enfrentar los modelos RL a agentes libres cuyas elecciones no fueran siempre las mismas. Al enfrentar los modelos RL contra agentes libres, donde debido a la estocasticidad podíamos obtener resultados diferentes, tuvimos que pensar la manera de implementar un sistema que fuera robusto y del que pudiéramos obtener una distribución real y representativa del sistema. Para ello, decidimos realizar la optimización de los parámetros del modelo RL y realizar 100 enfrentamientos contra el agente libre y hacer una media de los resultados obtenidos de la recompensa. De esta manera obtendremos una respuesta más ajustada a la realidad y podremos comparar los resultados de la recompensa obtenidas por los modelos RL frente a la recompensa que obtuvieron los sujetos al enfrentarse a un agente libre.

2.4 Justificación de la solución adoptada

2.4.1 Resultados

En este apartado vamos a ver los diferentes resultados obtenidos en el que comprobaremos que modelo explica mejor el comportamiento humano en función de maximizar las probabilidades de escoger la misma elección que el sujeto y viendo que

modelo compite mejor contra un agente estático y cómo se comporta contra uno dinámico. El proceso por el cual se generan los resultados de cada uno de los modelos es "mp_run_bads.m" el cual realiza la optimización de los parámetros iniciales mediante BADS para cada modelo RL y lo guarda en un archivo *.mat*.

2.4.1.1 Resultados del estudio del Comportamiento humano

A continuación, comprobaremos como se ha comportado cada uno de los modelos al tratar de conseguir realizar la misma elección que los sujetos. Se realiza para 30 sujetos por cada uno de los modelos. Para cada sujeto se hacen 200 *trials* y se saca la media de la probabilidad de elegir la respuesta del sujeto. Este proceso de optimización se ejecuta unas 100 veces y se obtiene el conjunto de datos de la media de probabilidades de elegir la respuesta del sujeto en las 100 ejecuciones. A partir de estos datos construimos la gráfica de densidades (Figura 6).

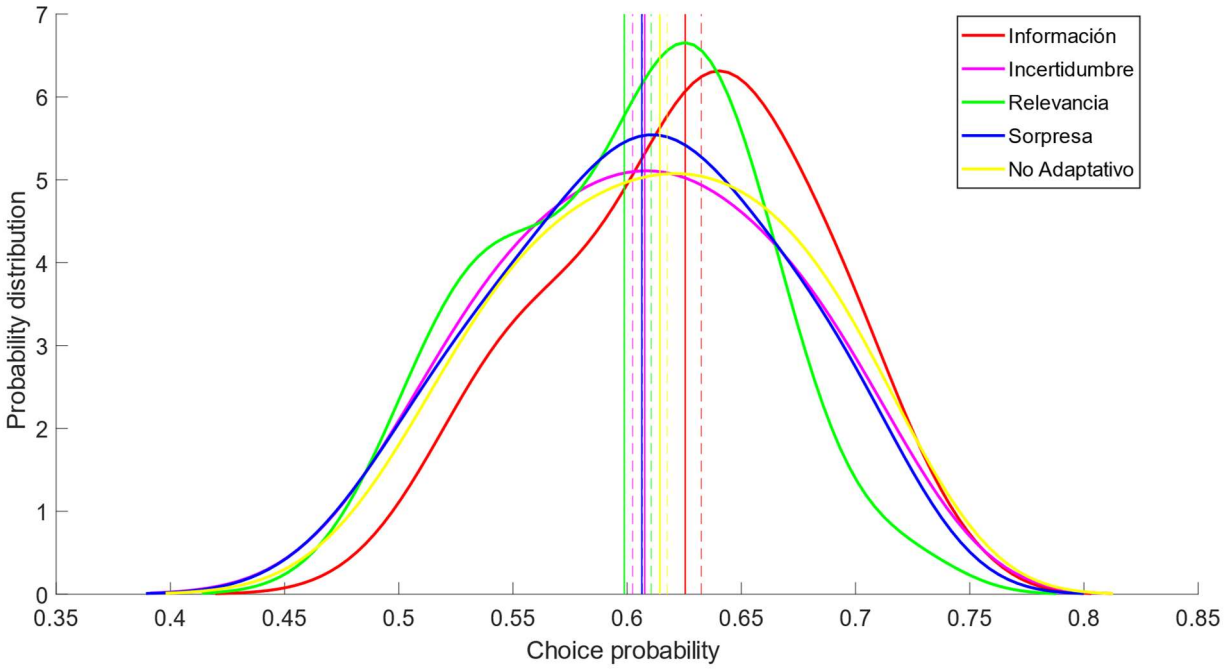

Figura 6. *Gráficas de densidad que compara las medias de las probabilidades de elegir la misma opción que los sujetos entre los distintos modelos. Las líneas verticales continuas representan la media total de las probabilidades por modelo, mientras que las líneas discontinuas muestran la mediana.*

De los modelos que se muestran, "No Adaptativo" fue el primer modelo que surgió basado en Rescorla-Wagner. A continuación, surgieron los modelos "Relevancia" (Mackintosh) y "Sorpresa" (Pearce-Hall) que proponían el concepto de la modificación del "*learning rate*". El modelo "Relevancia" funciona en base al valor, cuando este aumenta también lo hace su "*learning rate*" y este aumento se produce cuando el valor escogido le aporta recompensa. El modelo "Sorpresa", por el contrario, proponía que el "*learning rate*" variaba con la sorpresa, es decir, cuando el sujeto no espera una respuesta y la obtiene, esto produce un aumento del "*learning rate*", mientras que si a un valor que espera no obtiene la salida deseada, esto hará que esta tasa de aprendizaje disminuya.

El modelo "Información", se trata del algoritmo que hemos propuesto. Este modelo parte de la base del modelo de "Relevancia", que funcionaba en función del valor, pero en este caso vamos a tener en cuenta la información que el valor nos devuelve en base a su "*baseline*". Los valores se encuentran entre 0 y 1, y por tanto su "*baseline*" o línea basal es 0.5. Cuando más se aleje el valor del "*baseline*" mayor será la información recibida y aumentará el "*learning rate*", cuando los valores estén más cerca del 0.5 eso nos

proporcionará poca información y la tasa de aprendizaje disminuirá.
Mientras que “Relevancia” solo aumentaba la tasa de aprendizaje, en el modelo “Información” tanto en correlaciones positivas con valores cercanos a 1, como en correlaciones negativas con valores cercanos a 0 nos aporta información y este modelo podrá aumentar o disminuir.

El último modelo, “Incertidumbre”, parte de la idea del modelo de “Sorpresa” y funciona en base a la sorpresa ante cambios bruscos en la información. Se trata de un modelo que hemos añadido con el fin de completar las posibilidades planteadas.

Si observamos la gráfica de la figura 6 podemos ver que el modelo propuesto, “Información”, es el que mejor reproduce el comportamiento humano consiguiendo reproducir la misma selección que los sujetos con una probabilidad mayor al 60%. Podemos observar que mejora bastante a su predecesor, “Relevancia”, aunque este último veamos en la gráfica que su distribución es más picuda y ancha, eso no debe llevarnos a engaño, ya que debemos observar que los rangos de probabilidad en los que se mueve son menores y con una curva pronunciada en los valores bajos, si observamos la tabla 1 veremos que su media de probabilidad de elegir la misma opción que los sujetos es menor y no alcanza el 60%. En cambio, la que obtiene de media unos niveles de probabilidad más altos es el “Información”.

También podemos comprobar que los modelos “Incertidumbre” y “Sorpresa”, que ambos funcionan bajo la misma base, en función de la sorpresa, obtienen resultados similares, aunque no tan buenos como el de “Información”.

Modelo de *reinforcement learning*	Media de la probabilidad de los 30 sujetos
Información	0.6254
Incertidumbre	0.6077
Relevancia	0.5987
Sorpresa	0.6064
No adaptativo	0.6143

Tabla 1. *Media de las probabilidades totales de los sujetos por modelo.*

A continuación, para ver de forma más clara que modelo se comporta mejor respecto a los demás. En la figura 7 vemos una tabla comparativa donde se comparan las probabilidades de los modelos entre sí.
Los modelos se enfrentan dos veces entre sí, Modelo A contra Modelo B y Modelo B contra Modelo A, pero solo lo mostramos una vez, ya que las figuras son las mismas, pero con la curva invertida. Al no repetir enfrentamientos entre modelos mostramos la tabla en forma triangular donde cada fila corresponde al modelo que se toma como referencia y cada columna al modelo adversario. Por tanto, si la curva es positiva querrá decir que la probabilidad del modelo es superior a la del modelo adversario, y si es negativa querrá decir que las probabilidades del modelo son inferiores a las de su contrincante.

Para la realización de esta figura lo que hemos hecho ha sido sacar la diferencia entre las probabilidades de los modelos que se comparan e ir acumulando la suma de esta diferencia al largo de los 200 *trials*.

En esta figura vemos que el modelo “Información” es capaz de reproducir el comportamiento humano mejor que el resto de los modelos, podemos observar en la primera fila presenta una curva ascendente contra todos los modelos. Esto nos indica que evaluar las decisiones en función de la información que cada uno de estos valores nos proporciona y de sus correlaciones sistemáticas, se asemeja más al pensamiento humano y su criterio a la hora de decidir que opción escoger entre las dos disponibles.

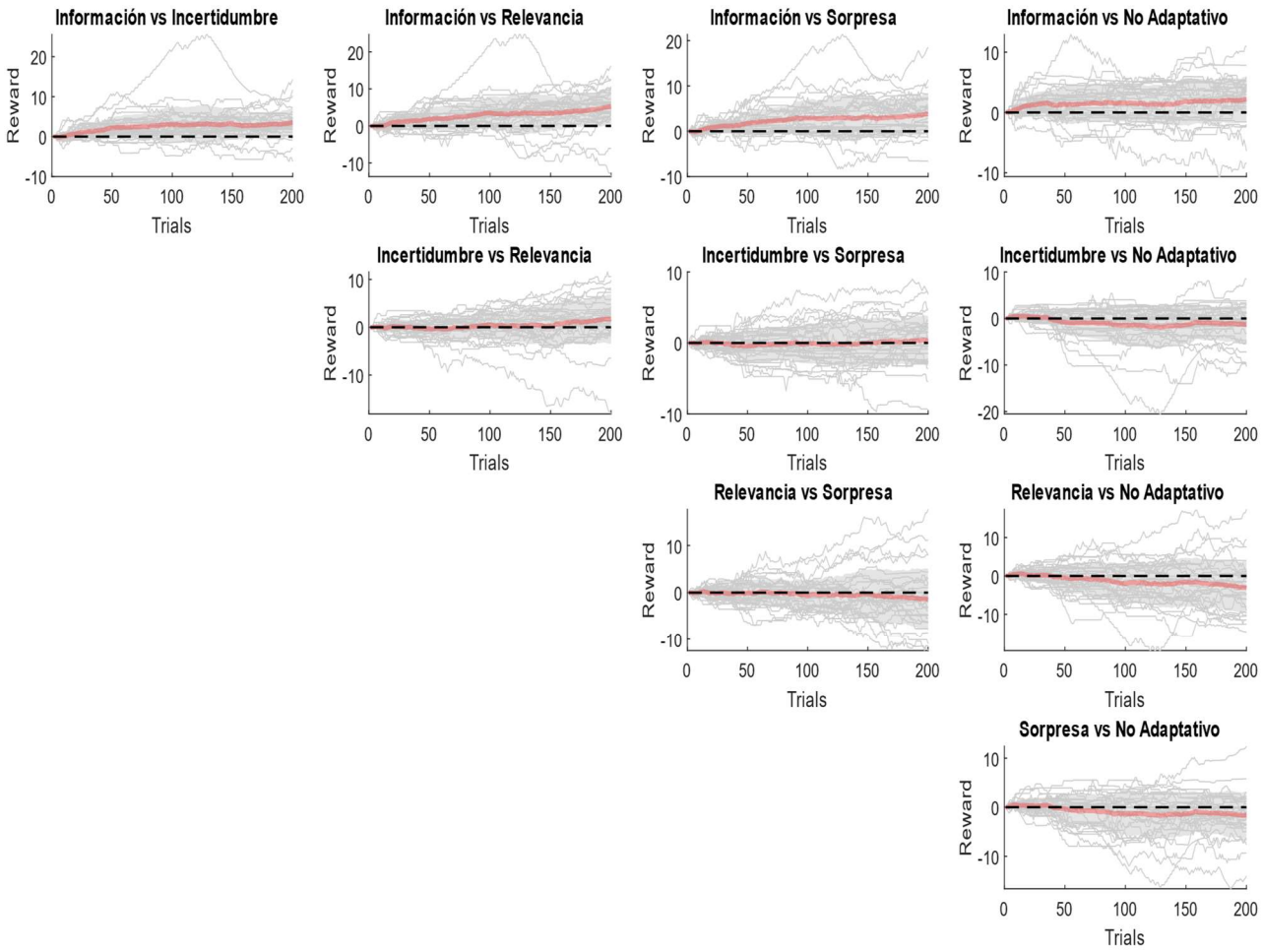

Figura 7. *Comparativa de reproducción del comportamiento humano entre los modelos de reinforcement learning. Las líneas grises representan las probabilidades por cada uno de los sujetos, mientras que la línea roja muestra la media de las probabilidades de todos los sujetos por modelo.*

También podemos corroborar que los modelos "Incertidumbre" y "Sorpresa" son prácticamente igual de eficientes, mientras que el modelo "Relevancia" en el que solo se produce un aumento del "*learning rate*" cuando los resultados obtenidos son los deseados es menos eficiente que el resto de los modelos.
Por último, observamos que el modelo "No Adaptativo", que al igual que "Relevancia", funciona en base al valor, pero sin tener en cuenta la dinámica de la toma de decisiones tiene una buena representación del comportamiento humano.
El proceso mediante el que lanzamos las comparativas de probabilidades de los modelos es "mp_run_bads_versus_models_prop.m"

2.4.1.2 Resultados de la maximización de la recompensa

A continuación, vamos a mostrar los resultados referentes a la maximización de la recompensa. Primero vamos a analizar cómo se comportan los diferentes modelos frente a un modelo estático. En este caso el uso de selecciones se hará de manera estocástica donde a través de la función SOFTMAX haremos que sea tan probable de elegir una opción como lo sea su probabilidad [15].
Para ver que modelo compite mejor frente a un sistema estático, donde el agente oponente siempre repite los patrones de selección vamos a comparar los diferentes modelos entre sí. En la figura 8 mostramos la comparativa de la recompensa entre los modelos. Los datos se obtienen a través de 100 ejecuciones por modelo, donde en cada una de estas ejecuciones se compara la recompensa obtenida por los agentes ante la

misma serie de respuestas realizadas por el agente computacional ante el que se enfrentan.

En este caso no mostramos la comparación de un modelo contra sí mismo. Al ser las elecciones estocásticas que parten de los mismos parámetros optimizados por BADS se darán solo ligeras variaciones en los valores y toma de decisiones, por ello, al tratarse de un sistema estático la diferencia entre un mismo modelo al ser comparado en dos ejecuciones distintas es mínima (recordemos que cada una de estas ejecuciones se lanza 100 veces por sujeto). Observamos que el agente en base a la "Información" es el que tiene ligeramente un mejor funcionamiento. La optimización de la recompensa es más eficiente en algoritmos cuyo "*learning rate*" evoluciona en función de la información respecto al "*baseline*".
Por el contrario, vemos que el algoritmo "Sorpresa" es el que optimiza ligeramente inferior al resto. Podemos deducir que al tratarse de un modelo cuyo "*learning rate*" aumenta en función de la sorpresa, al enfrentarse a un sistema estático donde no habrá mucha diferencia entre el valor esperado y el obtenido, su tasa de aprendizaje es menos dinámica, lo que implica que la optimización sea un poco menos eficiente.

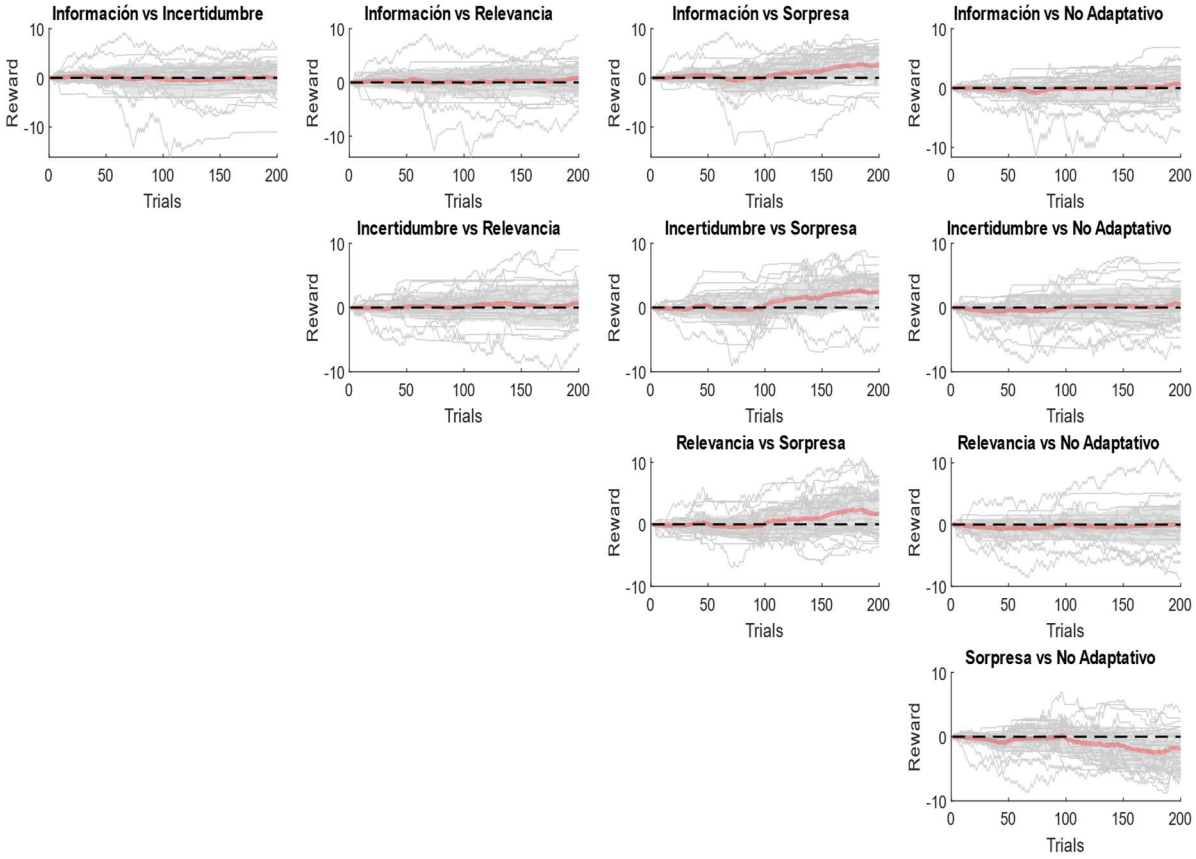

Figura 8. *Comparativa de la recompensa frente a un modelo estático entre los modelos de reinforcement learning.*

Por último, para comprobar el rendimiento de los modelos hemos creado la figura 9 donde mostramos el acumulado de la recompensa de cada uno de los modelos respecto al agente estático. Para esta gráfica hemos considerado interesante añadir también el acumulado de la recompensa obtenida por los sujetos, aunque la comparativa no es del todo justa, ya que, cada uno de los sujetos realizó una vez la prueba, mientras que los modelos la ejecutan 100 veces por cada uno de los sujetos, los cuales, tomamos de referencia para replicar la misma prueba que estos realizaron, pero enfrentándolo contra los modelos de aprendizaje por refuerzo. Si los humanos se enfrentaran a las mismas respuestas es posible que mejoraran a lo largo de 100 ejecuciones.

Aunque los sujetos solo hayan realizado una vez la prueba, a modo orientativo si nos sirve para ver que el acumulado de la recompensa que obtuvieron fue ligeramente negativo, aunque si observamos las líneas grises (figura 9 izquierda) que corresponden a los acumulados de cada uno de los sujetos vemos que, a excepción de unos cuatro sujetos que terminaron con una recompensa muy negativa, el resto oscila entre 20 y -20 y, por tanto, su porcentaje de acierto fue alrededor del 50%.Por el contrario, vemos que los algoritmos de aprendizaje por refuerzo obtienen una mayor recompensa al utilizarse la función BADS que optimiza los mecanismos de los modelos RL con tal de hackear al máximo las respuestas estáticas del agente computacional.

Entre los modelos cabe destacar que su evolución es muy similar a lo largo de los *trials,* a excepción del modelo "Sorpresa" que se va descolgando debido a la ausencia de sorpresa que hace que disminuya su rendimiento.

Podemos comprobar que, efectivamente, el modelo "Información" que hemos propuesto es el que mejor rendimiento presenta y por tanto la optimización en base al valor es la más eficiente, aunque como hemos podido comprobar todos son mejores optimizadores que los sujetos.

El proceso mediante el que lanzamos las comparativas de la recompensa de los modelos frente a un agente estático es "mp_run_bads_versus_models_reward.m"

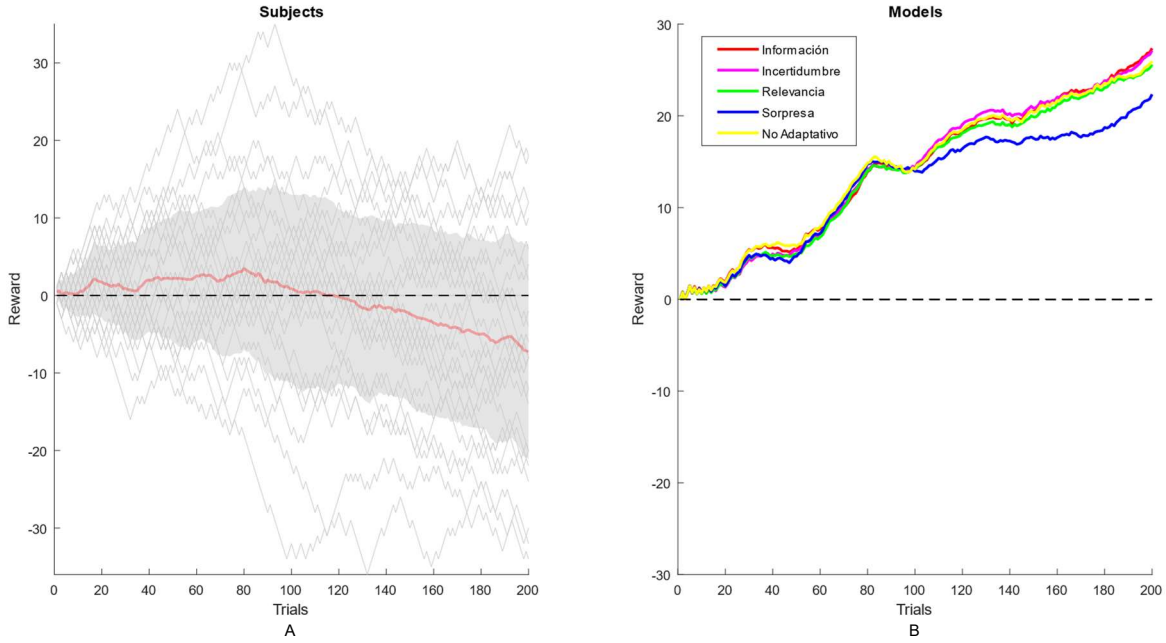

Figura 9. *Comparativa de la recompensa de los sujetos (A) frente a los modelos RL (B). A la izquierda tenemos la representación de la recompensa de los sujetos, donde las líneas grises representan la media de la recompensa de cada uno de ellos y la línea roja la media global. A la derecha se representa la recompensa frente a un modelo estático entre los modelos de reinforcement learning.*

Finalmente nos queda por ver cómo se comportan los modelos optimizando su recompensa contra un agente dinámico. Este es el caso más complejo, ya que el agente contrario, llamado "Computer choice", tratará de optimizar su recompensa dependiendo de los valores anteriores y de si obtuvieron o no recompensa, haciendo que nos encontremos en un escenario de mayor incertidumbre en el que no sepamos que va a elegir nuestro adversario.

Creamos la figura 10 para remarcar la gran variabilidad que se muestra en las diferentes ejecuciones respecto la mediana. En esta figura podremos observar el comportamiento de los modelos entre ellos, cada modelo se enfrentará al resto de modelos y al "Computer

choice". En esta ocasión también enfrentamos cada modelo contra sí mismo ya que la elección, aunque esté basada en la probabilidad, tiene cierta aleatoriedad. En este caso es difícil determinar cuál de los modelos se comporta de forma más eficiente, podemos observar que en los duelos entre ellos la diferencia de la recompensa obtenida es muy poca, apenas perceptible, y por ello en la figura se pretende ilustrar la variabilidad.

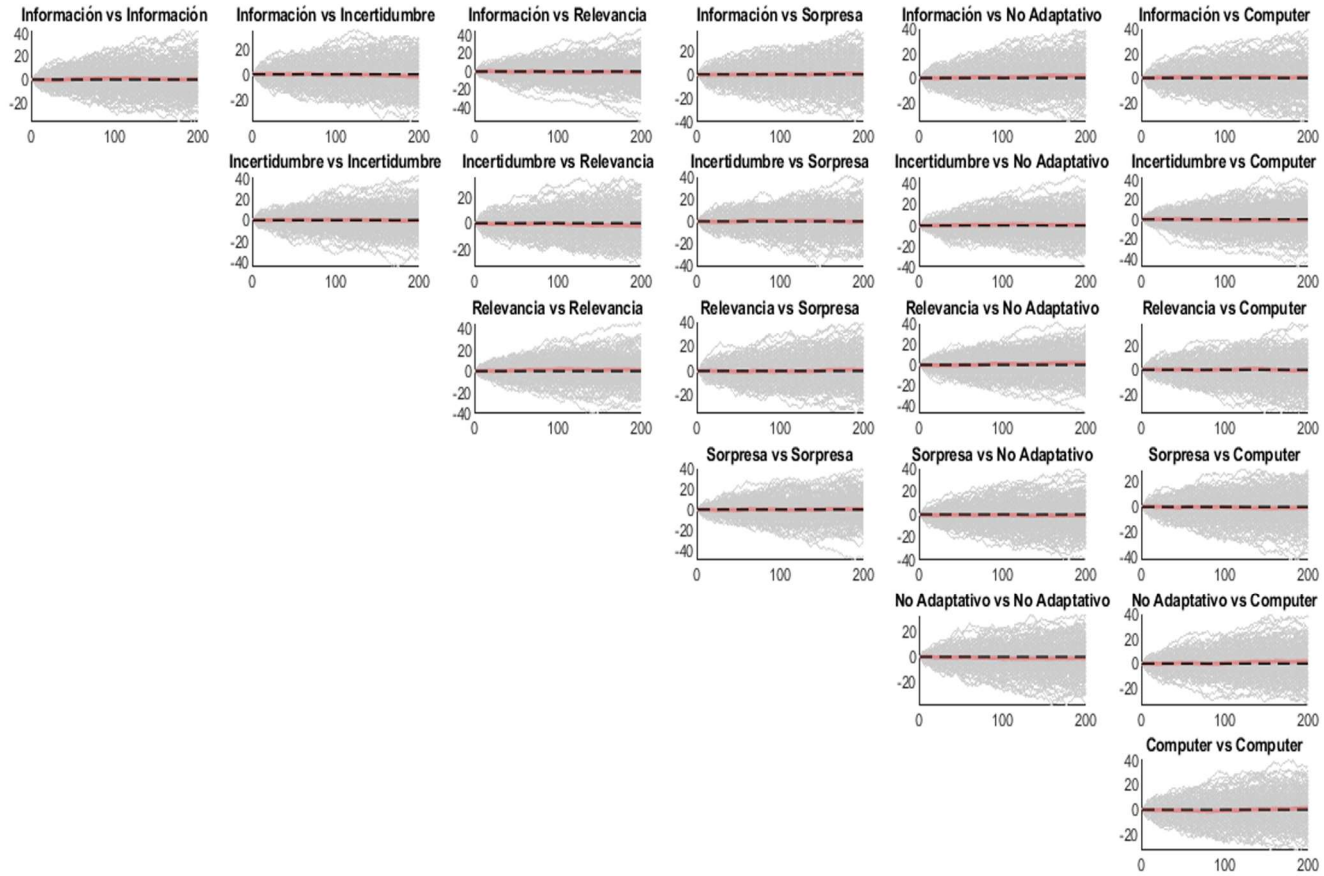

Figura 10. *Comparativa de la recompensa frente a un modelo dinámico entre los modelos de reinforcement learning y el modelo dinámico Computer Choice. Se muestra la variabilidad de cada una de las ejecuciones (líneas grises) respecto de la mediana (línea roja)*

Para poder observar cómo se comportan en su enfrentamiento contra el modelo dinámico y hacer una comparativa ajustada, hemos realizado la figura 11 donde mostramos la media de 100 enfrentamientos entre los modelos y el "Computer Choice", en esta figura los modelos y el agente dinámico ganan cuando el resultado elegido es el mismo por ambas partes y el "Computer Choice" adversario gana cuando es diferente.

La competición entre los "Computer Choice" remarca el nivel de estocasticidad ya que el mecanismo es exactamente el mismo. En estas figuras vemos que la diferencia entre los modelos es mínima y la recompensa oscila entre 2 y -2.
La conclusión que podemos sacar es que todos los modelos son igual de buenos frente a un agente estocástico, donde vemos que los modelos son estables y presentan ligeras desviaciones con pequeñas fluctuaciones generadas por ruido.

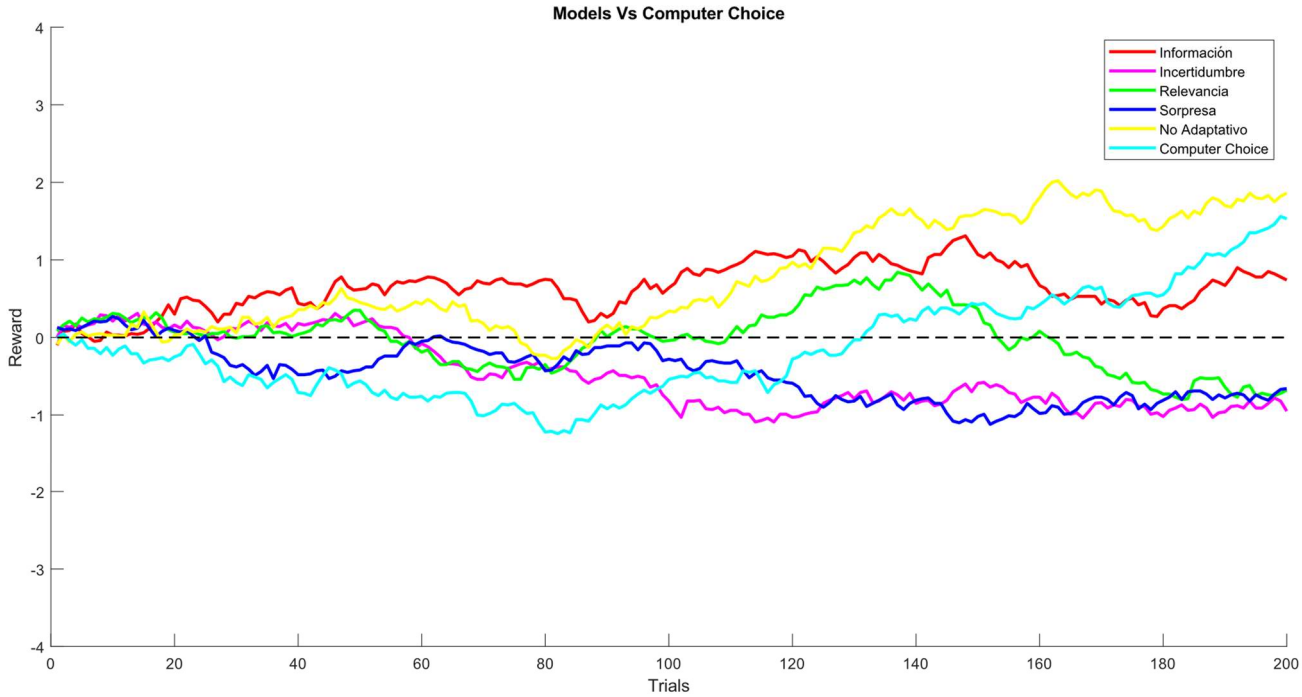

Figura 11. *Media de las recompensas de los modelos y el agente dinámico Computer Choice cuando se enfrentan cada uno a Computer Choice. Los modelos representados obtienen recompensa al elegir distinto resultado que el Computer Choice.*

El proceso mediante el que lanzamos las comparativas de la recompensa de los modelos frente a un agente dinámico es “mp_run_bads_versus_models_dyn_reward.m”

Por último, realizamos una comparación de la recompensa de los modelos contra el agente “Computer Choice” frente a las pruebas que hicieron los humanos contra el agente computacional. Mientras que para las anteriores figuras habíamos ejecutado 100 veces cada modelo, para esta comparación hemos ejecutados 30 veces cada modelo para que la prueba sea equivalente a los resultados de los 30 enfrentamientos que tenemos de los sujetos del estudio realizado por la escuela de medicina de Yale.

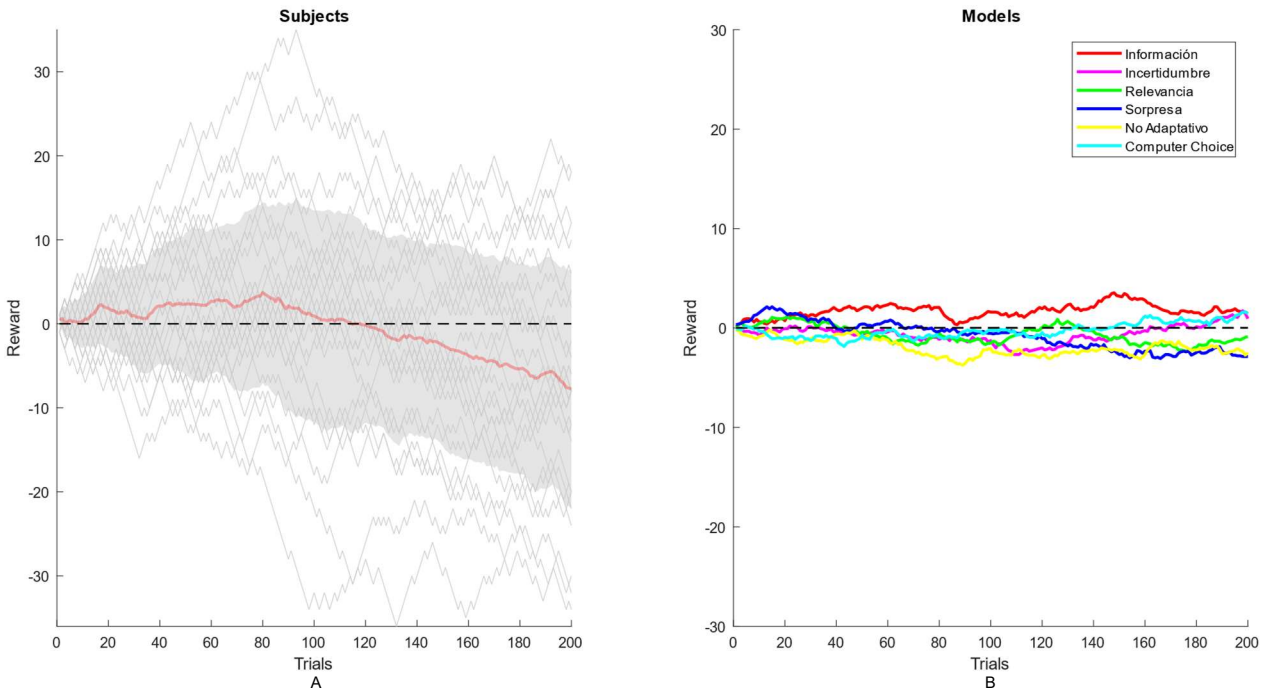

Figura 12. *Comparación de la media de la recompensa obtenida por los sujetos contra el agente computacional (A) contra las recompensas de los modelos cuando se enfrentan cada uno a Computer Choice (B). A la izquierda tenemos la representación de la recompensa de los sujetos, donde las líneas grises representan la media de la recompensa de cada uno de ellos y la línea roja la media global. A la derecha se representa la recompensa frente a un modelo dinámico entre los*

modelos de reinforcement learning.

En la figura 12 vemos que los modelos presentan un comportamiento similar al de los humanos a nivel de recompensa, aunque ligeramente superior. Por tanto, podemos deducir que estos modelos RL son tan buenos a la hora de tomar decisiones como la toma de decisiones de los humanos a la hora de enfrentarse a un agente libre.

Por último, hemos creado las figuras "violin plot" (diagrama de violín) de las distribuciones de la recompensa de los sujetos contra el agente computacional y del enfrentamiento entre los "Computer Choice" (figura 13). Donde realizamos 30 ejecuciones y de estas sacamos la media y la mediana del acumulado de la recompensa a lo largo de los 200 *trials* para los sujetos y la media para el duelo entre Computer Choice.

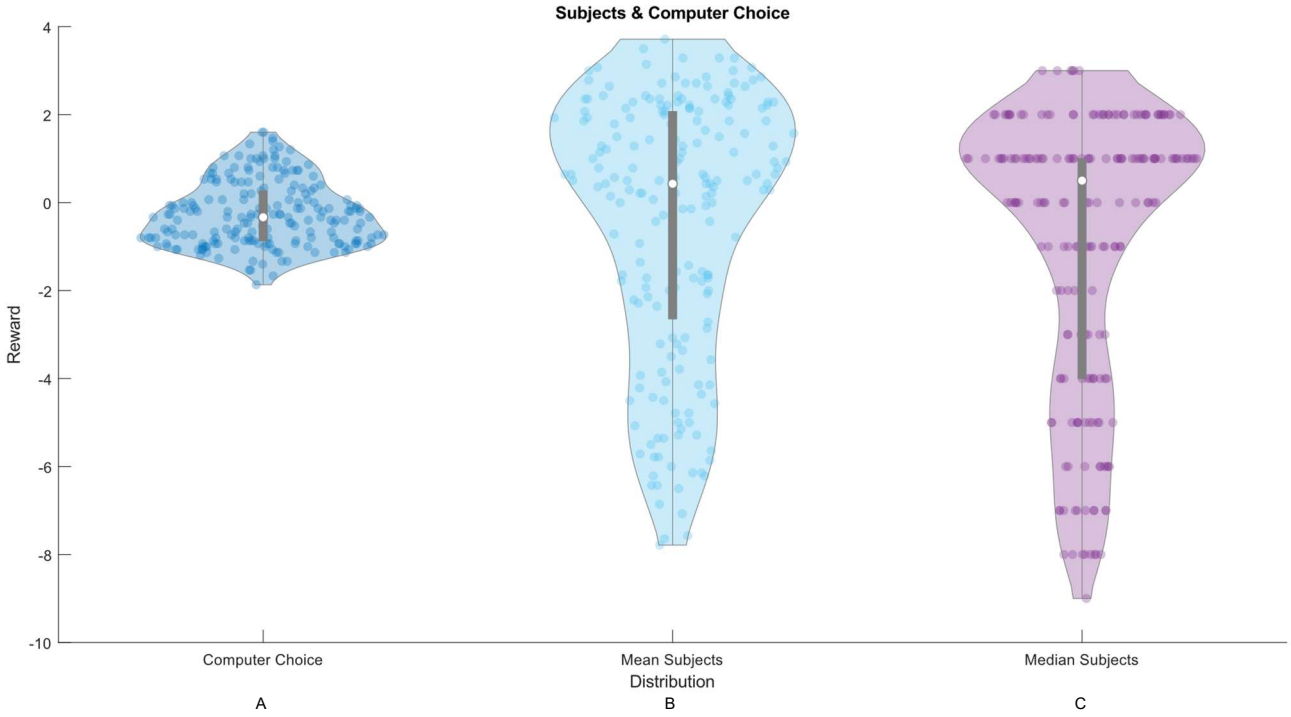

Figura 13. *Violin plot de la media del enfrentamiento entre Computer Choice (A), la media del enfrentamiento entre los sujetos y agente computacional (B) y su mediana (C). Estos gráficos muestran la forma de distribución de los datos. La anchura de densidad del gráfico indica la frecuencia de los datos. La barra negra gruesa en el centro representa el intervalo intercuartil y el punto blanco del centro la mediana.*

Referente al enfrentamientos entre los sujetos y agente computacional (figuras 13 B y C), podemos observar que la mayoría de los valores (podemos verlo porque el gráfico es más ancho) tanto de la media como la mediana de las recompensas de los sujetos se encuentran comprendidos entre 2 y 0, esto quiere decir que el enfrentamiento entre los humanos y el agente computacional está bastante igualado, aunque una parte de la distribución de los datos se encuentre en recompensas negativas que indican que los sujetos tienden a perder. Como vimos en la figura anterior, hay tres o cuatro sujetos que hicieron la prueba con recompensa muy negativa y esto se ve reflejado en la media y la mediana global, pero si atendemos a la mayoría de la distribución y la mediana (punto blanco dentro de la barra negra gruesa) observamos que la mayor distribución está cerca del cero y en términos globales los sujetos tienen una cantidad de aciertos y fallos equiprobables.
Por otra parte, en el caso del enfrentamiento entre los agentes computacionales la distribución es más homogénea quedando todos sus valores comprendidos entre -2 y 2, donde vemos que la mayor distribución de los datos se encentra muy próxima al cero debido a lo igualado del enfrentamiento entre ambos.

2.4.2 Conclusiones

Tras estos resultados podemos concluir que el modelo “Información” es el modelo que mejor funciona. Como podemos ver, tanto en la figura 6 como en la figura 7, es el que mejor consigue reproducir el comportamiento humano respecto al resto de modelos, tendiendo un porcentaje de probabilidad de elegir la misma elección que el sujeto de un 62%. Aunque como podemos ver todos los modelos de *reinforcement learning* presentan un comportamiento similar cuando se optimizan sus parámetros mediante BADS.

Respecto a la optimización de la recompensa frente a un agente estático también es el modelo “Información” el que presenta una mejor optimización consiguiendo más recompensa que el resto de los contrincantes.
Por otra parte, como vimos en la figura 10 frente a un modelo dinámico, donde la estocasticidad es mayor, los modelos tienen un rendimiento parecido donde no destaca especialmente ninguno en particular y como hemos visto finalmente a nivel de recompensa presenta un comportamiento similar al de los humanos.
En conclusión, la toma de decisiones en función de la información obtenido respecto a la línea basal, propuesto por el modelo “Información”, es el que es capaz de optimizar mejor y obtener mejores resultados en líneas generales ante diferentes escenarios, igualmente el resto de los modelos de *reinforcement learning* consiguen acercarse óptimamente al comportamiento humano y son capaces de competir tan bien como estos.

3. Estudio de los modelos logarítmicos a través de Rock-Paper-Scissors

A continuación, abordaremos las cuestiones referidas al estudio de la toma de decisiones mediante algoritmos de refuerzo computacional aplicados al juego Rock-Paper-Scissors.

3.1 Estudio de necesidades

3.1.1 Consideraciones

En esta parte crearemos bots con los distintos modelos de algoritmos implementados y los enfrentaremos, primero entre ellos, y finalmente contra otros bots a través de una competición de la plataforma Kaggle.
Kaggle es una plataforma web gratuita enfocada a la ciencia de datos, el análisis predictivo y el aprendizaje automático. Con usuarios de todo el mundo, reúne una comunidad muy grande con unos 536.000 miembros activos y destaca por sus competiciones de inteligencia artificial (IA). La competición en la que hemos formado parte, llamada Rock, Paper, Scissors, consiste en crear una IA y competir contra otros bots creados por otros participantes y ser el mejor de 100 rondas. Para ello hemos creado diferentes bots para cada uno de los algoritmos con el objetivo de comprobar cuál de ellos es menos predecible y es capaz de ganar más veces.
Los enfrentamientos se realizan entre bots que tengan una clasificación similar, esta clasificación aumenta con las victorias y disminuye con las derrotas. Esta clasificación está modelada por una N gaussiana (μ, σ^2) donde μ es la habilidad estimada, y σ^2 representa la varianza de esa estimación que disminuirá con las rondas, al inicio empezamos con una habilidad estimada de 600. Cada enfrentamiento consiste en 8 episodios y cada uno de estos tiene 1000 rondas [16].

3.2 Descripción detallada de la solución adoptada

En este apartado vamos a explicar los pasos que hemos seguido para realizar la implementación de algoritmos y comprobar su funcionamiento en el juego de "Piedra, Papel, Tijeras". Primero realizamos la implementación de los algoritmos en MATLAB y los hacemos competir entre ellos para ver cual compite mejor. A continuación, implementamos ese algoritmo en Python para poder enfrentarlos contra otros algoritmos en la competición de Kaggle "Rock Paper Scissors".

3.2.1 Modelos

Los distintos modelos que hemos utilizado son "Información", "Incertidumbre", "Relevancia", "Sorpresa" y "No Adaptativo" ya descritos en el apartado 1.2.3.

3.2.2 Creación del agente Theory Of Mind

Primero creamos los agentes en MATLAB. El objetivo de estos agentes será sacar la mayor recompensa posible en su enfrentamiento contra otro agente en el juego "Piedra, Papel, Tijeras".
El primer agente que creamos lo haremos aplicando lo que denominamos "Theory of Mind". Se utiliza ese término a la capacidad de predecir los estados mentales del agente contra el que se compite para así poder predecir su conducta y poder anticiparse a los acontecimientos. Una forma de probar la existencia de la teoría de la mente es teniendo distintas percepciones sobre una misma situación, llamado prueba de falsa creencia, donde el objetivo es verificar si puede predecir las distinciones entre las diferentes percepciones. Esta teoría va en la misma línea que la Teoría de juegos (apartado 1.3).

Por ejemplo, si el contrario eligió "tijeras" la mejor opción en base a lo que ha elegido el oponente sería elegir "piedra". Esta sería una estrategia de Theory of Mind de primer nivel. En una estrategia de Theory of Mind de segundo nivel si a priori la mejor opción según el historial de recompensas es escoger "piedra", el jugador en vez de elegir esta opción se replantearía que opción cree que escogerá el contrario, porque este podría realizar el mismo razonamiento. Esto podría llevarle a pensar que quizá el contrario pensase que el sacaría "piedra" y por tanto vaya a sacar "papel", lo cual puede llevar al jugador a escoger "tijera", que en un principio puede ser la peor opción, pero teniendo en cuenta lo que vaya a hacer el contrario, esta podría ser la mejor estrategia.
En este caso, se podría llevar a una estrategia de Theory of Mind hasta de tercer nivel, ya que tenemos tres opciones, a partir de la cual se volvería a pasar cíclicamente por cada uno de los niveles.

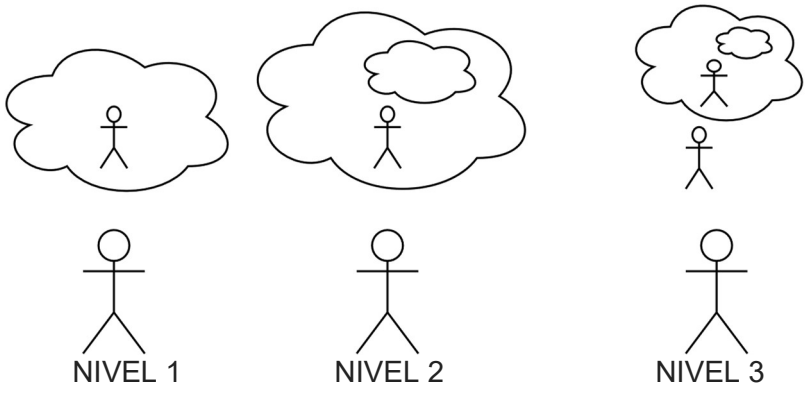

Figura 14. *En el primer nivel el sujeto1 se plantea su elección en función a lo que piensa que puede seleccionar el sujeto 2. En el nivel 2 el sujeto1 puede plantear su elección en base a lo que el sujeto 2 piensa que puede elegir el sujeto 1. En el nivel 3 el sujeto 1 se plantea su elección según lo que cree que el sujeto 3 piensa del sujeto 2 que piensa del sujeto 1.*

El agente que hemos implantado realiza la toma de decisión basándose en la decisión previa del oponente, independientemente de su propia decisión y de la señal de recompensa, que no interviene directamente, y solo actúa en la referencia de algunos modelos de meta-aprendizaje. En la selección del valor estamos eligiendo la mejor opción posible respecto a lo que eligió el adversario. Es decir, si el contrincante ha elegido "papel", el agente optará por escoger "piedra".
En este caso estamos usando la siguiente fórmula para calcular el valor:

$$\Delta V_i^n = V_i^{n-1} + \eta_i(\text{mejor}_{\text{opción}}(1 - V_i^{n-1}) - wdec \cdot Otras_{opciones} \cdot V_i^{n-1}) \quad (17)$$

Como hemos explicado en apartado anteriores ΔV_i^n es el incremento del valor asociado para cada uno de los estímulos i en la ronda n. V_i^{n-1} es el valor asociado a un estímulo previo a la ronda actual, η_i el "*learning rate*" de cada uno de los estímulos y $wdec$ el valor relativo a la tasa de aprendizaje que se utiliza para ajustarlo con la finalidad de disminuir el valor de los estímulos no elegidos para evitar el sobreajuste de los datos de entrenamiento.
En esta ecuación, se le asigna 1 a la opción deseable (mejor opción) y 0 a las opciones menos favorables (otras opciones), de manera que el valor de la opción deseable aumentará, mientras que disminuirá el valor del resto de opciones menos deseables.

Se ha implementado la función "rock_paper_scissors.m". En esta función creamos un proceso en el que dos agentes se enfrentan entre sí en 8 episodios, cada uno de ellos de 1000 rondas, lo que hace un total de 8000 enfrentamientos y devolverá la recompensa obtenida por el agente respecto al agente adversario (agente 2 en el código). Al principio del proceso definiremos el algoritmo a implementar para cada uno de los agentes ("Información", "Incertidumbre", "Relevancia", "Sorpresa" o "No Adaptativo") y será a través de la función "f_rl_agent.m" donde se implementa el algoritmo que nos devolverá la elección del agente. Finalmente se compararán las elecciones entre el agente y el agente oponente, donde el programa devolverá el acumulado de la recompensa obtenida por el agente, en caso de que gane, su recompensa será 1, en cambio, no obtendrá recompensa si pierde y se le restará 1. En caso de empate ni se le suma ni se le resta.

La llamada a la función es la siguiente:

function [choice, V, η] = f_rl_agent(R, V, opponent_choice, η, μ, meta_mechanism)

Donde los valores de entrada son:

R: Recompensa en la ronda anterior. 0 si no hay recompensa, 1 si la hay y 0,5 si hubo empate.
V: Valor asociativo de cada una de las opciones en la ronda anterior.
Opponent_choice: Elección del agente adversario. 0 si es piedra, 1 papel y 2 tijeras.
η: Tasa de aprendizaje.
μ: Parámetro que determina el cambio de ritmo del *"learning rate"*.
Meta_mechanism: Modelo del algoritmo utilizado, si es "Sorpresa", "Información"… etc.

La función devuelve los valores de la elección (choice), valor asociativo (V) y tasa de aprendizaje (η) de la ronda actual.

En este análisis no se aplicó la optimización de parámetros mediante BADS que si aplicamos en "Matching pennies". Para los valores iniciales se aplican unos parámetros fijos donde η inicial será 0 para las tres opciones disponibles. Inicializamos este valor para que a lo largo del primer episodio se actualice η y realizaremos el análisis a partir del primer episodio, por ello realizaremos el enfrentamiento para 9 episodios de los que

contaremos los últimos ocho. El valor de η no se inicializa a lo largo de los episodios, de manera que, en cada episodio partimos de lo aprendido en el episodio anterior.
Para wdec el valor es 0.9, queríamos poner un valor cercano a 1 para que penalice las malas tomas de decisiones, disminuyendo el valor de las opciones que no obtienen recompensa, pero sin que el valor estuviese en un extremo (1) para evitar que los valores de asociación de cada una de las opciones pudieran oscilar solo en los extremos (0 o 1).
Al parámetro T le asignamos el valor 1 y el valor de μ será de 0.01.

Vamos a ver el ejemplo para un ensayo del modelo "No Adaptativo" basado en Rescorla-Wagner. Las opciones que pueden escoger son piedra (0), papel (1) o tijeras (2).

Algoritmo1: Maximización de la recompensa a partir de las elecciones pasadas y la elección del contrincante.

1. De 1 a 8 episodios:
2. Inicializamos valores.
3. Para cada episodio de 1 a 1000 rondas:
4. Si ronda == 1
   R, V y opponent_choice ← valores iniciales aleatorios
   $\eta = (0,0,0)$
   μ = 0.01
   Si ronda > 1
   R, V, η y μ ← valores obtenidos en la ronda anterior
   opponent_choice ← elección previa del contrincante
   Fin-Si
5. wdec = 0.9;
6. Si opponent_choice == piedra
   best_choice = papel
   Si opponent_choice == papel
   best_choice = tijera
   Sino
   best_choice = piedra
   Fin-Si
7. Creamos las variables best_option y other_options con tres posiciones (0,0,0)
8. En best_option marcamos como 1 el estímulo elegido en best_choice.
9. En other_options marcamos como 1 los estímulos no elegidos.
10. Calculamos la tasa de aprendizaje η
11. En el modelo "No adaptativo" la tasa de aprendizaje es constante y por tanto η siempre será la misma.
12. T = 1
13. Calculamos el valor: V = V + η(best_option(1 - V) - wdec·other_options·V).
14. Calculamos las probabilidades de escoger cada una de las opciones:
    pOpt = exp(valOpt/T)-1;
    pOpt = pOpt/sum(pOpt).
15. Seleccionamos la respuesta a partir de las probabilidades:
    choice = find(cumsum(pOpts) >= rand, 1, 'first') – 1.
16. Si la elección del agente = elección del agente contrincante:
    Salida agente = empate
    Salida agente contrincante = empate.
    Si agente elige piedra y contrincante tijeras, o agente elige papel y contrincante piedra o el agente elige tijeras y el contrincante papel:
    Salida agente = gana.
    Salida agente contrincante = pierde.
    Si se dan condiciones contrarias:
    Salida agente = pierde.

Salida agente contrincante = gana
Fin-Si
17. Fin de la ronda.

3.2.3 Creación del agente con 2 sistemas

Creamos otro agente que toma las decisiones a partir de una combinación de dos sistemas expertos e independientes y finalmente decide en base a la congruencia.
El primero de ellos, visto en el punto anterior, toma las decisiones basándose en la decisión previa del oponente independientemente de la señal de recompensa. Esta elección se realiza en la función "f_rl_agent.m".
En el otro sistema, que realizamos mediante la función "f_rl_agent_tm.m", toma las decisiones basándose en la propia decisión previa modulada por la señal de recompensa. Este modelo es independiente de la decisión previa del oponente. El valor se actualiza como lo visto en la ecuación (3).
Si la mejor predicción por la selección de oponente coincide con la mejor predicción propia a partir de la señal de recompensa, se da la certeza o confianza en que es la mejor predicción para realizar la elección. Cuando hay conflicto y no coinciden, la resolución (no sesgada) resulta de una elección aleatoria.

La función utilizada es "f_rl_agent_tm".

function [choice, V, η, model_choice, model_v, model_ η] = f_rl_agent_tm(R, V, opponent_choice, η, model_choice, model_v, model_ η, μ, meta_mechanism)

Explicamos los valores de entrada son:

R: Recompensa en la ronda anterior. 0 si no hay recompensa, 1 si la hay y 0.5 si hubo empate.
V: Valor asociativo de cada una de las opciones en la ronda anterior.
Opponent_choice: Elección del agente adversario. 0 si es piedra, 1 papel y 2 tijeras.
η: Tasa de aprendizaje.
Model_choice: Elección previa del sistema modulado por la señal de recompensa.
Model_v: Valor asociativo, de cada una de las opciones, del sistema modulado por la señal de recompensa en la ronda anterior.
Model_ η: Tasa de aprendizaje, de cada una de las opciones, del sistema modulado por la señal de recompensa en la ronda anterior.
μ: Parámetro que determina el cambio de ritmo del *"learning rate"*.
Meta_mechanism: Modelo del algoritmo utilizado, si es "Sorpresa", "Información"… etc.

Vamos a ver el ejemplo para un ensayo del modelo "Información".

Algoritmo1: Maximización de la recompensa basado en el agente de dos sistemas

1. De 1 a 8 series:
2. Inicializamos valores.
3. Para cada episodio de 1 a 1000 rondas:
4. Si ronda == 1
R, V, opponent_choice, model_choice y model_v ← valores iniciales aleatorios
$\eta = (0,0,0)$
μ = 0.01
Si ronda > 1
R, V, η, model_choice, model_v, model_ η y μ ← valores obtenidos en la ronda anterior
opponent_choice ← elección previa del contrincante

Fin-Si

5. Se llama a la función f_rl_agent_tm.
6. Dentro de esta función primero llamamos al agente f_rl_agent que nos devolverá el valor de selección, choice, en base a las respuestas del contrincante (primer sistema).
7. wdec y ewdec = 0.9;
8. Calculamos la variable Información a partir del valor basado en su decisión previa y la señal de recompensa (segundo sistema):
   Información = $| 2\,(V - 0.5) |$
9. Calculamos la tasa de aprendizaje del segundo sistema:
   model_ η = model_ η + μ(informacion - model_ η)
10. Si en la ronda anterior no hubo empate calculamos el valor del segundo sistema:
    model_v = model_v + R·model_ η (model_option(R - model_v) - wdec·other_models·model_v) + (1 - R) model_η (other_models((1 - R) - model_v) - ewdec·model_option·model_v)

    Fin-Si
11. Calculamos las probabilidades de escoger cada una de las opciones:
    pModel = exp(model_v/T) - 1;
    pModel = pModel/sum(pModel);
12. Seleccionamos la respuesta:
    model_choice = find(cumsum(pModel) >= rand, 1, 'first') - 1
13. A partir de las dos elecciones choice (primer sistema) y model_choice (segundo sistema) elegimos la opción final:
    Si choice == model_choice
        Elección_final = choice
    Sino
        Elección_final = Se elige aleatoriamente
    Fin-Si
14. La función f_rl_agent_tm devolverá los valores choice, V, η, model_choice, model_v y model_ η.
15. Si elección del agente = elección del agente contrincante
    Salida agente = empate
    Salida agente contrincante = empate.

    Si agente elige piedra y contrincante tijeras, o agente elige papel y contrincante piedra o el agente elige tijeras y el contrincante papel:
    Salida agente = gana.
    Salida agente contrincante = pierde.

    Si se dan condiciones contrarias:
    Salida agente = pierde.
    Salida agente contrincante = gana.

    Fin-Si
16. Fin de la ronda volvemos al punto 3 y pasamos a la siguiente ronda.

3.2.4 Creación agentes Kaggle (Python)

Creamos estos dos agentes en Python para poder introducirlos en el concurso “Rock, Paper, Scissors” de Kaggle. Para ello hemos utilizado dos programas:

PyCharm: Se trata de un potente entorno de desarrollo integrado orientado a Python donde hemos realizado la conversión a Python de los programas que teníamos en Matlab y donde hemos podido ejecutarlos y comprobar que su comportamiento era idéntico.

Jupyter Notebook: Es una aplicación web de código abierto que permite crear y compartir contenido. Permite crear nuevos cuadernos que son documentos JSON que

contienen una lista de celdas ordenadas de entrada o salida donde almacenaremos el código y guardaremos con la extensión "ipynb".

Lo utilizamos para hacer la simulación de enfrentar a un agente contra otro de la competición de Kaggle. Para ello bajamos las librerías de https://github.com/Kaggle/kaggle-environments donde se encuentra la librería rps que nos permitirá emular los enfrentamientos.
A continuación, creamos los agentes para cada uno de los modelos. Un agente debe devolver el valor de la opción elegida (0 piedra, 1 papel, 2 tijeras).
Estos agentes los probamos en el Jupyter Notebook para comprobar su funcionamiento y a continuación los subimos a la plataforma de Kaggle para que se enfrenten al resto de contrincantes.

3.3 Justificación de la solución adoptada

3.3.1 Resultados

A continuación, veremos los resultados obtenidos en la maximización de la recompensa de los diferentes modelos en el juego de estrategia mixta “Piedra, Papel, Tijeras”.
El proceso por el cual se generan los resultados de cada uno de los modelos es “rps_statistics.m”.

Este fue el primer el estudio que realizamos. En este juego de estrategia la estocasticidad e incertidumbre es mayor al disponer de más opciones y otros escenarios como el empate.
Para disminuir el peso de la aleatoriedad y poder comparar las tendencias se ha aumentado el número de realizaciones de enfrentamientos entre los agentes. A las 8000 rondas en las que se enfrentan, ejecutamos 100 veces cada enfrentamiento para poder ver las tendencias de cada uno de ellos. Previamente se ejecuta un episodio de 1000 rondas en donde se estabiliza el valor de η.
En las siguientes figuras, 15 a 17, vamos a ver los enfrentamientos entre los diferentes modelos para comprobar su comportamiento. En estas figuras tenemos los modelos divididos en dos tipos: los modelos “Theory of Mind” y los modelos en base a dos sistemas de elección.

Por una parte, tenemos los modelos de reinforcement learning, basado en “Theory of Mind”, donde la selección del valor se tomará en base a lo que haya elegido el contrario.
En la figura 15 mostramos el enfrentamiento entre estos modelos, donde se muestra el acumulado de las recompensas de los distintos duelos entre ellos. En esta figura, observamos que el modelo que hemos planteado, “Información”, es el que obtiene mejores resultados al enfrentarse a los demás modelos. Este modelo, como hemos comentado anteriormente, evoluciona su tasa de aprendizaje en función de la información respecto a la línea basal (0.5). En este caso el “*learning rate*” aumenta tanto en correlaciones positivas en las que los valores se acerquen a 1, como en correlaciones negativas donde los valores se acerquen a 0.
Vemos también que los modelos “Incertidumbre” y “Sorpresa” que actúan bajo la misma filosofía, en la que ambos aumentan el “*learning rate*” bajo un componente sorpresivo, tienen resultados muy semejantes. Ambos son mejores que los modelos de “Relevancia” y “No Adaptativo” y al enfrentarse entre ellos tienen un enfrentamiento muy igualado con el acumulado de la recompensa cercano a 0.
El modelo “No adaptativo”, cuyo “*learning rate*” es constante es el que peor rendimiento obtiene de todos.

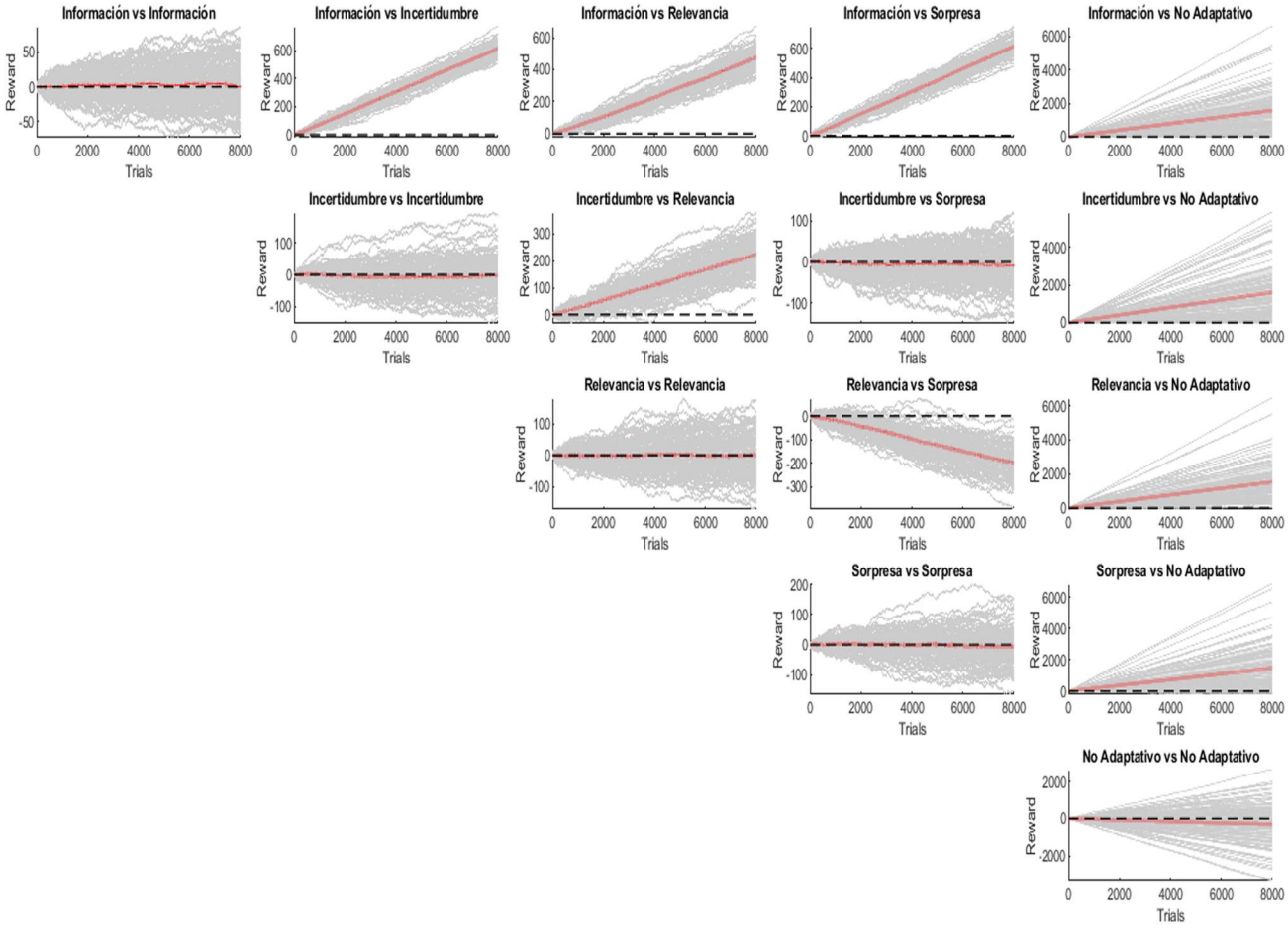

Figura 15. *Comparación del acumulativo de la recompensa entre modelos RL de Theory of Mind. Las líneas grises muestran las medias del acumulativo de la recompensa de cada una de las 100 ejecuciones en los 8000 trials. La línea roja muestra la media global de todas las ejecuciones. Cada fila corresponde a un modelo, que es el modelo que se toma como referencia para valorar los datos, y cada columna corresponde a un modelo adversario.*

Por otra parte, tenemos los modelos de aprendizaje por refuerzo de dos sistemas, que toman las elecciones a partir de una combinación del modelo "Theory of Mind", que considera las decisiones del oponente, y de un modelo cuya predicción se realiza en base a la señal de recompensa.
Cuando estos dos sistemas coinciden en su mejor predicción para realizar la elección el modelo realizará esa elección. Cuando no coinciden los sistemas, la elección del modelo será elegida al azar.

A continuación, realizamos la comparación entre los agentes con dos sistemas de decisión y los agentes "Theory of Mind". En la figura 16 podemos observar que el agente con dos sistemas de decisión obtiene mayor recompensa frente al agente "Theory of Mind" cuando se enfrenta a un mismo modelo ("Información", "Relevancia"…etc.).
Observamos que el modelo "Información" con 2 sistemas de decisión es el modelo que tiene un mejor funcionamiento frente al resto de agentes "Theory of Mind". Al igual que en el apartado anterior, vemos que los modelos "Incertidumbre" y "Sorpresa", en los que la tasa de aprendizaje aumenta ante algo inesperado, presentan un comportamiento similar en las que para el modelo de 2 sistemas solo presentan peor resultado ante el modelo "Theory of Mind" de "Información".

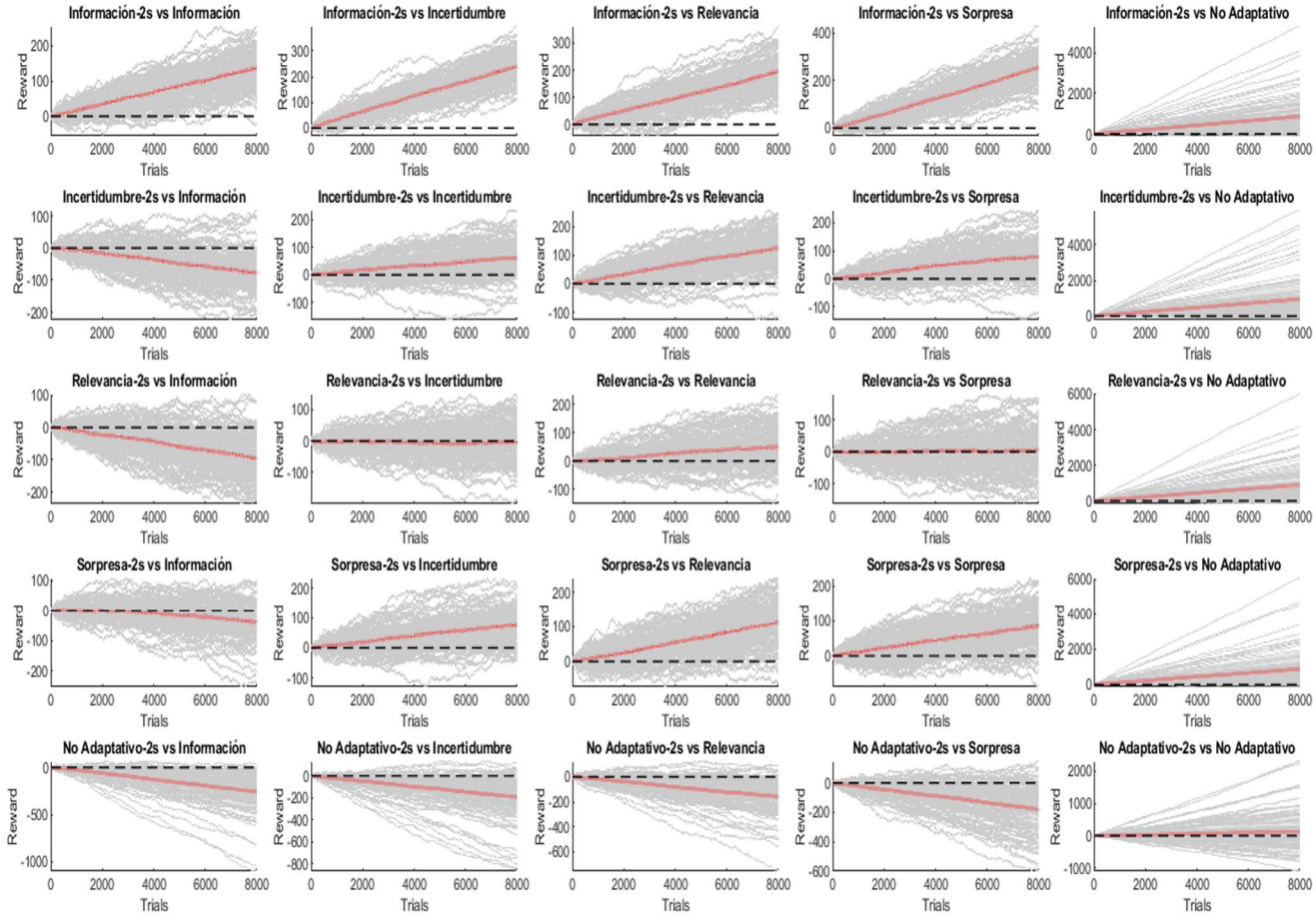

Figura 16. *Comparación del acumulativo de la recompensa entre modelos RL de 2 sistemas contra los modelos RL con Theory of Mind*

Como hemos mencionado anteriormente, en el modelo de dos sistemas cuando hay congruencia entre ambos sistemas el modelo elige esa respuesta. En la tabla 2 mostramos las estadísticas que tienen los modelos de 2 sistemas frente a los modelos "Theory of Mind", en esta tabla, mostramos los porcentajes de congruencia del sistema.

% Congruencia: Muestra el porcentaje de *trials* en que ambos sistemas coinciden frente al total de *trials*. En la tabla observamos que la cantidad de veces en que coinciden ambos sistemas es inferior al 50% y que en mayor medida hay incongruencia.
Cuando hay incongruencia, es decir, que los sistemas no coinciden en su mejor predicción, al elegir la respuesta aleatoriamente el porcentaje de respuesta en que el sistema obtiene recompensa, empata o pierde es de un 33% para cada uno de los posibles. Por tanto, es cuando hay congruencia donde se determina si el modelo de dos sistemas maximiza su recompensa frente al modelo "Theory of Mind". En la tabla comprobamos, dentro de los casos en los que los sistemas coinciden, cuantas veces se obtiene recompensa frente a las demás respuestas, ya que, cuando hay congruencia es cuando se determinan los resultados del modelo.
% Congr. R vs E: Se trata del porcentaje de *trials* que obtienen recompensa frente aquellos que erran en su elección cuando ambos sistemas coinciden. En este caso vemos que cuando el modelo de 2 sistemas gana su enfrentamiento frente al otro modelo este porcentaje está por encima del 50% y cuando pierde se encuentra por debajo.
% Congr. R vs T: Es el porcentaje de *trials* que obtienen recompensa frente aquellos que empatan en su elección cuando ambos sistemas coinciden. Podemos observar en la tabla que la proporción de recompensa frente al empate es siempre mayor.
El porcentaje de la recompensa frente al error y el empate cuando hay incongruencia es siempre alrededor de un 50%, con fluctuaciones mínimas, ya que la elección es al azar.

Modelo de 2 sistemas vs Modelo “Theory of Mind”	% Congruencia	% Congr. R vs E	% Congr. R vs T
Información-2s vs Información	36,83	53,03	64,39
Información-2s vs Incertidumbre	37,39	55,31	65,02
Información-2s vs Relevancia	37,45	54,39	59,93
Información-2s vs Sorpresa	37,40	55,41	64,08
Información-2s vs No Adaptativo	43,54	69,94	66,58
Incertidumbre-2s vs Información	37,04	48,48	61,41
Incertidumbre-2s vs Incertidumbre	37,50	51,39	62,72
Incertidumbre-2s vs Relevancia	37,69	52,81	59,88
Incertidumbre-2s vs Sorpresa	37,65	51,55	62,09
Incertidumbre-2s vs No Adaptativo	43,03	69,84	67,17
Relevancia-2s vs Información	35,50	47,62	56,54
Relevancia-2s vs Incertidumbre	35,94	49,86	58,51
Relevancia-2s vs Relevancia	36,25	51,38	57,68
Relevancia-2s vs Sorpresa	35,98	50,07	58,30
Relevancia-2s vs No Adaptativo	39,53	64,20	61,96
Sorpresa-2s vs Información	36,36	48,77	59,94
Sorpresa-2s vs Incertidumbre	36,73	51,62	61,74
Sorpresa-2s vs Relevancia	36,81	52,76	59,17
Sorpresa-2s vs Sorpresa	36,69	51,92	61,32
Sorpresa-2s vs No Adaptativo	41,93	68,43	65,45
No Adaptativo -2s vs Información	33,09	43,24	50,72
No Adaptativo-2s vs Incertidumbre	33,25	43,32	50,72
No Adaptativo-2s vs Relevancia	31,45	44,57	50,36
No Adaptativo-2s vs Sorpresa	32,06	44,89	50,85
No Adaptativo-2s vs No Adaptativo	32,93	55,58	52,64

Tabla 2. *Estadísticas de los enfrentamientos de los modelos de 2 sistemas frente a los modelos “Theory of Mind” cuando hay congruencia.*

Finalmente hemos enfrentado a los modelos de 2 sistemas entre ellos. Cuando se enfrentan al mismo modelo, podemos ver en la figura 16, que el enfrentamiento es muy igualado. Al igual que en las figuras anteriores, comprobamos que “Información” es el que tiene mejor funcionamiento frente al resto de sistemas, mientras que el “No Adaptativo” es el que peor maximiza la recompensa. Por tanto, vemos que se obtiene un mejor rendimiento bajo la filosofía de actualizar el “*learning rate*” en base a la existencia de correlaciones sistemáticas.

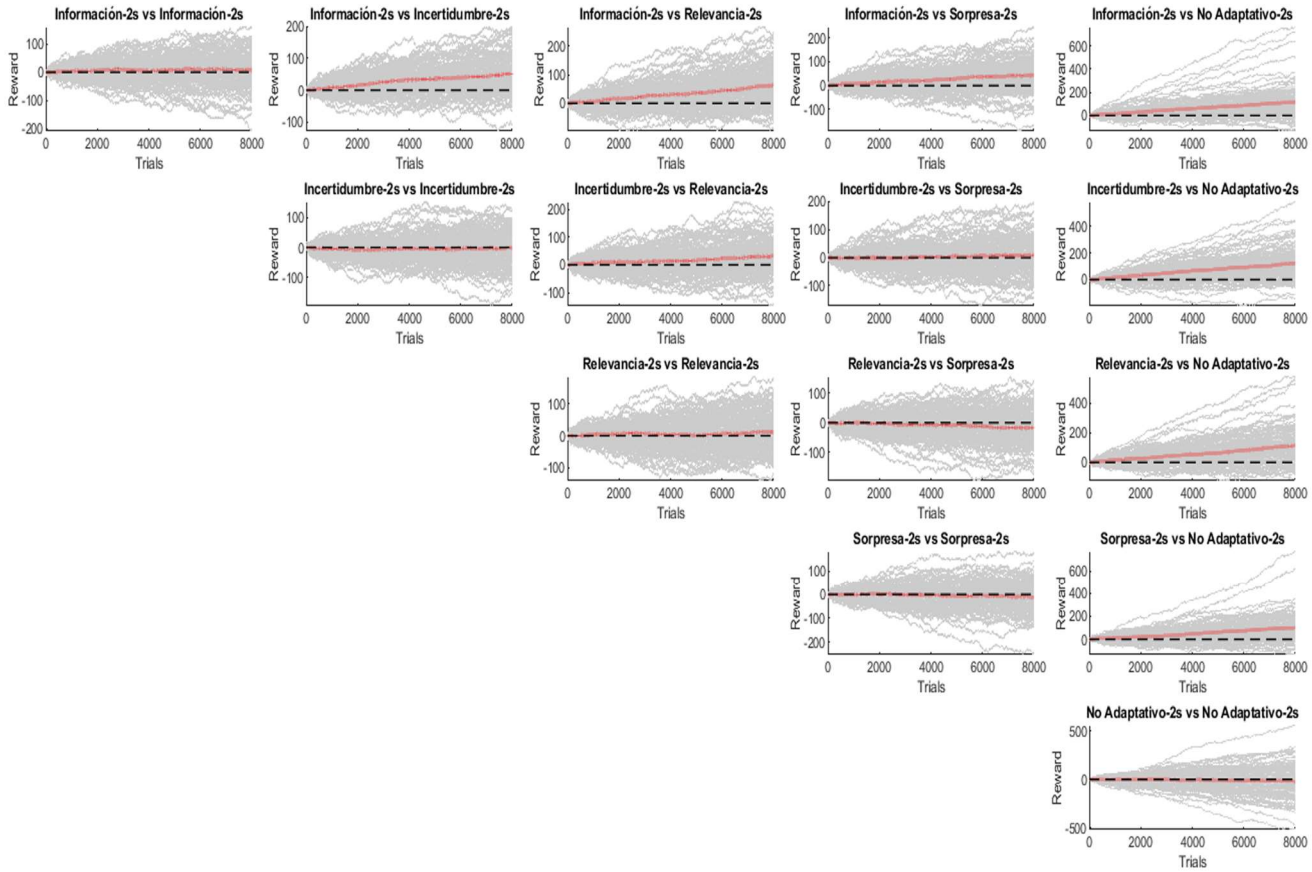

Figura 17. *Comparación del acumulativo de la recompensa entre modelos RL de 2 sistemas.*

A continuación, mostramos las estadísticas recogidas de los enfrentamientos cuando hay congruencia. En este caso observamos que la proporción de recompensa frente al empate es menor que en los enfrentamientos contra "Theory of Mind", aumentando las veces que empatan cuando se enfrentan modelos de dos sistemas entre sí. Aunque los enfrentamientos están igualados, podemos ver la tendencia del modelo por la proporción de la recompensa frente al error.

Modelo de 2 sistemas vs Modelo de 2 sistemas	% Congruencia	% Congr. R vs E	% Congr. R vs T
Información-2s vs Información-2s	36,43	50,02	54,48
Información-2s vs Incertidumbre-2s	36,70	51,21	55,12
Información-2s vs Relevancia-2s	36,58	51,60	53,94
Información-2s vs Sorpresa-2s	36,66	50,88	54,15
Información-2s vs No Adaptativo-2s	37,33	53,52	53,23
Incertidumbre-2s vs Información-2s	36,84	48,75	53,96
Incertidumbre-2s vs Incertidumbre-2s	37,03	50,00	54,38
Incertidumbre-2s vs Relevancia-2s	36,98	50,61	53,58
Incertidumbre-2s vs Sorpresa-2s	36,84	49,83	53,52
Incertidumbre-2s vs No Adaptativo-2s	37,15	52,67	52,30
Relevancia-2s vs Información-2s	35,68	48,29	52,41
Relevancia-2s vs Incertidumbre-2s	35,76	49,42	52,74
Relevancia-2s vs Relevancia-2s	35,90	49,90	52,23
Relevancia-2s vs Sorpresa-2s	35,78	49,39	52,47
Relevancia-2s vs No Adaptativo-2s	36,36	52,20	51,75
Sorpresa-2s vs Información-2s	35,84	48,90	53,17
Sorpresa-2s vs Incertidumbre-2s	36,14	50,12	53,96
Sorpresa-2s vs Relevancia-2s	36,13	50,63	52,99
Sorpresa-2s vs Sorpresa-2s	36,20	50,20	53,81
Sorpresa-2s vs No Adaptativo-2s	36,58	53,05	52,73
No Adaptativo -2s vs Información-2s	32,31	46,05	49,60
No Adaptativo-2s vs Incertidumbre-2s	33,03	46,29	49,43
No Adaptativo-2s vs Relevancia-2s	32,63	47,56	49,61
No Adaptativo-2s vs Sorpresa-2s	32,89	47,11	49,85
No Adaptativo-2s vs No Adaptativo-2s	33,83	50,08	49,09

Tabla 3. *Estadísticas de los enfrentamientos entre los modelos de 2 sistemas cuando hay congruencia.*

3.3.2 Resultados en la plataforma Kaggle

Por último, mostramos los resultados que obtuvimos de la participación en el concurso de "Rock, Paper, Scissors" que se realizó en la plataforma Kaggle (*https://www.kaggle.com/competitions/rock-paper-scissors*). Este fue el punto de partida de nuestro estudio y aunque los resultados no fueron demasiado buenos dio lugar a una serie de mejoras como la creación de los modelos con dos sistemas y el modelo información cuyos resultados mejoraron a los de los modelos iniciales.

Lanzamos varias versiones de los agentes. La puntuación inicial con la que partían los agentes era de 600, esta aumentaba si ganaban duelos y disminuía si los perdían.

Primero subimos unas versiones iniciales que correspondían a los agentes "Theory of Mind".

La siguiente versión incluía los agentes "Theory of Mind" con algunas modificaciones e hicimos varias subidas de estos agentes para ver su evolución y poder observar cuales tenían más éxito. Estas se trataban de las primeras versiones que hicimos de los modelos.

Hay que tener en cuenta que los enfrentamientos eran contra agentes aleatorios, normalmente con una puntuación similar a la nuestra. Es normal que un mismo agente pudiera tener puntuaciones distintas, ya que, no se enfrentan a los mismos adversarios y los escenarios pueden ser muy diferentes. Por ello realizamos varias subidas de cada uno de los agentes. El agente "Relevancia fue el que mejor funcionó en esta competición. En términos generales funcionaron mejor la última versión que subimos de los agentes "Theory of Mind".

Modelo	Puntuación
"Relevancia" – 2 sistemas	782.5
"Relevancia" – 2 sistemas	778.1
"Relevancia" – 2 sistemas	724.1

Tabla 4. *Mejores puntuaciones obtenidas en el concurso de Kaggle.*

3.3.3 Conclusiones

En las pruebas realizadas entre los enfrentamientos de modelos hemos comprobado que el modelo "Información" de dos sistemas es el que sale más victorioso ante diferentes estrategias. Si añadimos estas pruebas a las realizadas con el estudio de "matching pennies" podemos afirmar que este agente es el que mejor optimiza la recompensa y se acerca más al comportamiento humano, por tanto, podemos concluir, que el aprendizaje basado en la información, donde las correlaciones entre lo que se espera y lo que se obtiene interviene en la dinámica, positiva o negativa, de la tasa de aprendizaje a una determinada elección, son mejores predictores que las estrategias basadas en la sorpresa o aquellas que su aprendizaje evoluciona solo en función del valor y la recompensa obtenida.

También hemos comprobado, tanto en las comparaciones como en la competición realizada por Kaggle, que los agentes con dos sistemas, donde se tiene en cuenta la señal de recompensa, funcionan mejor contra diferentes agentes de variadas estrategias que aquellos que solo contemplan elegir la mejor opción.

Las diferencias entre los dos juegos es que en "matching pennies" se ha optimizado por BADS mientras que en este no.

Como trabajo futuro podríamos investigar qué pasa con “matching pennies” si se trabaja de la misma manera que en el juego de “Piedra, Papel, Tijeras”. ¿Existirían los mismos resultados? ¿Qué modelo predeciría mejor las respuestas de los sujetos?

Creemos que en términos generales el modelo “Información” sería el mejor (y quizás no se aleja demasiado de la mejor solución encontrada por BADS que hace una búsqueda exhaustiva) pero que al utilizar BADS encuentra soluciones en regiones mas aisladas que acaban pareciéndose mucho independientemente de los modelos.

II. PLIEGO DE CONDICIONES

1. Definición y alcance (objeto)

El objeto de esta sección reside en definir las condiciones mínimas requeridas para la realización del proyecto de aprendizaje por refuerzo aplicado a la teoría de juegos para el entendimiento del comportamiento humano, así como en el estudio de la maximización de la recompensa en los distintos modelos aplicados.

2. Condiciones generales

Para la correcta realización del trabajo hay que conocer y seguir la normativa sobre las disposiciones mínimas de seguridad y de salud relativas al trabajo con equipos que incluyen pantallas de visualización formulada en la directiva del consejo de la unión europea 90/270/CEE. Además de la seguridad y salud del trabajador hay que conocer también el uso adecuado de los datos utilizados, ya que pueden contener información sensible, para ello habrá que tener en cuenta la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.

Acerca de la directiva 90/270/CEE en la Sección I, Disposiciones Generales, Artículo 2 se definen los conceptos de pantalla de visualización, puesto de trabajo y trabajador a efectos de la Directiva:

"*a) pantalla de visualización: una pantalla alfanumérica o gráfica, independientemente del método de representación visual utilizado;*

b) puesto de trabajo: el conjunto que consta de un equipo con pantalla de visualización provisto, en su caso, de un teclado o de un dispositivo de adquisición de datos y/o de un programa que garantice la interconexión hombre/máquina, de accesorios opcionales, de anejos, incluida la unidad de disquetes, de un teléfono, de un módem, de una impresora, de un soporte de documentos, de una silla y de una mesa o superficie de trabajo, así como el entorno laboral inmediato;

c) trabajador: cualquier trabajador, con arreglo a la letra a) del artículo 3 de la Directiva 89/391/CEE, que habitualmente y durante una parte relevante de su trabajo normal utilice un equipo con pantalla de visualización.[17]"

Los requisitos mínimos referidos al equipo, entorno e interconexión hombre/máquina se encuentran clasificados en el Anexo de disposiciones mínimas de la directiva 90/270/CEE:

2.1 Equipo

- Observación general

El uso del equipo no debe suponer una amenaza de riesgo para los trabajadores.

- Pantalla

La pantalla deberá presentar una imagen clara y estable, sin destellos o reflejos que puedan molestar la visibilidad del usuario. Los caracteres deberán estar bien definidos, de manera que sean claros y de tamaño adecuado disponiendo del suficiente espacio entre los caracteres y renglones. El usuario deberá poder ajustar de manera sencilla la luminosidad y el contraste de la pantalla, así como la orientación e inclinación de la misma para poder adaptarse a las condiciones del entorno y trabajar de la manera más cómoda posible.

- Teclado

El teclado deberá ser inclinable e independiente de la pantalla para facilitar una postura cómoda en el trabajador evitando el cansancio en brazos y manos, siendo necesario además que haya espacio suficiente para que pueda apoyarlos entre el usuario y el teclado. La disposición del teclado deberá tender a facilitar el uso del mismo. Las teclas deberán ser legibles desde la posición normal de trabajo.

- Mesa o superficie de trabajo

La superficie de trabajo deberá ser poco reflectante, de dimensiones adecuadas que permitan la colocación flexible de la pantalla, teclado, así como documentos u otros materiales necesarios, además de permitir a los trabajadores a estar en una posición cómoda.

- Asiento de trabajo

El asiento de trabajo deberá ser cómodo, estable y que otorgue al usuario de libertad de movimiento. La altura del asiento deberá ser regulable, así como el respaldo deberá ser reclinable y ajustable. Deberá de otorgarse un reposapiés a quienes lo deseen.

2.2 Entorno

- Espacio

La dimensión del espacio de trabajo deberá ser suficiente para permitir cambios de postura y de movimientos durante la realización del trabajo.

- Iluminación

La iluminación, tanto general como especial, deberán asegurar una suficiente luz y contraste adecuado entre la pantalla y usuario. El lugar de trabajo deberá tener el acondicionamiento adecuado que evite los deslumbramientos y reflejos en la pantalla u otra parte del equipo.

- Reflejos y deslumbramientos

Deberán disponerse los puestos de trabajo de manera que evite que fuentes de luz, como ventanas, provoquen deslumbramiento directo o reflejos en la pantalla. Las ventanas dispondrán de un dispositivo de cobertura adecuado y regulable para atenuar la luz del día que ilumine el puesto de trabajo.

- Ruido

El ruido producido por los equipos del puesto de trabajo no debe perturbar la palabra ni la atención del trabajador.

- Calor

Los equipos utilizados en el puesto de trabajo no deben producir un calor añadido que puedas molestar a los trabajadores.

- Emisiones

Para la protección y seguridad de los trabajadores deberá reducirse a niveles insignificantes toda radiación, a excepción de las radiaciones del espectro electromagnético visible

- Humedad

La humedad tiene que crearse y mantenerse a niveles aceptables.

2.3 Interconexión Ordenador/Hombre

Deberán tenerse en cuenta los siguientes factores para la elaboración, elección, compra, modificación de pantallas, así como otras tareas que requieran de pantallas de visualización:

- Adaptar el programa a la realización de la tarea.
- El programa deberá estar adaptado al nivel de formación y experiencia de los trabajadores.
- Los sistemas deberán indicar a los usuarios información sobre su desarrollo en un formato y ritmo adaptado a los operadores.
- Los principios de ergonomía deberán aplicarse en particular al tratamiento de la información por parte del hombre.

3. Condiciones específicas

En el presente apartado se detallarán las condiciones específicas de carácter técnico, estas especificaciones hacen referencia tanto a los elementos físicos que constituyen el sistema informático (hardware) como al conjunto de programas (software) utilizados durante el desarrollo y realización del trabajo.

3.1 Hardware

A continuación, se detallarán las características de los equipos que se han utilizado durante el trabajo. Las características del equipo de trabajo utilizado por escritorio remoto, para la agilización de optimización de datos, y la más recomendable por poder hacer uso de la computación en paralelo son:

- CPU: AMD Ryzen Threadripper PRO 3995WX 4.3 GHz 64-Cores (128 Threads)
- Tarjeta gráfica: 2 × Quadro RTX 8000 Quadro 4608-Cores 48 GB
- RAM: 512 GB
- Unidades de disco duro: 2 × nvme de 1 TB, 4 × nvme de 2 TB
- Sistema Operativo: Linux 5.10.0-1052-oem (64 bits)

Incluimos las características del otro equipo que hemos utilizado. Se recomienda utilizar un equipo con las características del anterior ya que en este modelo los procesos eran mucho más costosos y lentos, aunque lo hemos utilizado para poder lanzar más procesos a la vez. En este equipo no era posible usar la computación en paralelo.

- CPU: Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70 GHz
- Tarjeta gráfica: Intel(R) HD Graphics 620

- RAM: 16,0 GB
- Sistema Operativo: Windows 10 Home Versión 21H2

3.2 Software

En este apartado vamos a enumerar los programas de software necesarios para la realización del proyecto:

Nombre	Versión	Función
Matlab	R2019a	Plataforma de programación con un lenguaje propio (M) que destaca por sus prestaciones de cálculo numérico. Se ha utilizado para la programación y procesamiento de los modelos.
PyCharm	2020.2.4	Entorno de Desarrollo integrado orientado a Python, utilizado para pasar a Python los modelos de Rock-Paper-Scissors para su prueba en la plataforma Kaggle.
Jupyter Notebook	6.1.4	Aplicación web de código abierto que permite crear nuevos cuadernos que contienen una lista de celdas ordenadas de entrada o salida donde almacenaremos el código y guardaremos con la extensión "ipynb". Utilizado para probar los enfrentamientos entre agentes para Kaggle
BADS (Bayesian Adaptive Direct Search)	v1.0.6	Algoritmo de optimización bayesiana para la resolución de problemas de optimización complejos. Es ideal para el ajuste de modelos computacionales.
Libreria kaggle - environments	1.9.11	Librerías necesarias para la emulación de la competición Rock-Paper-Scissors de Kaggle
Filezilla	3.55.1	Aplicación que consta de un cliente y un servidor FTP. Necesario para intercambiar los archivos entre el PC y el escritorio remote.

Tabla 5. *Listado de las aplicaciones y librerías específicas para el desarrollo del proyecto.*

Además, habrá que añadir las fuentes de los procesos que se podrán descargar a través de GitHub.

III. PRESUPUESTO

1. Introducción

En este apartado detallaremos los diferentes importes para la realización del proyecto. Los precios de algunos materiales son estimados y su coste podría variar. Los materiales descritos para la realización del proyecto serán aquellos con los que se ha trabajado,
Además de incluir los materiales para el montaje de una computadora en la que el proceso de optimización de los modelos tenga un buen rendimiento, se incluye también el equipo adicional utilizado para hacer tareas en paralelo o pequeñas comprobaciones.
Es importante que el rendimiento de ejecución de los modelos sea óptimo, ya que, con un equipo inadecuado podría implicar un aumento del coste de trabajo que supondría un aumento de las horas trabajadas.

2. Costes de hardware

Unidad	Denominación	Cantidad	Precio	Total
Ensamblaje del ordenador de sobremesa				
Ensamblaje del dispositivo hardware para el desarrollo e implementación de los modelos de Aprendizaje por Refuerzo				
Materiales				
U	Teclado Logitech K120 Teclado con Cable USB Negro	2	10,99 €	21,98 €
U	Ratón Óptico Inalámbrico 2.4GHz NGS Fog Black	2	5,19 €	10,38 €
U	Placa Base ASUS Prime TRX40-Pro S - AMD ATX sTRX4 Ryzen	1	445,10 €	445,10 €
U	CPU AMD Ryzen Threadripper PRO 3995WX	1	8.104,97 €	8.104,97 €
U	RAM Kingston Branded Memory 64GB DDR4-3200MHz Reg ECC Module KTD-PE432	8	479,59 €	3.836,72 €
U	Samsung 970 EVO Plus 2TB SSD NVMe M.2	4	120 €	480 €
U	Samsung 970 EVO Plus 1TB SSD NVMe M.2	2	62,99	125,98 €
U	Asus ROG Strix LC 360 Kit de Refrigeración Líquida	1	211,99 €	211,99 €
U	NVIDIA-tarjeta gráfica Quadro RTX 8000, 48GB	2	3.387,14 €	6.774,28 €
U	Caja PC Gamer Diamond ARGB Torre Mediana ATX – Frontal Diamante Plexiglás y Pared Lateral de Vidrio Templado – 4 Ventiladores LED RGB direccionables	1	75,99 €	75,99 €

U	HP Notebook Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz	1	599,99 €	599,99 €
U	Fuente de alimentación Seasonic Focus-GX 1000 1000W 80 Plus Gold Modular	1	169,99 €	169,99 €
U	HP M24fe - Monitor de 24" Full HD	1	134,00 €	134,00 €
Mano de obra				
h	Técnico informático	2	13,59 €	27,18 €
Costes directos complementarios				
%		2%	21.018,55 €	420,37 €
Total capítulo				
U		1	21.438,92 €	21.438,92 €

Tabla 6. *Presupuesto de la partida del ensamblaje del ordenador de torre desglosado en materiales, mano de obra y costes directos complementarios (2%).*

3. Costes software

Unidad	Denominación	Cantidad	Precio	Total
Instalación y configuración del software				
Instalación de los programas y librerías específicas en el ordenador de torre				
U		1	120,60	120,60
Materiales				
U	Matlab Educativa Perpetual Licence	1	500,00 €	500,00 €
U	Linux 64 bits	1	0,00 €	0,00 €
U	PyCharm	1	0,00 €	0,00 €
U	Jupyter Notebook	1	0,00 €	0,00 €
U	Bayesian Adaptive Direct Search (BADS)	1	0,00 €	0,00 €
U	Librería Kaggle - environments	1	0,00 €	0,00 €
U	Filezilla	1	0,00 €	0,00 €
Mano de obra				
h	Técnico Informático	2.5	13,59 €	33,97 €
Costes directos complementarios				
%		2%	533,97	10,68 €
Total capítulo				
U		1	544,65 €	544,65 €

Tabla 7. *Presupuesto de la partida de la instalación del software específico en el ordenador de torre desglosado en materiales, mano de obra y costes directos complementarios (2%).*

4. Costes del desarrollo del proyecto

Unidad	Denominación	Cantidad	Precio	Total
Desarrollo del proyecto				
Presupuesto del desarrollo de los modelos de aprendizaje por refuerzo				
Mano de obra				
h	Ingeniero de datos	320	16,40 €	5.248,00 €
Costes directos complementarios				
%		4%	5.248,00 €	209,92 €
Total capítulo				
U		1	5.457,92 €	5.457,92 €

Tabla 8. *Presupuesto de la partida de desarrollo del proyecto desglosado en mano de obra y costes directos complementarios (4%). Se asume que el estudio tendrá una duración aproximada de 6 semanas.*

5. Resumen del presupuesto

Capítulo	Importe
Capítulo 1. Ensamblaje del ordenador de torre	21.438,92 €
Capítulo 2. Instalación y configuración de software	544,65 €
Capítulo 3. Desarrollo del proyecto	5.457,92 €
TOTAL PRESUPUESTO DE EJECUCIÓN MATERIAL	27.441,49 €
Medios auxiliares (4%)	1097,66 €
TOTAL PRESUPUESTO EJECUCIÓN POR CONTRATA	28.539,15 €
IVA (21%)	5.993,22 €
TOTAL PRESUPUESTO BASE DE LICITACIÓN	34.532,37 €

Tabla 9. *Presupuesto de la partida del desarrollo del proyecto desglosado en mano de obra y costes directos complementarios (4%). Se asume que el proyecto tendrá una duración aproximada de 12 semanas.*

IV.ANEXOS

1. ANEXO I. RELACIÓN DEL TRABAJO CON LOS OBJETIVOS DE DESARROLLO SOSTENIBLE DE LA AGENDA 2030

Anexo al Trabajo de Fin de Grado y Trabajo de Fin de Máster: Relación del trabajo con los Objetivos de Desarrollo Sostenible de la agenda 2030

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No procede
ODS 1. **Fin de la pobreza.**				x
ODS 2. **Hambre cero.**				x
ODS 3. **Salud y bienestar.**				x
ODS 4. **Educación de calidad.**				x
ODS 5. **Igualdad de género.**				x
ODS 6. **Agua limpia y saneamiento.**				x
ODS 7. **Energía asequible y no contaminante.**				x
ODS 8. **Trabajo decente y crecimiento económico.**				x
ODS 9. **Industria, innovación e infraestructura.**	x			
ODS 10. **Reducción de desigualdades.**				x
ODS 11. **Ciudades y comunidades sostenibles.**				x
ODS 12. **Producción y consumo responsables.**			x	
ODS 13. **Acción por el clima.**				x
ODS 14. **Vida submarina.**				x
ODS 15. **Vida de ecosistemas terrestres.**				x
ODS 16. **Paz, justicia e instituciones sólidas.**				x
ODS 17. **Alianzas para lograr objetivos.**				x

Este proyecto trata sobre la optimización del comportamiento de bots aplicado a la teoría de juegos en el que mediante el aprendizaje por refuerzo implantamos diferentes algoritmos para comprobar cuál de estos mecanismos se parecen más a la toma de decisiones humana o animal, y cuál de estos es capaz de optimizar sus elecciones a través de los juegos competitivos. Esta optimización del comportamiento en la toma de decisiones de bots está claramente relacionada con el objetivo 9 de industria, innovación e infraestructuras donde el aprendizaje por refuerzo se aplica a varios campos, como el de la robótica donde puede ser utilizado para la automatización de tareas o en los vehículos autónomos donde ya se está implantando en los coches Tesla, entre otros.
Aunque este proyecto no se relaciona de forma directa con el objetivo 12, producción y consumo responsables, el estudio del aprendizaje por refuerzo se utiliza para el

tratamiento de datos y se puede aplicar a otras áreas para reducir el consumo energético a través de la información recibida pudiendo predecir qué acciones tomar para minimizar la energía a futuro.

2. ANEXO II. CÓDIGOS

Enlace del código del estudio de Matching Pennies en Github:

https://github.com/LofNaDI/TFG_RL

Enlace del código del estudio de Rock, Paper, Scissors en Github:

https://github.com/LofNaDI/TFG_RL_MP

V.BIBLIOGRAFIA

1. ROUHIAINEN, Lasse. Inteligencia artificial. *Madrid: Alienta Editorial*, 2018, p. 20-21.

2. RIVERA, Luis Felipe Sarmiento; FLÓREZ, Jorge Alexander Ríos. Bases neurales de la toma de decisiones e implicación de las emociones en el proceso. *Revista chilena de neuropsicología*, 2017, vol. 12, no 2, p. 32-37.

3. YANG, Tianming; SHADLEN, Michael N. Probabilistic reasoning by neurons. *Nature*, 2007, vol. 447, no 7148, p. 1075-1080.

4. LEE, Daeyeol, et al. Reinforcement learning and decision making in monkeys during a competitive game. Cognitive brain research, 2004, vol. 22, no 1, p. 45-58.

5. LEE, Daeyeol; MCGREEVY, Benjamin P.; BARRACLOUGH, Dominic J. Learning and decision making in monkeys during a rock–paper–scissors game. Cognitive Brain Research, 2005, vol. 25, no 2, p. 416-430.

6. HU, Jianfeng; LI, Xiaofeng; YIN, Jinghai. Learning and decision making in human during a game of matching pennies. *International Journal of Digital Content Technology and its Applications*, 2010, vol. 4, no 2.

7. OTTERLO, Martijn van; WIERING, Marco. Reinforcement learning: State-of-the-Art. Springer-Verlag Berlin Heidelberg 2012, Chapter 16.

8. MACKINTOSH, Nicholas J. A theory of attention: Variations in the associability of stimuli with reinforcement. Psychological review, 1975, vol. 82, no 4, p. 276.

9. VOGEL, Edgar H; SOTO, Fabián A; CASTRO, María E; SOLAR, Paola A. Modelos matemáticos del condicionamiento clásico: evolución y desafíos actuales. Revista Latinoamericana de Psicología, 2006, volumen 38, Nº2, 215-243

10. SERRA, Francisco Fernández; DE LA CASA RIVAS, Luis Gonzalo. Una revisión teórica de los intentos explicativos del fenómeno de la inhibición latente. Revista de psicología general y aplicada: Revista de la Federación Española de Asociaciones de Psicología, 1989, vol. 42, no 4, p. 425-439

11. ARDID, Salva; BALCARRAS, Matthew; WOMELSDORF, Thilo. “Adaptive learning” as a mechanistic candidate for reaching optimal task-set representations flexibly. BMC Neuroscience, 2014, vol. 15, no Suppl 1, p. P8. <https://doi.org/10.1186/1471-2202-15-S1-P8> [Consulta: 3 de junio de 2023]

12. BINMORE, Ken. Teoría de juegos. Madrid: McGraw-Hill, 1994.

13. LEE, Daeyeol. Game theory and neural basis of social decision making. Nature neuroscience, 2008, vol. 11, no 4, p. 404-409.

14. Imagen descargada de Pixabay. Imagen libre de derechos de autor bajo la licencia Creative Commons CC0. <https://pixabay.com/es/vectors/piedra-papel-tijera-juego-roshambo-156171/> [Consulta: 19 de agosto de 2023]

15. GAO, Bolin; PAVEL, Lacra. On the properties of the softmax function with application in game theory and reinforcement learning. arXiv preprint arXiv:1704.00805, 2017.

16. Página de Kaggle acerca de la competición Rock, Paper, Scissors <https://www.kaggle.com/competitions/rock-paper-scissors/> [Consulta: 11 de octubre de 2023]

17. España. Directiva del Consejo, de 29 de mayo de 1990, referente a las disposiciones mínimas de seguridad y de salud relativas al trabajo con equipos que incluyen pantallas de visualización (quinta Directiva específica con arreglo al apartado 1 del artículo 16 de la Directiva 89/391/CEE. DOCE, 21 de junio de 1990, núm. 156, p. 14-18. Departamento de Comunidades Europeas. <https://www.boe.es/doue/1990/156/L00014-00018.pdf> [Consulta: 11 de octubre de 2023]