UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Grado en Ingeniería de Computadores

Study of tabular reinforcement learning for mobile robot wall-following

Estudio del aprendizaje por refuerzo tabular para el seguimiento de paredes por un robot móvil.

Realizado por
Rafael Núñez López

Tutorizado por
Juan Antonio Fernández Madrigal
Ana M. Cruz Martín

Departamento
Ingeniería de Sistemas y Automática

UNIVERSIDAD DE MÁLAGA

MÁLAGA, FEBRERO DE 2021

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
Grado en Ingenieria de Computadores

Study of tabular reinforcement learning for mobile robot wall-following

Estudio del aprendizaje por refuerzo tabular para el seguimiento de paredes por un robot móvil.

Realizado por
Rafael Núñez López

Tutorizado por
Juan Antonio Fernández Madrigal
Ana M. Cruz Martín

Departamento
Ingeniería de Sistemas y Automática

UNIVERSIDAD DE MÁLAGA

MÁLAGA, FEBRERO DE 2021

Fecha defensa: Febrero de 2021

Resumen

El objetivo de este proyecto consiste en la implementación de un algoritmo de aprendizaje por refuerzo para la resolución de un problema de seguimiento de paredes en un entorno, en un robot LEGO con pocos recursos hardware (un sensor de distancia lateral y un giroscopio).

Se ha realizado un estudio teórico sobre el aprendizaje por refuerzo para dar un contexto al algoritmo usado, en concreto SARSA. Para ello, se han hecho varias versiones de este que cambian en las técnicas de exploración utilizadas y la variación de parámetros dentro del aprendizaje.

Finalmente se muestran los resultados obtenidos y un análisis y comparación de las técnicas implementadas. Este análisis de datos proporcionará información para entender cómo se comportan este tipo de implementaciones y la capacidad de aprendizaje en este tipo de problema.

Palabras clave:

Aprendizaje por refuerzo, SARSA, robótica.

Abstract

The goal of this project is the implementation of a learning reinforcement algorithm which is used for the resolution of the following wall problem in a environment, in a low hardware resources LEGO robot, with a side distance sensor and a gyroscope.

A theoretical study on reinforcement learning has been done to give a theoric context to the algorithm used, specifically SARSA algorithm. To do this, several versions have been tested changing the exploration techniques used and some learning parameters.

Finally, the obtained results and an analysis and comparison of the configurations are shown. This data analysis will provide information to understand how such implementations behave and the ability to learn about this kind of problems.

Keywords:

Reinforcement algorithm, SARSA, robotics

Índice

Índice de figuras

Índice de tablas

1

Introducción

1.1 Motivación

La robótica, de manera simplificada, consiste en el uso de máquinas (robots), para mejorar el rendimiento en multitud de tareas.

La Implantación de robots en la Industria se ha traducido en un aumento sobresaliente del nivel de productividad de las plantas. Por ello, se ha aumentado el rango de campos y tareas que un robot puede abarcar. Con el aumento del rango de tareas surge la necesidad de que los robots no solo sean capaces de hacer tareas físicas, si no que sean capaces de tomar decisiones por ellos mismos, como haría cualquier humano.

Cualquier sistema físico puede ser modelado en el dominio del tiempo, donde la respuesta del sistema dado será una función de entradas, valores e instantes temporales. Un sistema de control consiste en subsistemas y procesos (o plantas) ensamblados con el propósito de obtener una salida deseada a partir de una entrada referencia.

El control de sistemas es un ámbito fundamental de la sociedad moderna. Hoy en día hay numerosas aplicaciones conviviendo entre nosotros, desde un cohete que se lanza al espacio para poner un satélite en órbita hasta los frenos de un vehículo. El ser humano no es el único creador de sistemas controlados automáticamente, sino que también existen de forma natural [1]. Por ejemplo, en el cuerpo humano tenemos el páncreas que regula la azúcar en sangre. A la hora de controlar un sistema existen diferentes enfoques. Por ejemplo, los controladores clásicos, como los PID. Pero también existen otras aproximaciones, como el controlador adaptativo, en el que podríamos incluir el ¡presentado en este TFG.

En este caso, el trabajo está enfocado en el aprendizaje de una tarea de navegación autónoma móvil. Se trata de un área esencial en el campo de la robótica, ya que engloba una amplia gama de aplicaciones. En concreto se centra en un problema de seguimiento de paredes de un robot móvil. Consiste en un problema en

el que un robot tiene que aprender a moverse en un entorno, delimitado por una única pared infinita que tendrá que seguir sin chocar ni alejarse demasiado.

El aprendizaje por refuerzo RL (Reinforcement Learning), consiste en determinar qué acciones debe escoger un agente en un entorno dado con el fin de maximizar una recompensa. El problema de seguimiento de paredes tiene característica de recompensa a largo plazo (maniobras) por lo que es indicado como escenario donde analizar el desempeño de métodos de aprendizaje por refuerzo. El algoritmo implementado es SARSA que se detalla en el apartado 3.5.

Por último, se hace un estudio de este algoritmo con diferentes variantes y se comparan los resultados arrojados para realizar un análisis de la eficacia del aprendizaje para las diferentes versiones de este.

Los sistemas de control han traído numerosas ventajas a los sistemas, como ampliar la capacidad, control remoto o capacidad de tratar con perturbaciones [1].

Entre las aplicaciones reales en las que han sido implementados con éxito algoritmos de RL se encuentran algoritmos para planificación de recursos en clusters, sistemas multi agente de control de tráfico, etc[2].

1.2 Objetivos

En este TFG se ha usado un algoritmo de RL para resolver un problema básico de navegación como es el seguimiento de una pared usando el algoritmo de aprendizaje SARSA.

En principio, aunque el objetivo final era el mismo, uno de los puntos planteados era implementar un controlador clásico que permitiese resolver este problema, para comprobar que no se adaptaban de forma óptima al problema y sirviera de justificación al aprendizaje por refuerzo. Sin embargo, al tratarse de una investigación para un TFG esto escapaba a su dimensión ya que no se podía implementar como un sistema SISO y requería de más tiempo del inicialmente planteado.

Por tanto, finalmente se ha realizado un estudio sobre la influencia del parámetro factor de descuento del aprendizaje (capítulo 4) realizando varias configuraciones de la misma implementación. Es necesario escoger un buen factor de descuento ya que puede influir en la calidad de la solución dada. También se han adoptado dos estrategias diferentes de implementación de elección de acción que se presentan en el apartado 3.4.

Todo esto se ha implementado mediante un simulador de un robot LEGO, con pocos recursos que se verán en el apartado 2.4

Para ello se desarrollará una arquitectura de control para que el robot, con capacidades limitadas (solo contamos con un sensor lateral de proximidad y un sensor de posición) sea capaz de adaptarse a cualquier entorno de interior.

Por tanto, una vez ejecutado, el principal objetivo será analizar la capacidad que tiene el aprendizaje por refuerzo tabular para conseguir aprender a resolver esta tarea, y su desempeño, comparado las diferentes versiones implementadas.

1.3 Metodología

Se ha usado en este proyecto un proceso de tutorización en cascada donde los tutores han ido asignando progresivamente las tareas pudiéndose dividir en cuatro fases.

- Investigaciones previas. En primer lugar, se llevó a cabo una documentación sobre los algoritmos que se iban a implementar, además de adquirir una comprensión a nivel teórico del problema y las posibles soluciones
- Implementación de algoritmos. Se ha desarrollado el desarrollo del código fuente del proceso para todas las configuraciones del RL.
- Pruebas y análisis de resultados. Casi de forma paralela a la implementación del algoritmo, se han ido desarrollando las pruebas de este ya que se han ido modificando parámetros y estructuras en función del funcionamiento del algoritmo, para llegar a una versión eficiente del mismo.
- Este TFG se basa en el uso de una metodología basada en el análisis de datos. A partir de la implementación de controladores, y análisis de los resultados teóricos obtenidos, se entenderá mejor cómo se comportan ciertos controladores, y las capacidades del aprendizaje en este tipo de problemas. Como el aprendizaje por refuerzo se basa en probabilidad y estadística, es conveniente usar esta metodología.
- Documentación y memoria. Finalmente se ha recogido y documentado la información en una memoria, que a su vez ha requerido de una profundización a nivel teórico para su desarrollo, llevando de una visión global a una visión más concreta de cada algoritmo.

1.4 Estructura de la memoria

La memoria se divide en los siguientes capítulos.

- En el segundo capítulo se presentan los entornos en los que se va a trabajar de forma detallada.
- En el tercer capítulo se realiza una documentación previa sobre el aprendizaje por refuerzo que sirve de marco teórico para realizar el proyecto.
- En el cuarto capítulo se detalla la implementación realizada del algoritmo SARSA.
- En el quinto capítulo se realiza un estudio sobre el desarrollo del aprendizaje para entender como aprende el robot y como afectan los parámetros variados al mismo.
- En el sexto capítulo se lanza finalmente el robot para que a partir de los datos que ha adquirido del entorno a través del lanzamiento del aprendizaje, realice la navegación autónoma y se compraran los resultados obtenidos para saber que implementación ha funcionado mejor
- En el séptimo capítulo se habla sobre las conclusiones que se extraen una vez realizado el proyecto y se exponen ideas para proyectos futuros que continúen puedan continuar el actual.
- En el apéndice A se muestran las distancias euclídeas entre dos V(s) consecutivas para todas las configuraciones.
- En el apéndice B se muestran la recompensa acumulada en cada episodio para todas las configuraciones.
- En el apéndice C se muestran la recompensa acumulada con ruido en cada episodio para todas las configuraciones.
- En el apéndice D se recogen todas las tablas Q(S,A) una vez lanzado el aprendizaje de todas las configuraciones.

2

Herramientas utilizadas

2.1 Introducción

En este capítulo se expondrán los recursos software y hardware que se han utilizado durante la realización de este TFG, software y hardware.

Como se ha comentado en el apartado 1.2, se han utilizado una serie de recursos para poder implementar el algoritmo de aprendizaje por refuerzo. La situación pandémica ha dificultado los traslados a la escuela por lo que se ha decidido el uso de un simulador del robot Lego.

Este simulador implementa las funcionalidades hardware de las que dispone el robot real y permite hacerlo desde el propio entorno MATLAB. Esta ha sido uno de los principales motivos por los que se decidió usar este programa (además de su interfaz intuitiva y fácil de usar) y sus librerías de funciones que lo hacen muy adecuado para el análisis de datos y comparativa entre resultados, ya que están recogidas en el mismo entorno.

2.2 Robot Lego Mindstorms EV3

La base EV3 [3], ilustrada en la figura 1, es un ordenador empotrado que ejecuta una distribución particular de Linux. Tiene un procesador ARM9 de 300MHz de frecuencia, 64GB de RAM, 256KN de EEPROM y 16MB de Flash interna. Además, dispone de capacidad de ampliar memoria por medio de tarjeta SD o de implementar S.O alternativos.

La base se alimenta mediante una batería recargable de 2000mAh que proporciona los 9V de alimentación necesarios.

En condiciones normales, si se trabajase con el robot real se programaría en el entorno MATLAB & Simulink [4] y se volcaría la compilación de dichos modelos a un lenguaje tipo bytecode que es interpretado dentro del robot por una máquina virtual propia de Lego [5] .

Por último, para compartir los ficheros generados por el robot al equipo personal se usaría un software original de LEGO para el robot.

Figura 1 Base robot Lego Mindstorms EV3 [6]

Esta base dispone de 4 puertos de entrada etiquetados 1,2,3 y 4. En este TFG se han usado tres sensores. Un detector de colisiones, un sonar que apunta de manera lateral para medir la distancia con obstáculos y un giroscopio para la orientación. Los sensores se conectan a la base y la mayoría se detectan automáticamente. Solo en los sensores analógicos NXT se requiere una configuración manual del puerto [7].

Además también dispone de 4 puertos de salida etiquetados A,B,C y D. Estos puertos se usan principalmente con motores. En este TFG se han usado dos actuadores que son ruedas cada una con un motor independiente.

2.3 Matlab

En cuanto al software, los algoritmos han sido completamente desarrollados en el entorno MATLAB R2019b [4]. Se ha escogido este entorno, por ser un lenguaje de programación de alto nivel, que integra computación, visualización y programación en un ecosistema fácil de usar. En programación, se trata de un lenguaje de alto nivel basado en vectores y matrices.

Además, se usa el simulador [8] del robot Lego Mindstorms EV3 [6] facilitado por el tutor implementado también en MATLAB, lo que permite reunir todos los elementos en el mismo entorno.

2.4 Lego Mindstorms Minimalist Matlab Library

LM3LT es una clase minimalista escrita íntegramente en Matlab que simula, con cierto grado de realismo físico, un robot Lego Mindstorms con sus sensores y actuadores básicos. El simulador tiene el aspecto gráfico que se muestra en la figura 2. Consta de:

- Dos motores independientes (izquierda y derecha) con velocidades de rotación simuladas de primer orden.

- Sensor ultrasónico con ruido longitudinal de magnitud similar al real y también con posibilidad de añadir ruido angular y espesor de haz.
- Sensor de luz reflejado con ruido de medición y con área de medida modificable. Los archivos de imagen (por ejemplo, png) se pueden utilizar como mapa de luz en el suelo.
- Sensor gyro que imita el comportamiento (ángulo + velocidad) del sensor de giroscopio EV3.
- Detección de colisiones con obstáculos (paredes).

Los parámetros físicos de la configuración (posición de los sensores, radios de las ruedas, etc.) se pueden definir a voluntad [8].

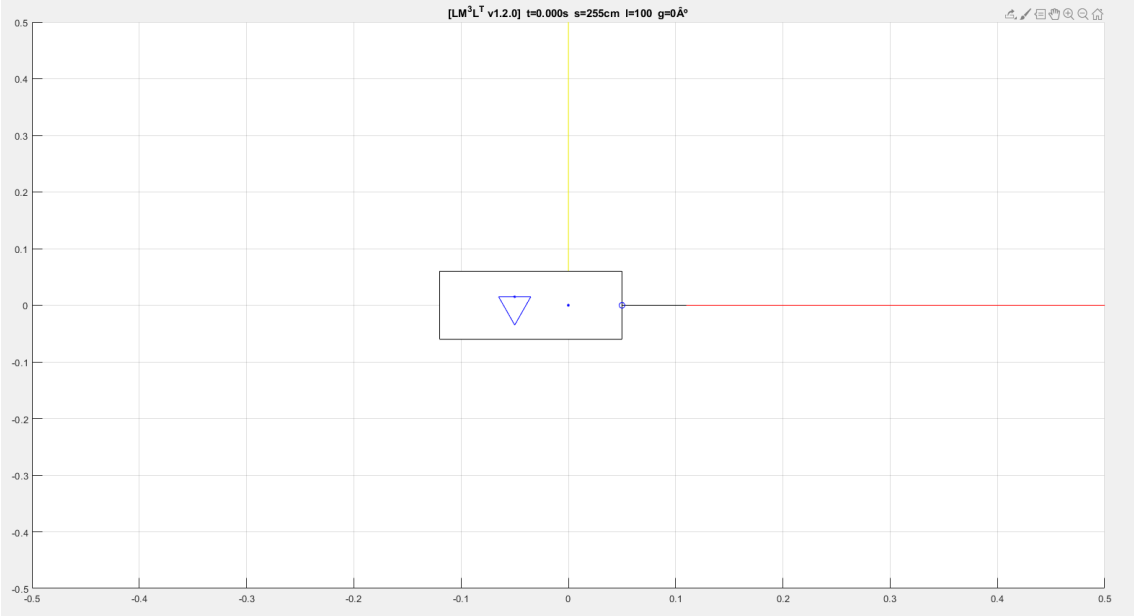

Figura 2. Aspecto del simulador extraído del entorno MATLAB

2.4.1 Funciones incorporadas

El simulador en MATLAB dispone de un programa principal que implementa las funcionalidades del robot usando 5 funciones extras de una librería privada.

En concreto para el desarrollo del algoritmo se han usado las siguientes funciones del programa Lego.m del simulador.

- "Lego()". Construye un robot Lego simulado.
- "changeDrawEnv()". Cambia el entorno en el plano.
- "defineWalls ()". Se usa para definir las paredes en el plano.
- "setUltrasonicDrawing()". Se usa para activar el sonar. El sonar dispone de una amplitud de 2.55m y está ubicado en uno de los laterales del robot apuntando perpendicularmente a la base.
- "simulate()". Se usa para simular durante un tiempo determinado.
- "collision()". Devuelve un parámetro cuyo valor determina si se detecta colisión.
- "delete()". Elimina el robot Lego simulado.

2.5 Hardware usado

En cuanto hardware, se ha usado un ordenador portátil personal, con los siguientes recursos:

- GPU: Nvidia 1050 4Gb dedicados.
- Memoria RAM: 8 Gb de RAM.
- Disco Duro: HDD Sata 1Tb.
- Windows 10.
- Intel Core i5 8300k 4 núcleos.

3

Aprendizaje por refuerzo

3.1 Introducción

El ser humano aprende interactuando con el entorno. Las personas responden a los estímulos que les llegan del ambiente con el que interactúan y aprenden en base a los resultados obtenidos.

El aprendizaje por refuerzo se define como un método para aprender cómo asignar acciones a un agente, de forma que se maximice la recompensa numérica obtenida. No se programa explícitamente qué acción escoger, si no que ejecutando el aprendizaje se aprende cuál es la acción que dará como resultado la mayor recompensa al escogerse en determinado momento. Existen diversos métodos para resolver este problema.

En este capítulo se introduce la teoría al aprendizaje por refuerzo. Se definen los conceptos básicos que son necesarios para la implementación del algoritmo de aprendizaje en el que se basa este TFG.

3.2 Elementos del aprendizaje por refuerzo

El aprendizaje por refuerzo es un problema en el que interviene un agente en un entorno. El agente interactúa con el entorno a través de acciones como se muestra en la figura 3. En cada paso el agente recibe información del estado en el que se encuentra y elige una acción que genera como salida. La acción cambia el entorno, y a través de una recompensa al ejecutar esta acción se calcula el valor de esta transición de estado y se envía al agente como se ve en la figura 3.

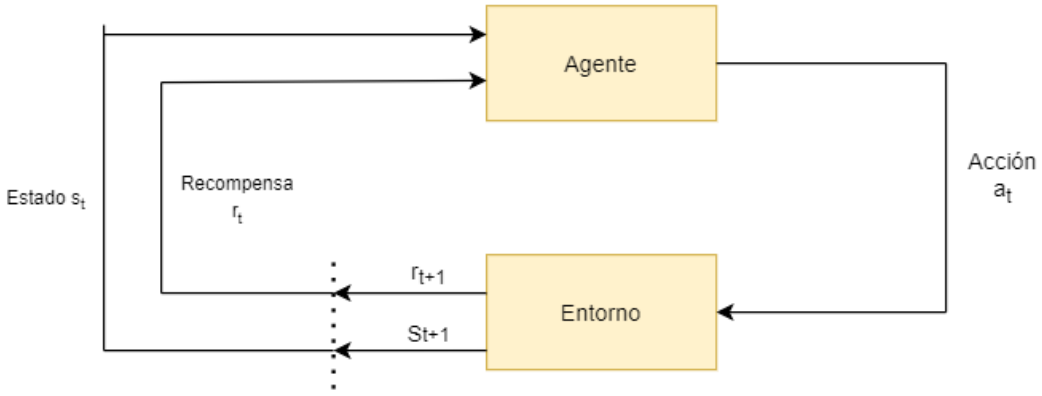

Figura 3. Esquema simplificado interacción del agente y el entorno.

A continuación, se definen los elementos generales que contienen la mayoría de los algoritmos de aprendizaje por refuerzo [9].

- Política π(s). La política determina el aprendizaje del agente. En términos generales, una política es un mapeo de los estados que definen el entorno y las acciones que se deben de tomar en esos estados.
- Recompensa r. Define los objetivos en el problema del aprendizaje por refuerzo. En cada paso (iteración) del aprendizaje se genera una recompensa a partir del entorno que define cómo de buena o mala es una acción para el agente en un estado determinado.
- Función valor v(s). De igual manera que la recompensa define cómo de beneficioso es un estado en el instante actual, la función valor especifica cómo de bueno es en el largo plazo. Entonces el valor de un estado define la cantidad de recompensa que el agente puede esperar acumular en el futuro, a partir de ese estado.
- Conjunto de acciones A. Conjunto de todas las acciones a definidas.
- Conjunto de estados S. Conjunto de todos los estados s que representan el entorno en el que trabaja el agente.

El agente debe escoger las acciones que tiendan a aumentar los valores de la recompensa acumulada a largo plazo, repitiendo sucesivamente el esquema de la figura 3.

3.3 Métodos tabulares de aprendizaje por refuerzo

Entre los métodos de aprendizaje por refuerzo, dos de los algoritmos más conocidos son Q-learning y SARSA. En este TFG se ha escogido SARSA como el algoritmo a implementar, ya que en general reduce el número de pasos para aprender una tarea, y también evita políticas cercanas a recompensas negativas, lo que podría dañar a un robot real [10]

Este tipo de problema se puede modelar como Procesos de Decisión de Markov (PDM), que es la base del aprendizaje por refuerzo.

3.3.1 Propiedad de Markov

Ahora se define formalmente la propiedad de Markov para el aprendizaje por refuerzo [11] Para mantener simpleza matemática, se supone que hay un número finito de estados y valores de recompensas. Esto permite trabajar en términos de sumas y probabilidades, en lugar de integrales y distribuciones de probabilidad.

Considerando un entorno en un tiempo t+1 que proviene de ejecutar una acción en el tiempo t, esta respuesta debería de depender de todo lo que ha pasado anteriormente. Esta dinámica puede ser definida especificando únicamente la distribución de probabilidad concreta.

$$Pr\{ r_{t+1} = r, s_{t+1} = s' | \ | s_0, a_0, r_1, \ldots, s_{t-1}, a_{t-1}, r_t , s_t , a_t\}, \ (1)$$

para todo r, s' y todos los valores posibles en pasos anteriores: s_0, a_0, r_1 …. s_t, a_t, r_t. Si el estado tiene la propiedad de Markov, la respuesta en el instante t+1 depende únicamente del estado s_t y la acción a_t escogidas en el paso anterior pudiendo definirse de la siguiente manera.

$$p(s', r|s, a) = \Pr\{r_{t+1} = r, s_{t+1} = s' \mid s_t , a_t\}, \ (2)$$

para todo, s', s_t y a_t. En otras palabras, la discretización del entorno cumple las propiedades de Markov si y sólo si (1) es igual a (2). En este caso se dice que el entorno tiene la propiedad de Markov.

Esto significa que el futuro es independiente del pasado, es decir, que el estado presente tiene toda la información de los estados pasados, por lo tanto, no sería relevante mantener información extra de ellos.

Si la solución cumple la propiedad de Markov su dinámica permite predecir el próximo estado y su recompensa dado un estado actual y una acción. Por tanto, a través de la iteración de esta expresión, se puede demostrar que se pueden predecir todos los estados futuros y las recompensas esperadas a partir del conocimiento que se tiene del estado actual. Otro aspecto al que se puede concluir y que es fundamental en el RL es que los estados de Markov proporcionan una base para escoger las acciones futuras de forma eficaz.

3.3.2 Proceso de decisión de Markov

Un proceso que satisface la propiedad de Markov explicada en el apartado 3.3.1 es conocido como proceso de decisión de Markov (MDP).

Si los estados y las acciones son finitos, entonces el proceso se conoce como un proceso de decisión de Markov finito. Estos son particularmente importantes para la implementación del aprendizaje por refuerzo.

Un MDP es definido por sus estados y acciones en cada paso en el entorno. Dado cualquier estado s y a, la probabilidad de pasar y recompensa, s',r, se define como.

$$\mathrm{p}(\mathrm{s}', \mathrm{r}|\mathrm{s}, \mathrm{a}) = \Pr\{S_{t+1} = \mathrm{s}', R_{t+1} = \mathrm{r} \mid S_t = \mathrm{s}, A_t = \mathrm{a}\} \ (3)$$

Dado el modelo presentado en la ecuación (3) se pueden conocer elementos del entorno tales como las recompensas esperadas para cada par de acción estado (4)

$$\mathrm{r(s,a)} \;=\; \mathrm{E}[, R_{t+1} \mid S_t = \mathrm{s}, A_t \;= \mathrm{a}] \quad (4)$$

Las funciones de valor estiman como de bueno resulta al agente estar en un estado S siguiendo una política determinada π. Existen dos tipos de funciones de valor: función V que estima como de bueno es estar en un estado y la función Q que estima como de bueno es realizar una acción en un estado [12].

El valor de s siguiendo una política π se define como la suma de recompensas que se espera obtener desde ese estado (5).

$$V^{\pi}(s) = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \text{ (5)}$$

La función valor estado-acción se puede definir como el valor asociado a tomar una acción a desde un estado s siguiendo una política (6).

$$Q^{\pi}(s,a) = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \text{ (6)}$$

Ambas usan un parámetro γ (descuento) que define el peso de las recompensas. Se pueden aplicar varias políticas, pero la finalidad del RL es adoptar una que maximice la recompensa. Esta se conoce como óptima. Las funciones valor, cumplen ciertas propiedades recursivas, por lo que pueden ser definidas en términos de ecuaciones de Bellman [13]. . De tal manera que desarrollando según [12] finalmente se llega a que la relación entre V* y Q* siendo estas los valores máximos de entre cualquier política de V y de Q y viene dada por (7).

$$V^*(s) = max_a Q^*(s,a) \text{ (7)}$$

Finalmente, la acción óptima se expresa en (8), es decir la acción que tenga mayor valor de entre las posibles de un estado s será la mejor.

$$\pi^*(s) = argmax_a Q^*(s,a) \text{ (8)}$$

Un MDP consiste en modelar el problema de forma que se discretiza para cada instante del tiempo el problema en estados. En otras palabras, dan una discretización del entorno donde se desarrolla el RL.

3.3.3 Elementos de un MDP

Para modelar el problema como un MDP se necesitan los siguientes elementos

- Conjunto de estados S: $\{s_1,s_2,..,s_n\}$ donde s_t denota el estado s del conjunto S en el tiempo t.
- Conjunto de acciones A: $\{a_1,a_2,..,a_n\}$ donde at es una acción escogida en el tiempo t contenida del conjunto A.
- Función de recompensa R_t, que es un número real resultado de aplicar una acción at en un estado s_t.

- Una función de probabilidad de transición de estado p(). Esta función es definida por p(s',r|s,a), que fue usada en el apartado 3.3.1 para definir la propiedad de Markov. Define la probabilidad de llegar a un estado s' a partir de la ejecución de una acción a en un estado s.

3.4 Exploración y explotación

Uno de los desafíos que surgen en el aprendizaje por refuerzo es el compromiso entre explotación y exploración a la hora de actualizar la política. Para maximizar la recompensa, el algoritmo debería preferir las acciones que ya ha intentado y son beneficiosas para producir recompensas (explotación). Sin embargo, puede haber acciones que no haya ejecutado lo suficiente, como para determinar si pueden producir una recompensa mayor a largo plazo, o que directamente no haya ejecutado, por lo que también tiene que explorar nuevos estados.

No se puede usar uno de los dos métodos de forma exclusiva para tener un buen aprendizaje. El agente debe de explorar las acciones suficientemente, para que progresivamente vaya escogiendo las que producen la mayor recompensa.

Dos métodos ampliamente conocidos para equilibrar exploración y explotación son ε -greedy y softmax.

3.4.1 Estrategia ε-greedy

Esta estrategia es una forma sencilla de escoger entre explotación y exploración. Se parte de un valor 'e' que establece una probabilidad fija de escoger una acción aleatoria en lugar de escoger la acción más beneficiosa con respecto a Q(S,A). En (9) se ve la implementación de este algoritmo. De esta manera el algoritmo se asegura de que se van a explorar todos los estados de QIS,A)para aprender cuál es la mejor acción para cada estado [14].

$$a(t) = \begin{cases} acción\ random\ sobre\ A & si\ e < \varepsilon \\ argmax_a Q^*(s,a) & en\ otro\ caso \end{cases} \quad (9)$$

Donde 0 ≤ e ≤1 es un número aleatorio uniforme extraído en cada paso del tiempo.

3.4.2 Estrategia SoftMax

Como se hizo referencia en el punto de exploración y explotación, la estrategia ε -greedy deja una probabilidad estática de coger cada acción. Sin embargo, hay procesos en los que escoger una acción aleatoria puede resultar insatisfactorio a largo plazo, por lo que esta técnica surge de la idea de usar probabilidad variable.

$$P(a|s) = P_r\{a_t = a|S_t = s\} = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_b e^{\frac{Q(s,a)}{\tau}}} \quad (10)$$

En contraste a la función 9 ε -greedy vista en el apartado 3.4.1, softmax toma como entrada el vector de acciones y lo normaliza en una distribución de probabilidad, que se determina clasificando las estimaciones de la función de valor utilizando una

distribución de Boltzmann (10). De esta manera un valor mayor repercutirá en una probabilidad mayor de escoger esa acción (12).

Donde τ es un parámetro positivo llamado temperatura. Una alta temperatura hace que las acciones sean casi equiprobables, mientras que sí la temperatura baja, se hace más propenso a la explotación [14].

La principal diferencia reside en la evolución del algoritmo, mientras ε-greedy es estático, es decir, la probabilidad es siempre la misma, la estrategia SoftMax va modificándola a lo largo del tiempo. Aparentemente para un problema en el que el escenario no varía es más interesante que la exploración vaya disminuyendo a medida que aumentan las visitas a los estados.

3.5 Algoritmo de aprendizaje SARSA

El objetivo es aprender una función valor. Se va estimando esta función $Q\pi$(s,a), para todos los estados S y acciones A. Esto se hace teniendo una función valor que guarda la mayor recompensa esperada para cada estado. Cada episodio consiste en una alternación de pares de estado-acción como se ve en la figura 4.

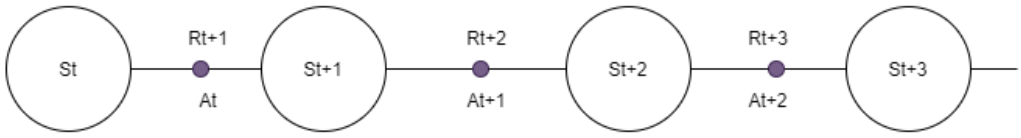

Figura 4. Flujo de pares acción-estado algoritmo SARSA.

Este algoritmo depende del estado actual (s), la acción escogida actual (a), la recompensa obtenida (r), el estado siguiente (s') y la acción siguiente (a') como se ve en la la ecuación 11. En la figura 5 se ve como es el flujo del algoritmo para la actualización de estos parámetros.

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma * Q(s',a') - Q(s,a)) \quad (11)$$

Se enumerarán y se explicarán los principales parámetros y los elementos usados durante el aprendizaje.

Parámetros:

- Ratio de aprendizaje (α). Conocido como learning rate. Este parámetro es un valor entre 0 y 1 que definirá la forma de aprendizaje del proceso. Si α=1 solo se tiene en cuenta lo aprendido en la iteración actual y se olvida lo anterior, si α=o se queda con lo que sabía y no se incorpora la información nueva. En este problema al tratarse de un entorno fijo se recomienda establecer un parámetro α dinámico que garantice que se converja en el tiempo.
- Factor de descuento (γ). Conocido como discount factor. El factor de descuento determina la importancia de las recompensas futuras. Mientras que un valor 0 hace que solo se tengan en cuenta las recompensas actuales, un valor cercano a 1 hace que el algoritmo se enfoque en las recompensas futuras.

Elementos:

- Matriz Q. Se trata de una matriz de dimensión SxA. En esta matriz se almacena el valor de cada acción para cada estado que se actualiza a través de SARSA con esta expresión. Cada expresión Q(s,a) hace referencia al par estado-acción de la matriz Q como se ve en la tabla 1.
- Acción a. Hace referencia a la acción escogida en el instante t.
- Acción a'. Hace referencia a la acción escogida en el instante t+1.
- Estado s. Estado actual en el que se encuentra el robot.
- Estado s'. Estado del robot en el instante t+1.

Esta actualización se realiza al final de cada estado s no terminal. Si s_t es terminal, finaliza la ejecución del programa. Si las propiedades de convergencia del algoritmo SARSA son adecuadas el algoritmo converge con probabilidad 1 a una política óptima [11].

A S	a1	a2	----	an
s1	$Q(s_1, a_1)$	$Q(s_1, a_2)$	---	$Q(s_1, a_n)$
s2	$Q(s_2, a_1)$	$Q(s_2, a_2)$	---	$Q(s_2, a_n)$
---	---	----	---	----
sm	$Q(s_m, a_1)$	$Q(s_m, a_2)$	----	$Q(s_m, a_n)$

Tabla 1*: Formato de tabla Q para cada par estado-acción*

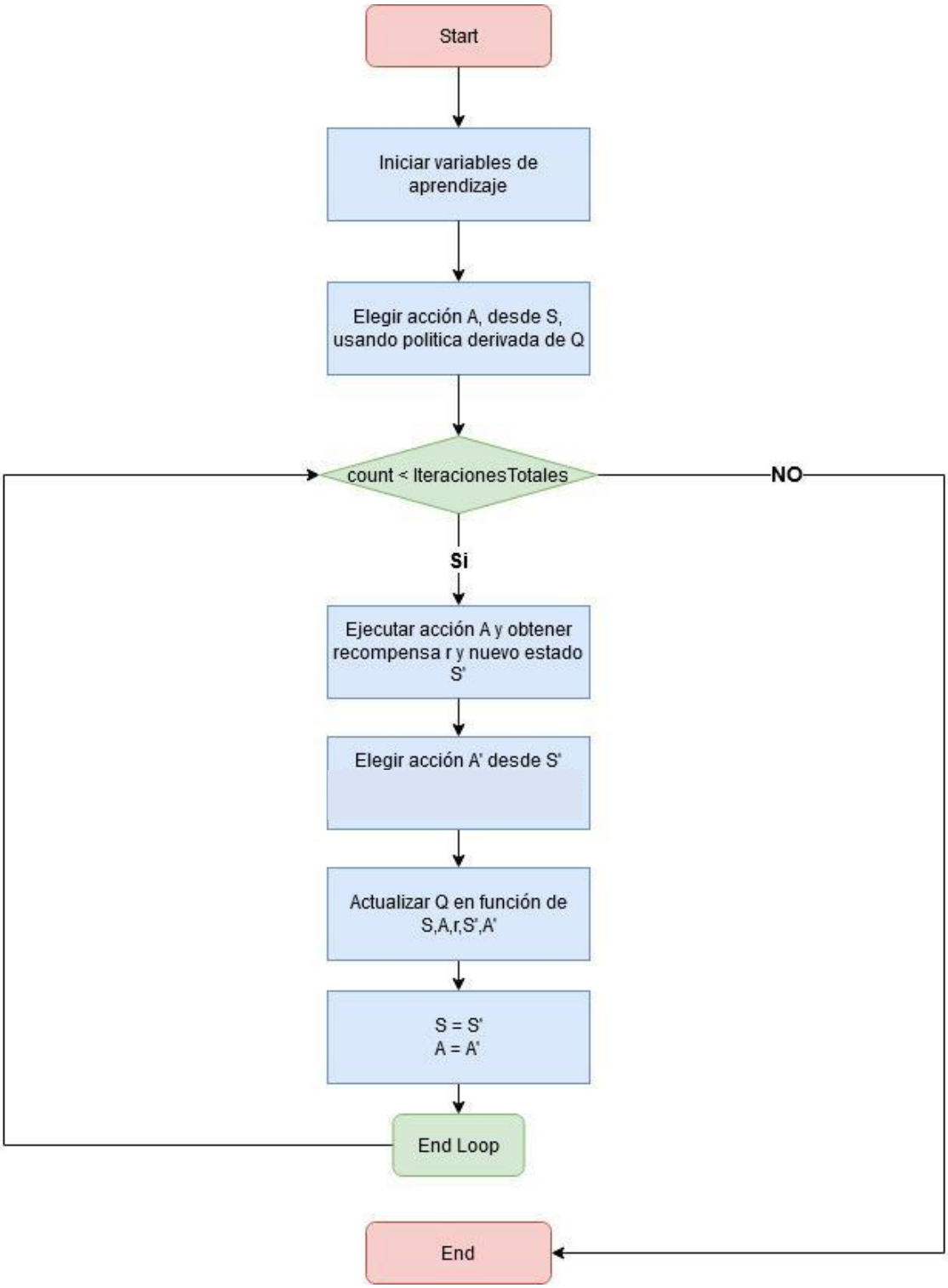

Figura 5. Abstracción del flujo de datos que sigue el algoritmo SARSA

4

Implementación del aprendizaje por refuerzo

4.1 Introducción

En este capítulo se explica de forma detallada el problema modelado y las estructuras usadas para el aprendizaje por refuerzo.

4.2 Problema a modelar

Para el diseño del problema se tiene en cuenta que se va a utilizar únicamente un sensor de distancia (sonar) y un giroscopio. El objetivo es que el agente sea capaz de desplazarse de forma autónoma, es decir, sin programar el movimiento del robot explícitamente siguiendo la pared infinita.

En primer lugar, se crea el entorno en el que se va a mover el robot, se trata de un plano XY que representa una habitación vacía con un muro a lo largo del eje 0X. En la figura 6 se ve la situación inicial del agente y el entorno. Como se mencionó anteriormente en el apartado 2.3.1, el sonar dispone de un rango de 2.55 m, por tanto, la implementación quedará construida de esta manera.

- Señal de entrada. Será u[k], un valor unidimensional que indica el giro que el robot tiene que hacer sobre sí mismo en cada momento, para alejarse o acercarse a la pared; u[k] se traduce a potencias de las dos ruedas de la siguiente manera:
    - pR[k]=P+u[k]
    - pL[k]=P-u[k]

Donde R es la rueda derecha, L la rueda izquierda y P una potencia constante que en este caso será 40. Las potencias de este cálculo se acotan al rango [-100,100], para que el robot sea físicamente capaz de realizar los giros.

- Agente. Es el robot LEGO del que se ha hablado en el capítulo 2
- Entorno. Consiste en una única pared que representa un muro de longitud infinita a lo largo del eje OX.
- Señal de salida. Es una señal que representa la distancia lateral entre robot y la pared respecto al tiempo. Esta es la señal que se quiere controlar. Como el sonar tiene un rango de 2.55m, se establece como señal de referencia 1.25m que es un valor cercano a la mitad.

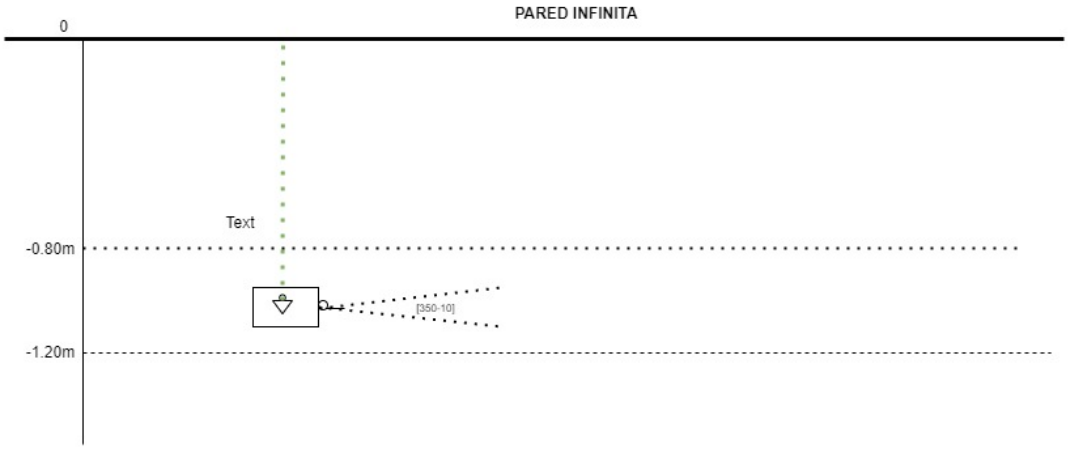

Figura 6. Representación del entorno y el agente en el estado inicial con el rango de distancias y de ángulos representados en líneas discontinuas.

En base a los recursos que ofrece el robot el problema ha quedado diseñado de la siguiente manera.

- Teóricamente la pared comprenderá los valores en el plano $x \in (-\infty, \infty)$ $y=0$ Sin embargo, en la práctica el programa tendrá un número finito de pasos, y la velocidad del robot no es lo suficientemente alta como para acercarse al límite establecido, por lo que la coordenada x estará comprendida en el intervalo (-5,10000).
- El robot iniciará la ejecución en una ángulo y distancia lateral favorable a la pared, es decir, estará en una orientación casi frontal y con el sonar apuntando directamente a la pared marcando una distancia media.
- Si se detecta una colisión con la pared, se reiniciará la posición del robot para empezar de nuevo en la posición favorable reiniciándose la ejecución del programa.
- Una vez terminada la ejecución, se mantendrán en memoria datos útiles para la evaluación del funcionamiento del algoritmo.

4.3 Discretización de estado

El problema debe adaptarse al algoritmo SARSA, para ello hay que discretizar el entorno de manera que cualquier situación sea representada por un estado.

Los estados propuestos se formulan teniendo en cuenta dos magnitudes, la distancia a la pared que mide el sonar en centímetros y la orientación que marca el giroscopio. Estos estados se convertirán a una magnitud numérica que definirá el número de estado en el que se encuentra el robot. La distancia a la pared se divide en los rangos mostrados en la figura 7 quedando un total de 7 intervalos.

Rango de distancias en orden numérico ascendente(cm): [0-40); [40-80]; [80-120); [120-160); [160-210); [210-255); [255]

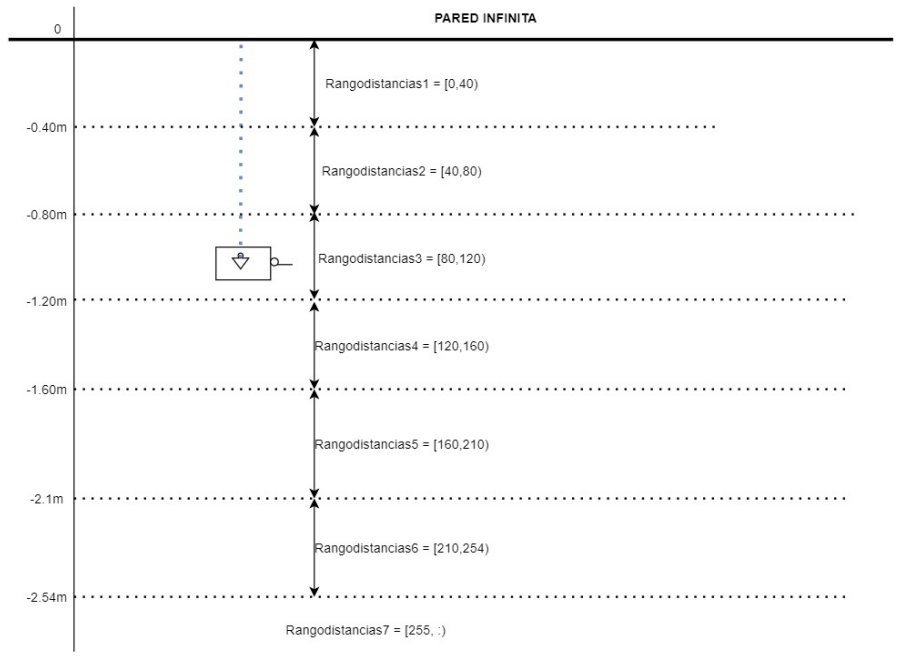

Figura 7. Representación de la discretización de estados por distancia lateral que indica el sonar. La distancia está representada en centímetros.

De igual manera que con la distancia, el rango de orientaciones en el que se encuentra el robot corresponde a un valor numérico que representa uno de los rangos de la figura 8.

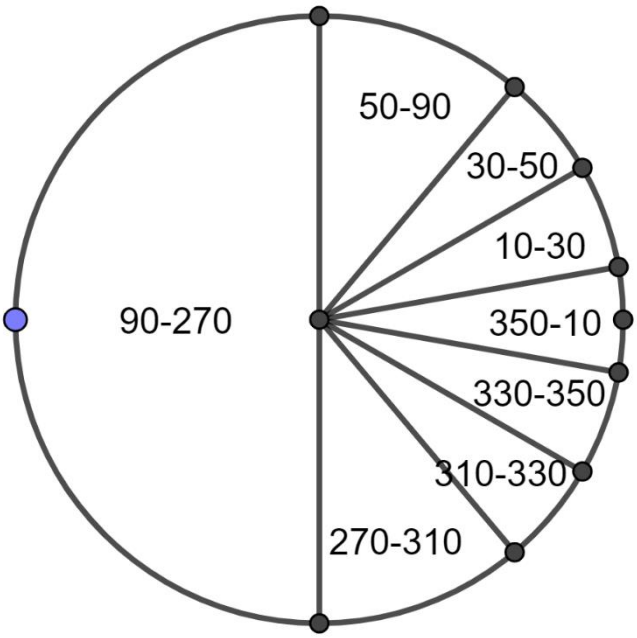

Figura 8. Representación del rango de amplitudes angulares usado.

Rango de ángulos en orden numérico ascendente (º): [350-10); [10-30); [30-50): [50-90); [90-270); [270-310); [310-330); [330-350)

Con estos dos valores se define una base numérica que define el estado en el que se encuentra el robot según los parámetros explicados. Este valor discreto será un número de dos cifras, en el que cada cifra hace alusión al intervalo numérico en el que se encuentra. La primera cifra indica su ubicación en los rangos de distancias y la segunda hace lo propio con el momento angular. En el ejemplo se ve un caso real para un robot con el sonar marcando 30cm y con 15º de orientación. Consultando la tabla 2:

$$BaseNumerica = \begin{cases} 1 \ Rangodistancias \\ 2 \ RangoAngular \end{cases} = 12 \ <=> 2 = BaseDiscreta \quad (12)$$

En la tabla 2 aparecen en amarillo los valores en la base numérica y en rojo su equivalente en valor discreto. En total se establecen 7 rangos de distancias y 8 rango de ángulos por lo que habrá un total de 56 estados.

A lo largo del desarrollo del TFG, se ha probado con distintas amplitudes y número de estados. Estos cambios han venido por requisitos del problema; por ejemplo, todos los estados no son de la misma amplitud: como se ve en la tabla 4, el quinto rango angular abarca de 90º a 270º ya que al ser el sonar lateral, el robot no va a visitar estos entre estos ángulos por lo que se acota en uno único. Por tanto, se ha realizado una reducción de estados de forma pragmática, quedando la configuración de estados tal y como queda representada en la tabla 2.

Se considera como estado favorable el 17 de la tabla 2. En el apartado 4.6 se explica el porqué de este estado.

La distancia lateral arroja información fundamental sobre el movimiento del robot. Como se ha mencionado anteriormente, el sonar dispone de un rango de 2.55 metros, por lo que se considera que, si se alcanza esta medida, el robot ha perdido la referencia de la pared. Este hecho en particular puede ser realmente perjudicial, ya que si se da en una etapa temprana del aprendizaje el robot puede que no sea capaz de volver a situaciones ya exploradas por no tener suficiente información en los estados, desembocando en que el robot se pierda y no sea capaz de volver a ubicarse, perdiéndose mucho tiempo de aprendizaje.

Valor discreto	Rango Distancias (cm)	Rango Ángulos (º)	Valor discreto	Rango Distancias (cm)	Rango Ángulos(º)
1	0-40	350-10	29	120-160	90-270
2	0-40	10-30	30	120-160	270-310
3	0-40	30-50	31	120-160	310-330
4	0-40	50-90	32	120-160	330-350
5	0-40	90-270	33	160-210	350-10
6	0-40	270-310	34	160-210	10-30
7	0-40	310-330	35	160-210	30-50
8	0-40	330-350	36	160-210	50-90
9	40-80	350-10	37	160-210	90-270
10	40-80	10-30	38	160-210	270 310
11	40-80	30-50	39	160-210	310-330
12	40-80	50-90	40	160-210	330-350
13	40-80	90-270	41	210-254	350-10
14	40-80	270-310	42	210-254	10-30
15	40-80	310-330	43	210-254	30-50
16	40-80	330-350	44	210-254	50-90
17	80-120	350-10	45	210-254	90-270
18	80-120	10-30	46	210-254	270-310
19	80-120	30-50	47	210-254	310-330
20	80-120	50-90	48	210-254	330-350
21	80-120	90-270	49	254-256	350-10
22	80-120	270-310	50	254-256	10-30
23	80-120	310-330	51	254-256	30-50
24	80-120	330-350	52	254-256	50-90
25	120-160	350-10	53	254-256	90-270
26	120-160	10-30	54	254-256	270-310
27	120-160	30-50	55	254-256	310-330
28	120-160	50-90	56	254-256	330-350

Tabla 2. *Configuración final de estados con estado favorable en verde.*

Es por eso por lo que surge la necesidad de dividir la ejecución del algoritmo de aprendizaje en capítulos o episodios, de forma que, si se da una situación como la descrita anteriormente, se vuelva a lanzar el aprendizaje para aprovechar al máximo lo aprendido en cada iteración.

4.4 Asignación de acciones

Se han establecido tres opciones para que el robot pueda desplazarse: recto, rotar a la izquierda o rotar a la derecha. Son representadas por un vector cuyas posiciones indican que acción se realiza y su valor influye en el movimiento de los ejes, como se vio en el apartado 4.2.

La amplitud giro se ha determinado de forma experimental para que el cambio de orientación sea lo suficiente como para transitar a otro estado diferente.

En cada paso se ejecuta una acción que viene determinada por una de las estrategias de selección de acción que se explican en el apartado 4.7. Esta acción se simula durante 0.7s para que tenga suficiente tiempo como para influir en la posición del agente.

4.5 Aprendizaje episódico

El aprendizaje se realiza de forma episódica, es decir, que, durante la ejecución del programa, el proceso se segmenta en episodios. Si el robot aprende, normalmente la duración de los episodios se debe de ampliar y reflejar ese aprendizaje.

El fin de episodio viene dado por los siguientes sucesos:

- Se detecta un choque con la pared, la librería del simulador incluye una función que nos devuelve un parámetro cuyo valor determina si se detecta colisión.
- El sonar llega al máximo de distancia medible, es decir, deja de detectar la pared. Este caso se dará cuando el sonar devuelva una distancia igual a 2.55m. La distancia del sonar se calcula en cada iteración por medio de una función ya implementada en el simulador.
- Se llega a un total de 500 pasos seguidos. Una vez que se han dado esta cantidad de pasos se considera que el episodio no va a aportar más información y se inicia un nuevo episodio. Se ha llegado a este valor porque en algunas ejecuciones el agente aprendía la tarea rápidamente y no se daba ninguna de las circunstancias para el fin de episodio.

Tras varios experimentos, se configuró el robot para que el primer paso del aprendizaje lo diese en una situación favorable, por tanto, el estado inicial será siempre el coloreado en verde de la tabla 2.

4.6 Reparto de la recompensa

A largo plazo en el aprendizaje por refuerzo el objetivo principal es que la recompensa acumulada sea lo mayor posible, por lo que la manera de definir y asignar las recompensas es fundamental para el correcto funcionamiento del proceso.

Para este problema concreto, se ha decidido repartir las recompensas de manera que, si se produce un fin de episodio, ya sea por choque o porque el robot pierde la referencia de la pared, se le penaliza con un valor de -100.

Por otro lado, si el robot se mantiene en un estado que se ha considerado como el óptimo, se aplica una recompensa positiva de valor 10. El objetivo es que el robot se mantenga en la franja que se aplica esta recompensa el mayor tiempo posible para que la recompensa sea máxima.

En las primeras versiones del algoritmo implantado en el TFG, no se aplicaba esta recompensa en la franja considerada como favorable, esto finalmente repercutía en una mayor oscilación del sistema debido a que no tener esta recompensa hacía que el robot considerase como óptimo un rango mayor de distancias y de inclinaciones por lo que finalmente se estableció esta recompensa para evitar este efecto.

4.7 Escoger una acción

Para este caso se ha optado por tres opciones distintas. La primera y la segunda funcionan siguiendo la estrategia ε-greedy explicada en el apartado 3.4.1, usando como ε un valor de 0.30 para la primera configuración y un valor de 0.15 para la segunda.

$$ai = \begin{cases} acción\ random\ escogida\ A(s) & si\ e < \varepsilon \\ argmaxQ(s,a) & en\ otro\ caso \end{cases} \quad (13)$$

Por último, se decidió usar una política que vaya incrementando la exploración como es softmax vista en el apartado 3.4.2. Se ha hecho un ajuste para evitar problemas de overflow que surgían en el entorno utilizado (Matlab) y fueron provocados por el gran número de pasos que requiere el aprendizaje. Se partía de la ecuación 14 pero convergía a 0 y se optó por la aproximación a la ecuación 15.

$$P_t(a_k) = \frac{e^{\frac{Q(St,Ak)}{To*\lambda^{t-1}}}}{\sum_{i=0}^{n} e^{Q(St,A)}} \quad (14)$$

$$P_t(a_k) = \frac{e^{\frac{Q(St,Ak)-Qmax}{To*\lambda^{t-1}}}}{\sum_{i=1}^{n} \frac{e^{Q(St,Ai)-Qmax}}{To*\lambda^{t-1}}} \quad (15)$$

4.8 Elementos de aprendizaje

En el apartado 3.5 se comentaron los parámetros y elementos del algoritmo SARSA. En este, se explican los valores utilizados en la implantación de dicho algoritmo que se ha realizado en el TFG. Algunos de ellos son estáticos y se definen en la cabecera del programa, por lo que no variarán durante el transcurso del aprendizaje y otros serán dinámicos.

- Learning rate (α). Se establece inicialmente a 1. Habrá dos configuraciones distintas, una que converge más rápidamente que se calcula en base al número

de pasos totales que se han realizado (ecuación 17). Como alternativa la otra configuración se calcula en base al número de visitas del estado actual (ecuación 16).

$$\alpha_t \begin{cases} \alpha = 1 & \text{si visitas}(s,a) = 0 \\ \alpha = \frac{1}{visitas(s,a)} & si\ visitas\ (s,a) > 0 \end{cases} \quad (16)$$

$$\alpha = \frac{1}{NoIteracion} \quad (17)$$

- Discount factor (γ). Se establece en un valor cercano a 1, en concreto 0.99.
- Matriz Q. La matriz Q (56x3) se inicializa vacía y se ira actualizando en cada paso por medio de la expresión definida por el algoritmo SARSA en la ecuación 11.
- Acciones A. Este vector de tamaño 3 define las tres acciones posibles para cada estado S. El número que representa a la acción viene dado por su posición en el vector. Cada acción es un indicador de giro que se envía a las ruedas de igual manera que se detalla en el apartado 2.2 de manera que:
  - Acción 1. Tiene un valor de 0. El robot se desplaza hacia adelante sin ningún tipo de cambio de orientación.
  - Acción 2. Tiene un valor de -30. El robot se desplaza hacia adelante con giro en sentido horario.
  - Acción 3. Tiene un valor de 30. El robot se desplaza hacia adelante con giro en sentido antihorario.
- Vector valor V. Es un vector de 56 posiciones, que se inicializa vacío, se va actualizando en cada paso y representa el máximo valor acumulado para cada estado.

4.9 Implementaciones para análisis de datos

Hasta ahora se han comentado detalles de la implementación del propio algoritmo. Sin embargo, se han usado estructuras para el almacenaje de información en las ejecuciones. También se han desarrollado funciones para tratar esta información y mostrarla de forma útil en el estudio que se va a realizar a continuación, en el capítulo 5.

Las funciones más relevantes han sido las usadas para obtener las gráficas e información recogidas en los apéndices de este TFG.

5

Estudio comparativo entre versiones del aprendizaje

5.1 Introducción

Una vez visto lo relativo al aprendizaje con refuerzo usando el algoritmo SARSA y a su implementación, en este apartado se realizará un estudio de diferentes configuraciones obtenidas variando alguno de los parámetros del algoritmo, para conocer cómo influyen en el aprendizaje y ver qué tal se desenvuelve el agente a la hora de aprender.

En total han resultado un total de 6 configuraciones distintas. Todos los experimentos se han ejecutado con las mismas condiciones iniciales que se explican en el apartado 5.1.1.

Se realiza un estudio previo sobre cada variante de manera individual para observar la evolución del propio aprendizaje y conocer como es este proceso en cada caso.

5.1.1 Establecimiento de parámetros para el aprendizaje

Se enumerarán los parámetros comunes que definen la simulación para cada uno de los experimentos:

- Distancia a la pared y orientación. Al comienzo, se establece una distancia lateral de entre 0.80m-1.20m y una orientación entre 350-10º. Este rango es abarcado por un único estado, que es el 17 (favorable) de la tabla 2. Se comienza en este estado para que el robot no requiera de muchos pasos para

ubicarse en esta posición. Así se consigue una mayor exploración en los primeros pasos.

- Pasos totales. Se ha determinado el número de pasos (iteraciones) totales de forma experimental de manera que el agente tenga el suficiente tiempo como para poder realizar el proceso del aprendizaje. Cuando se alcanzan los últimos pasos, el robot ha conseguido alargar sus episodios y mantenerse en la zona recompensada, por tanto, se considera que no va a arrojar información nueva significante por lo que se decide guardar los datos y reiniciar el experimento. Con el total de 7000 iteraciones y cada paso de 0.7s se está lanzando un experimento cuya duración es de 4900s lo que equivale a 1h 21min y 6s.
- Pasos máximos para cada episodio. Se terminó por adoptar un total de 400 pasos como límite de episodio, ya que se considera que una vez llegado a esta duración el robot no va a aprender información nueva y se prefiere reiniciar el episodio.
- Número de episodios. No se ha establecido un límite de episodios. El límite en la ejecución completa del programa viene dado por el número de pasos totales por lo que habrá tantos episodios como de tiempo.

5.1.2 Elección parámetros de aprendizaje

Como se ha comentado en el apartado 5.1, han resultado un total de 6 configuraciones distintas. Se ha hecho de esta manera para comparar las estrategias de escoger acción (softmax y ε-greedy), además de variar el parámetro learning rate, explicado en el apartado 3.4 para inferir en el nivel de explotación. Se ha asignado una etiqueta a cada configuración como se ve en la tabla 3.

Etiqueta	Configuración
ε-greedy0.15DivTotal	ε-greedy con ε=0.15 y con actualización de learning rate mediante el número de iteración actual.
ε-greedy0.15DivVisitas	ε-greedy con ε=0.15 y con actualización de learning rate mediante el número de visitas del estado en la iteración actual.
ε-greedy.30DivTotal	ε-greedy con ε=0.30 y con actualización de learning rate mediante el número de iteración actual.
ε-greedy0.30DivVisitas	ε-greedy con ε=030 y con actualización de learning rate mediante el número de visitas del estado en la iteración actual.
SoftmaxDivTotal	Softmax con actualización de learning rate mediante el número de visitas del estado en la iteración actual.
SoftmaxDivVisitas	Softmax con actualización de learning rate mediante el número de visitas del estado en la iteración actual.

Tabla 3.*: Etiquetas que representan a cada una de las configuraciones desarrolladas.*

5.2 Análisis del aprendizaje.

Es importante comprobar en primer lugar que el agente está aprendiendo correctamente y cómo lo hace, para tener una idea de qué resultado van a arrojar las configuraciones. En este apartado se analiza de manera empírica el desarrollo del aprendizaje. Se realizará una ejecución con cada versión de 7000 pasos y se analizarán parámetros que reflejan qué tal se desenvuelve el aprendizaje. Para ello se compara lo siguiente: distancia euclídea entre V(S) consecutivas, ajuste de las gráficas V(S) a funciones exponenciales y recompensa acumulada en cada episodio.

5.2.1 Comparación de distancias euclídeas V(S)

Una vez que se ha lanzado el aprendizaje, se muestra para cada paso la distancia euclídea entre $V(S)_t$ y $V(S)_{t-1}$ de tal manera que reflejará en valor absoluto el grado de aprendizaje que ha tenido el agente en cada paso. Lógicamente, si nuestra implementación de SARSA es correcta, el agente debería ir aprendiendo y reflejando menor aumento de valor de forma descendente para acabar convergiendo a la política determinada.

En el apéndice A están recogidas las gráficas de todos los experimentos. Se representa la distancia euclídea comentada anteriormente y además todos los finales de episodio según el motivo por el que se provocó. En general se aprecia que el valor se va reduciendo conforme aumentan las iteraciones, pero a simple vista se ve que no lo hacen de la misma manera (figura 9 contra figura 10).

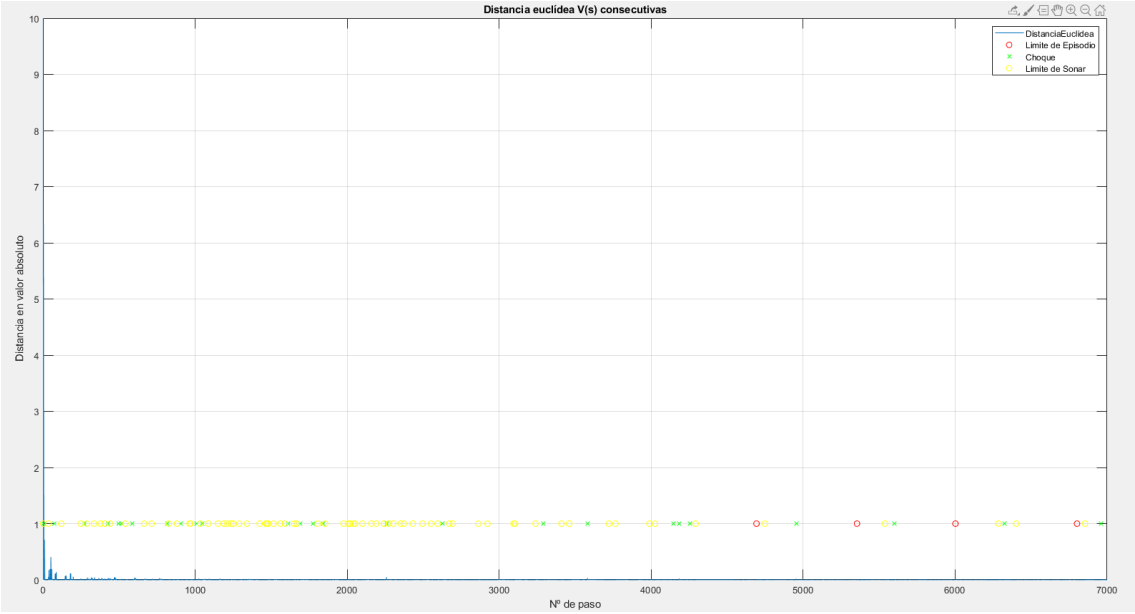

Figura 9. Distancia euclídea entre V(s) consecutivas para cada paso con estrategia softMax y actualización de learning rate por número de iteración.

La diferencia más notoria se produce en el método usado de actualización del learning rate. Por ejemplo, en softmax usando el número de iteración para actualizarlo se ve como la diferencia de valor es máxima al principio y se va reduciendo de manera exagerada con el paso de las iteraciones (figura 9). Sin embargo, en la figura 10 se aprecia que hay numerosos repuntes de valores. Esto se debe a que al actualizar el

learning rate en función del número de visitas, cada vez que se visita un estado por primera vez el nivel de aprendizaje es máximo.

Softmax con actualización del learning rate mediante el número de visitas se intuye más beneficiosa para el algoritmo ya que cuando se visita un estado, los primeros pasos en ese estado son de un alto grado de exploración lo que hace que el agente aprenda más rápido cómo de beneficiosa o perjudicial puede ser la acción para el mismo. Al ser un cambio en un parámetro del aprendizaje esta tendencia se repite en las implementaciones de ε-greedy cómo se ve en las figuras del apéndice A.

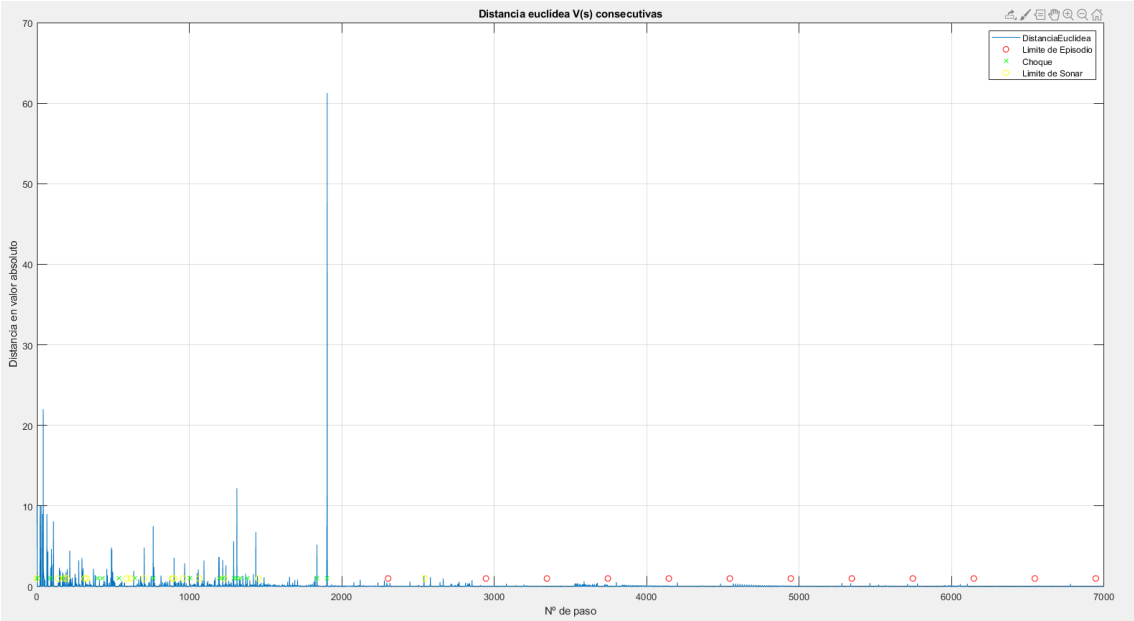

Figura 10. Distancia euclídea entre V(s) consecutivas para cada paso con estrategia SoftMax y actualización de learning rate por el número de visitas de cada estado.

En la figura 11 se muestra una comparativa entre ε-greedy con ε igual a 0.3 en naranja con una ejecución de softmax (azul), ambas actualizando el learning rate con el contador de visitas. En este caso se ve como con ε-greedy hay un mayor número de repuntes cuando está en la zona de pasos altos (3000-7000).

Con esta información se aprecia como softmax tiene una tendencia agresiva a explorar en los primeros pasos (1-100), mientras la exploración en ε-greedy se da de forma más dilatada en pasos. Esto hace indicar que la variante con softmax aprende de manera más rápida, lo que en un problema como este en el que el entorno permanece invariable en el tiempo parece la mejor opción.

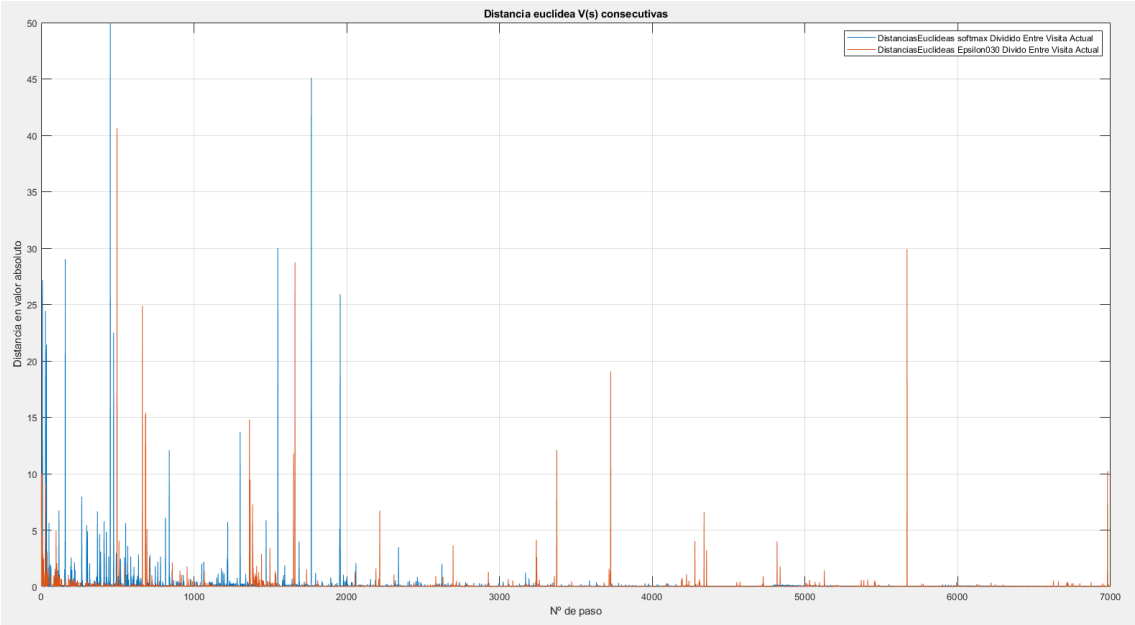

Figura 11. Grafica superpuesta de softmax y ε-greedy con ε igual a 0.3 para la misma configuración.

5.2.2 Comparación del ajuste exponencial de distancias euclídeas V(s)

Uno de los parámetros usados para reflejar la velocidad con la que se alcanza la política determinada de forma más certera, consiste en realizar un ajuste exponencial en forma y = A * e^{B*x}. Estos ajustes han quedado finalmente representados por las curvas exponenciales que aparecen en la tabla 4.

Como en el apartado anterior, el learning rate puede afectar al aprendizaje de manera que si decrece muy rápidamente se acelera la convergencia, pero si lo hace muy rápido el aprendizaje es corto y puede derivar en una política subóptima [12]

Configuración	Ajuste ($y = A * e^{B*x}$)
ε-greedy0.15DivTotal	$13.52 * e^{((-0.05921)*x)}$
ε-greedy0.15DivVisitas	$3.24 * e^{((-0.001355)*x)}$
ε-greedy.30DivTotal	$13.52 * e^{((-0.1885)*x)}$
ε-greedy0.30DivVisitas	$1.732 * e^{((-0.0006465)*x)}$
SoftmaxDivTotal	$0.3424 * e^{((-0.02441)*x)}$
SoftmaxDivVisitas	$16.25 * e^{((-0.005522)*x)}$

Tabla 4. *Resultado del ajuste exponencial implementado a cada una de las configuraciones.*

Para realizar el ajuste exponencial se ha aproximado la función de valores consecutivos V(s) del apartado 5.2.1, a una función escalonada tomando sus máximos locales como amplitud de estos. Este ajuste se realizado con la herramienta Curve Fitting [15] de MATLAB que implementa el ajuste exponencial de la función escalonada sobre el eje X, dando como resultado las curvas que se ven en la figura 12.

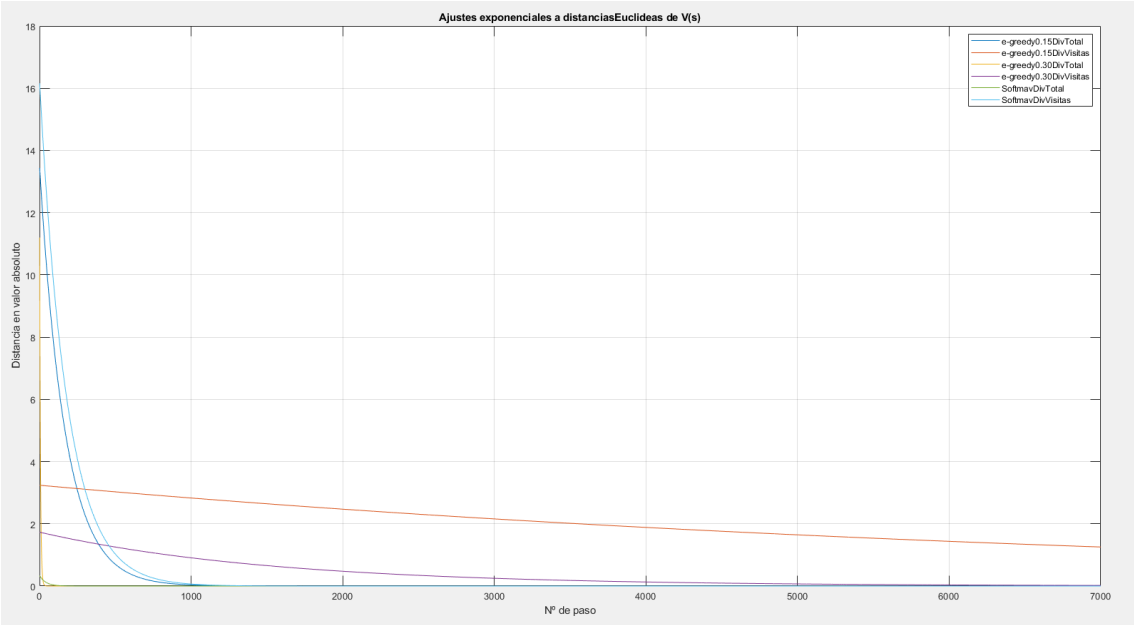

Figura 12. Representación de todos los ajustes exponenciales a las gráficas de distancias euclídeas de V(s).

En este caso haciendo énfasis en las curvas de bajada de la figura 12, en general se aprecia principalmente que la estrategia de actualizar learning rate, por medio del número de iteración actual hace que se converja antes a la política final (línea azul oscura, verde y amarilla), por tanto, la velocidad del aprendizaje es mayor. A igualdad configuración, softmax es más rápido aprendiendo que ε-greedy (función azul clara frente a naranja y morada) como se ve en la figura 13.

En conclusión, se reafirman las ideas expuestas en el capítulo 5.2.1. Entre las dos configuraciones, se obtiene mayor velocidad de aprendizaje con la actualización de learning rate por medio del número de iteración, que hace que disminuya mucho más rápido.

Además, a igualdad de configuración, softmax realiza un aprendizaje más rápido que ε-greedy, porque esta estrategia de selección de acción hace que se dé la exploración mayor en los primeros pasos.

Sin embargo, como se comentó en el apartado 1.2, una velocidad de convergencia demasiado grande puede derivar en una política subóptima por lo que habría que buscar un equilibrio entre velocidad de aprendizaje y los resultados arrojados.

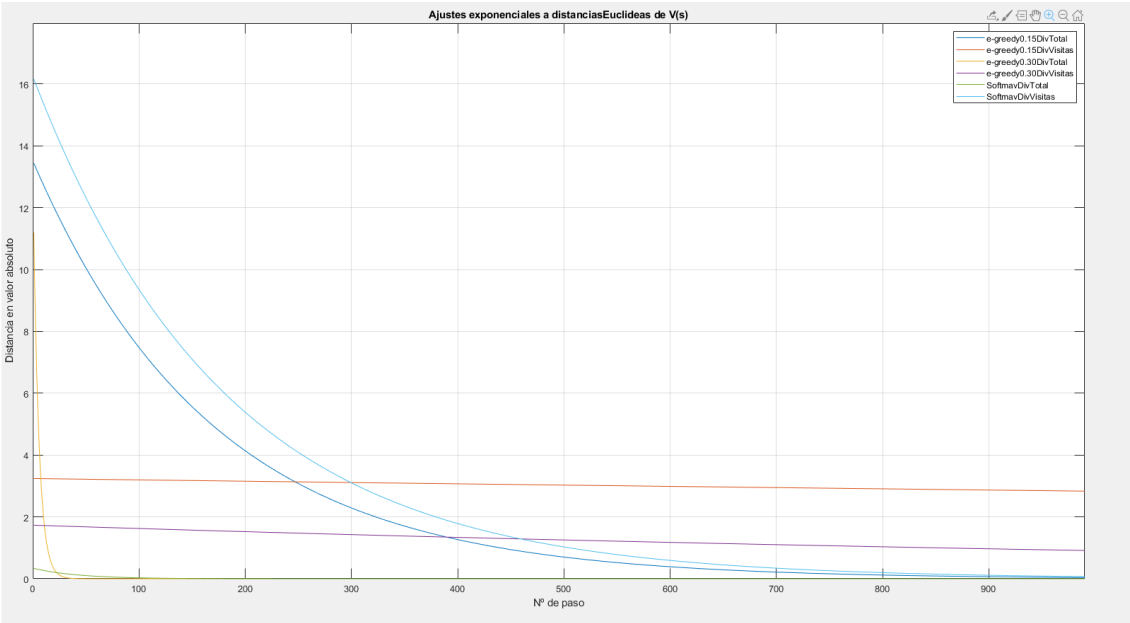

Figura 13. Representación de todos los ajustes exponenciales a las gráficas de distancias euclídeas de V(s) ampliada en las bajadas de la curva.

5.2.3 Vector recompensa acumulada

Para cuantificar como de bueno es el aprendizaje un parámetro que se puede atender es la recompensa media acumulada en cada episodio. De esta forma, se debe de apreciar que los episodios cada vez tienen mayor recompensa acumulada.

Una vez visto que el robot aparentemente aprende, se lanzaran 50 ejecuciones de cada configuración y se calculará un vector recompensa acumulada almacenada en cada episodio. Así han resultado las gráficas que se encuentran recogidas en el apéndice B. En la memoria se comentarán los resultados más significativos que devuelve.

En primer lugar, todas las configuraciones comienzan a acumular recompensa relativamente pronto excepto softmax con el learning rate actualizado mediante la iteración actual que tarda unos 70 episodios en comenzar a dar resultados. Además, la recompensa máxima acumulada es de 350, mientras que la media, una vez que ha empezado a acumular tendencias positivas, oscila entre los 200 pero se presenta muy irregular, como se ve en la figura 14.

Por otro lado, en las implementaciones que usan ε-greedy la recompensa empieza a acumularse requiriéndose de un menor número de episodios independientemente del ε usado. Se repite la tendencia de que las implementaciones que usan el número de visitas del estado acumulan mayor recompensa como es el caso de la figura B4 y B6 frente a B3 y B5 del apéndice B.

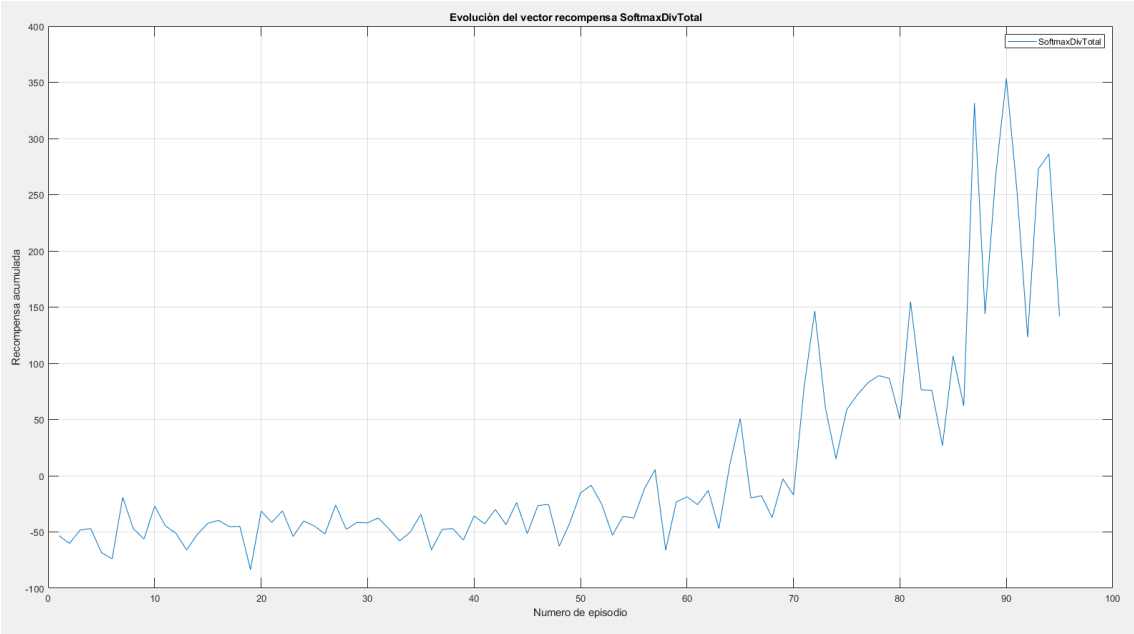

Figura 14. Recompensa acumulada por episodio para SARSA con Softmax y learning rate actualizado por el número de iteración actual.

Entre B4 y B6 la configuración con la mayor recompensa acumulada se presenta en la figura 15 que finalmente ha sido la implementación con ε-greedy 0.30 y learning rate actualizado por el número de visitas del estado actual.

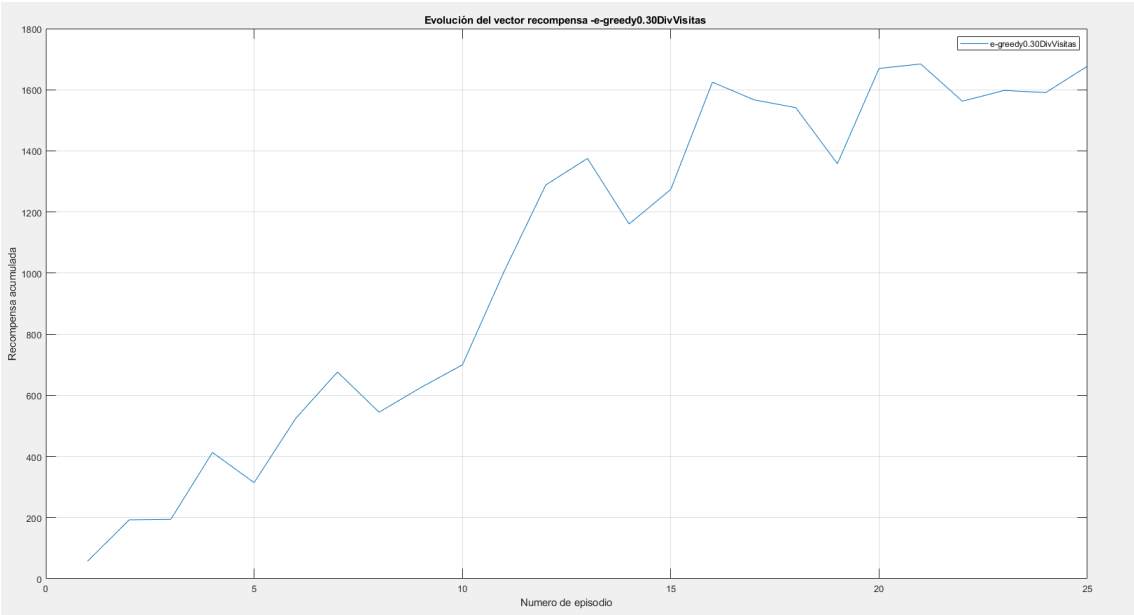

Figura 15. Recompensa acumulada por episodio para SARSA con ε-greedy 0.30 y learning rate actualizado por el número de visitas del estado actual.

Frente a softmax (figura 16), la recompensa acumulada es superior y se empieza a acumular con un menor número de episodios. Sin embargo, esto no quiere decir que aprenda más rápido ya que softmax explota mucho al principio lo que hace que los primeros episodios tiendan a ser muy cortos.

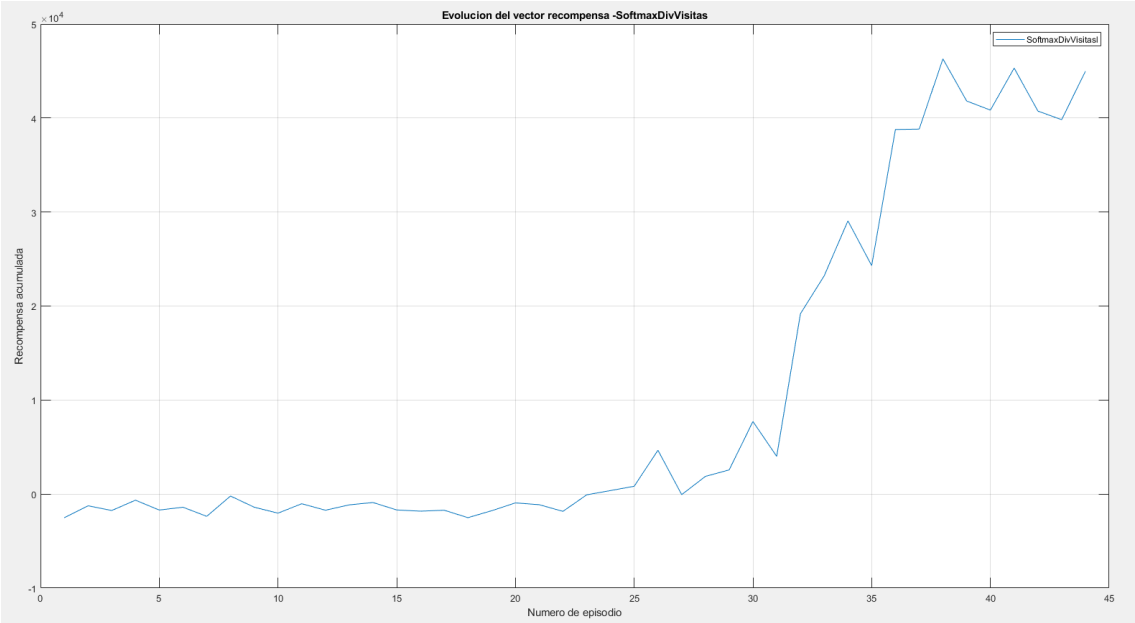

Figura 16. Recompensa acumulada por episodio para SARSA con softmax y learning rate actualizado por el número de visitas del estado actual.

5.2.4 Conclusiones

Por tanto, para softmax ha funcionado indiscutiblemente mejor la configuración que usa el número de visitas para actualizar learning rate. En ε-greedy se repite esta tendencia en cuanto a valor total acumulado, aunque con el número de iteración actual también tiene un buen comportamiento en este aspecto.

Esto se explica porque a diferencia de softmax que tiene una explotación muy agresiva al principio, ε-greedy se mantiene con una probabilidad estática que hace que esta explotación llegue a más largo plazo. Por tanto, al usar un learning rate que disminuye muy rápido el robot aprende rápidamente a desarrollar la tarea propuesta, lo que explica la rápida convergencia en la gráfica de las V(s) consecutivas de la figura 12.Parece que el robot va a aprender rápidamente a hacer la tarea, pero no va a acumular tanta información del entorno.

En el apéndice D se ven las mismas gráficas que el apéndice C, pero aparecen con ruido representado. Este ruido se calcula a partir de una estimación hecha con la desviación estándar de cada uno de los episodios para los 50 experimentos. En la figura 17 se ve esta estimación para softmax con actualización de learning rate por el número de visitas. Se aprecia que el error en los primeros episodios es muy inferior al mostrado por ε-greedy con ε=0.30 y la misma configuración de actualización de learning rate. Esta tendencia se repite para la otra configuración que está recogida en el apéndice C.

Esto indica que los primeros 30 episodios de softmax, tienen una duración regular y son generalmente cortos porque no tienen mucha recompensa acumulada. Una vez que ya empieza a ejecutar la tarea bien empieza a aumentar la varianza entre episodios. En cambio ε-greedy (figura 18) muestra esta varianza desde el principio.

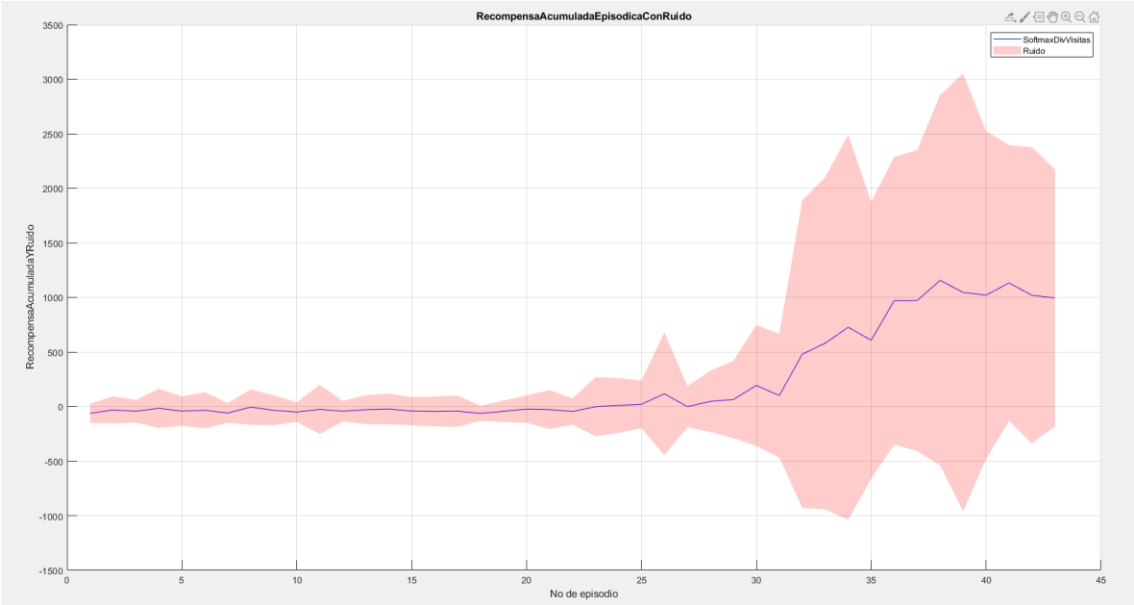

Figura 17. Recompensa acumulada por episodio para SARSA con softmax y learning rate actualizado por el número de visitas del estado con ruido.

Estas dos figuras son especialmente representativas de las tendencias de cada algoritmo. Se aprecia como softmax hace que se acorten los primeros episodios por su tendencia exploratoria, mientras que ε-greedy hace que empiecen a alargarse desde el principio por su tendencia exploratoria.

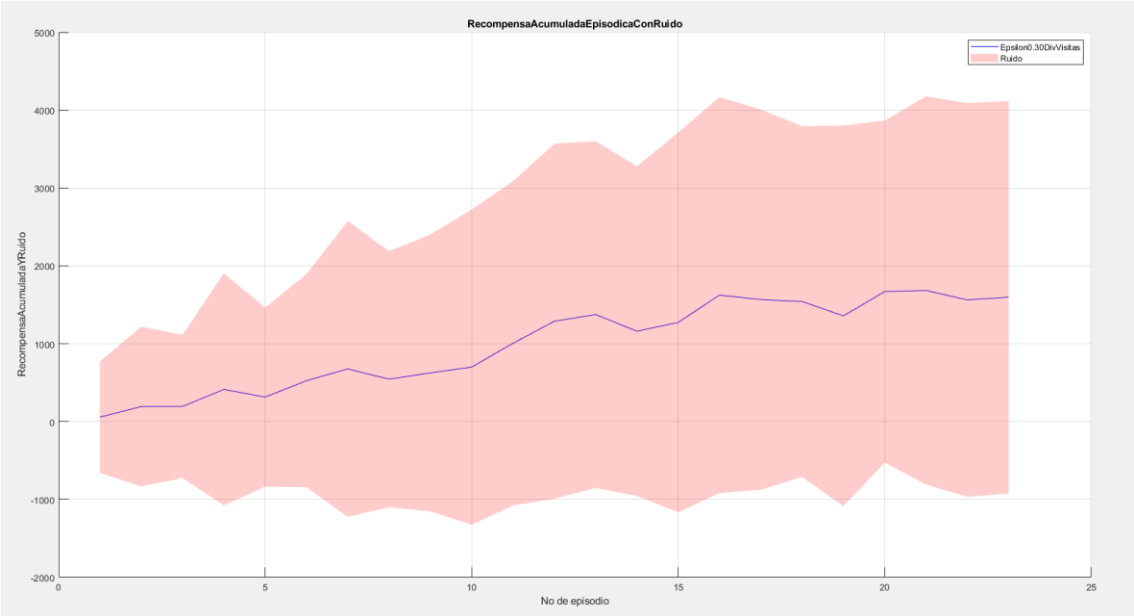

Figura 18. Recompensa acumulada por episodio para SARSA con ε-greedy 0.30 y learning rate actualizado por el número de visitas del estado con ruido.

Finalmente, equilibrando todos los factores la estrategia que parece ideal para este problema es softmax asociado al número de visitas. A igualdad de estrategia converge más rápido que ε-greedy a la política final gracias a su tendencia exploratoria durante los primeros pasos. Además, el acumulado de recompensa no se queda lejos del arrojado por ε-greedy.

6

Análisis del resultado

6.1 Establecimiento parámetros de aprendizaje

En este proyecto se ha utilizado una metodología basada en el análisis de datos. A partir de la implementación de diferentes configuraciones de un controlador SARSA, y el análisis de los resultados teóricos obtenidos, se entenderán las capacidades del aprendizaje en este tipo de problemas.

Por tanto, se han desarrollado varias versiones con distintos parámetros, como se ha visto en el capítulo 4. En este apartado se calculará una Q(s) media resultante de 50 aprendizajes de cada configuración y se lanzará una explotación de 10.000 pasos.

6.2 Estudio de la matriz Q(s,a)

Por el equilibrio entre velocidad de convergencia y la recompensa acumulada, softmax con actualización de learning rate mediante número de visitas por estado se postula como una de las versiones que mejor se ha desenvuelto en el aprendizaje de la tarea propuesta. Por tanto en esta sección se hará un análisis de la matriz Q(s,a) media obtenida de todos los aprendizaje. La tabla se presenta dividida en 2 partes, con la mejor acción para cada estado resaltado en amarillo pálido. Para mostrar los resultados en la tabla también aparecen el rango de distancias (nd) y de ángulos (na) que representa cada estado discreto. El resto de las tablas para todas las configuraciones se recogen en el apéndice D.

En la tabla 5 están recogidos los estados más cercanos al estado recompensado (17). En general tiene mayores valores que los de la tabla 6.

En los estados con una orientación inclinada hacia la pared como el 2,3,10,11,18,19,26,27 aprende a rotar en sentido horario corrigiendo la orientación. De igual manera cuando la orientación es inclinada opuestamente a la pared el robot aprende a corregir su posición girando en sentido antihorario, como se ve en los estados 15,16,22,23,24,30,31,32,38,39 y 40.

Además, se ve que el estado con mayor valor es el 17 con 33.026 que es el favorable y aprende a desplazarse recto.

nd	na	Estado	Acción		
			Adelante	Giro horario	Giro antihorario
1	1	1	-4.73163	-5.4911	-8.2414
	2	2	-12.8145	-11.4551	-27.1092
	3	3	-35.9465	-22.00972	-45.5827
	4	4	-65.2603	-91.1316	-66.2377
	5	5	0	0	0
	6	6	0	0	0
	7	7	-0.1030	-0.3124	-0.0990
	8	8	0.2486	0.2579	-2.8683
2	1	9	3.8105	3.9081	3.5852
	2	10	0.0117	1.3332	-1.6207
	3	11	-7.0040	-5.4214	-5.8665
	4	12	-32.5909	-36.3034	-23.2125
	5	13	0	0	0
	6	14	-2.0175	-2.6358	0.2620
	7	15	1.4063	-0.0948	2.0825
	8	16	5.5415	4.0863	4.0766
3	1	17	33.0326	18.2224	18.0286
	2	18	14.5502	20.4595	4.5792
	3	19	0.0472	2.3390	-4.2205
	4	20	-10.0398	-13.5871	-20.6754
	5	21	0	0	0
	6	22	-0.3387	-27.6020	1.06755
	7	23	4.5052	0.03574	12.8041
	8	24	16.8923	6.3174	23.7770
4	1	25	3.4379	2.0942	3.1134
	2	26	8.1207	12.3284	4.4320
	3	27	1.8543	7.8970	-6.5431
	4	28	-4.7702	-3.4180	-52.3062
	5	29	0	0	0
	6	30	-9.2075	-22.4924	0.70217
	7	31	-2.4931	-10.4526	2.3520
	8	32	1.0845	-3.3021	3.9442

Tabla 5. *Tabla Q^{π} (S.A) para cada uno de los pares estado-acción para SoftmaxDivVsitas(parte 1).*

nd	na	Esto	Acción		
			Avance	Giro horario	Giro antihorario
5	1	33	-1.8540	-4.1093	-2.2612
	2	34	0.3672	0.4988	-4.5100
	3	35	0.2912	2.7907	-15.0777
	4	36	-3.3644	-17.8784	-67.8651
	5	37	0	0	0
	6	38	-10.5271	-102.5833	-3.2852
	7	39	-6.8913	-36.4919	-1.3661
	8	40	-2.3964	-10.7854	-1.0454
6	1	41	-0.8319	-5.6815	-3.4436
	2	42	-0.2335	-0.3337	-28.4427
	3	43	-1.5077	-0.8553	-90.9163
	4	44	-12.1905	-0.0816	-92.5000
	5	45	0	0	0
	6	46	-54.6958	-75	-8.3863
	7	47	-14.5489	-103.71250	-2.7012
	8	48	-2.5164	-49.1905	-2.2261
7	1	49	-11.8750	-2.5000	0
	2	50	0	0	0
	3	51	0	-0.0003	0
	4	52	0	0	-15
	5	53	0	0	0
	6	54	-10	-5	0.0293
	7	55	-12.5000	-5	-9
	8	56	-5	0	-0.4551

Tabla 6. *Tabla Q^{π} (S.A) alcanzada para cada uno de los pares estado-acción (parte 2).*

6.3 Explotación del aprendizaje

Una vez visto como el significado de los valores en la tabla Q(S,A), se procede a lanzar una ejecución, pero esta vez no se realizará el proceso del aprendizaje, sino que en cada estado se escogerá la acción más óptima que es la que se ha visto en la ecuación (8) del apartado 3.3.2, es decir, se coge la acción más valiosa según la política aprendida.

Para esta ejecución se han almacenado las recompensas obtenidas acumulándose en un vector cada 100 iteraciones. Se van a agrupar los episodios según su valor de recompensa acumulado para ver si el robot es capaz de mantenerse en la banda recompensada.

Las condiciones iniciales del experimento serán las mismas, que las explicadas en el apartado 4.2. En este caso se hará un experimento de 10k pasos. Los resultados se han guardado en 6 histogramas.

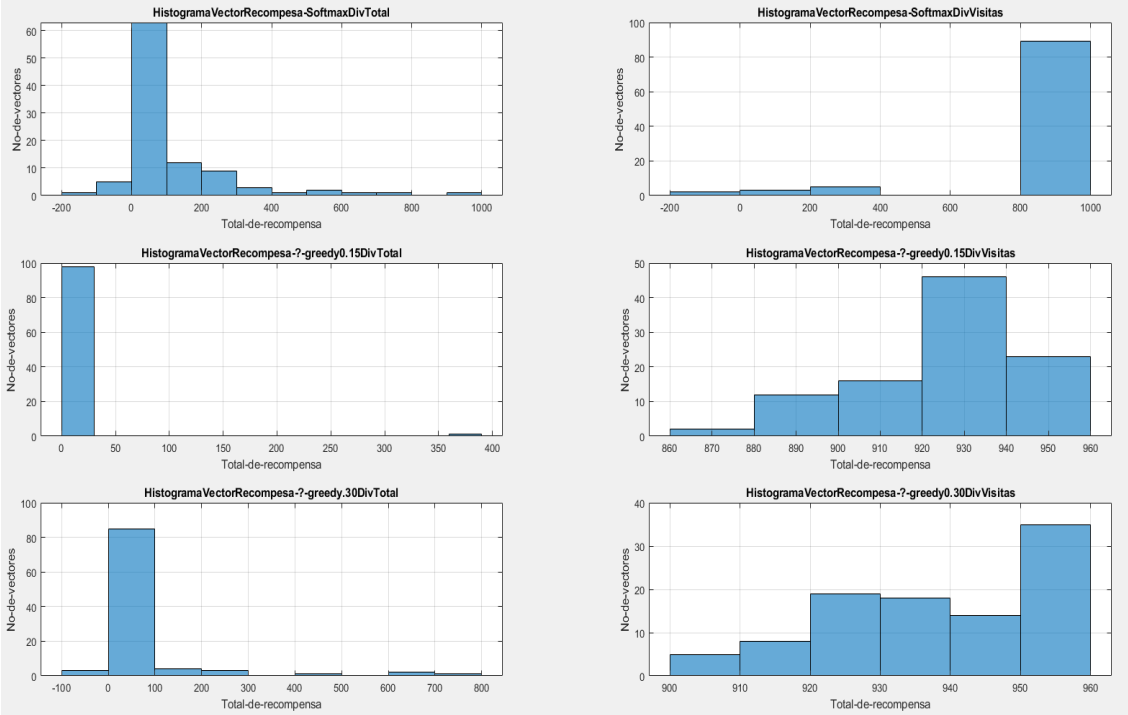

Figura 19. Histogramas resultantes de cada una de las configuraciones ejecutadas. En el eje 0X aparecen los rangos de valores de recompensas y en el eje y el número acumulado de episodios.

En la figura 19 se ve que las mejores implementaciones han resultado ser las tres configuraciones que actualizan el learning rate en base al número de visitas del estado actual, como se había planteado en el capítulo 5, porque son las que más recompensa acumulada tienen (eje x) y más episodios con esas recompensas (eje y).

Entre estas tres configuraciones ε-greedy ha funcionado mejor que softmax. Se puede deber a que las demás configuraciones convergen demasiado rápido a la política final y no les da tiempo a explorar todo el entorno por lo que hay ejecuciones en la que el robot es capaz de mantenerse en la zona recompensada pero no explora el resto de los estados, por lo que al hacer un gran número de experimentos acaban penalizando en el cómputo medio a los estados menos visitados y perjudicando a la política media resultante.

Entre las dos configuraciones que utilizan ε-greedy los resultados son muy similares en la explotación. Sin embargo, como se ha visto en el capítulo 5 la que utiliza ε=0.30 aprende más rápido ya que converge antes a la política final. Por tanto, la mejor configuración en este problema ha sido ε-greedy con ε=0.30 con actualización de learning rate por medio del número de visitas.

7

Conclusiones y futuras mejoras

7.1 Conclusiones

Este proyecto me ha ayudado a profundizar en la teoría de control por computador, además de introducirme en el campo de la IA en el control de sistemas. Este algoritmo ha resultado muy ilustrativo a la hora de reflejar el aprendizaje por refuerzo con su sistema de recompensas y su modelado del entorno.

Durante la carrera en la ETSI solo había modelado sistemas mecánicos o eléctricos, usando controladores clásicos. Por tanto, he sido capaz de ir un paso adelante e implementar una nueva técnica en este problema.

Gracias al simulador facilitado por uno de los tutores de este proyecto, se ha podido sustituir al robot físico y realizar este proyecto de forma totalmente segura en situación de pandemia.

Por el potencial que tiene la herramienta MATLAB y por la gran cantidad de funcionales para el desarrollo y análisis de datos fue el software escogido para la implementación. Gracias a lo aprendido durante la carrera, y a la gran cantidad de documentación que existe en línea sobre este programa no ha resultado complejo montar el algoritmo.

El trabajo se ha desarrollado siguiendo las recomendaciones de los tutores y de algunas propuestas personales que no iban más allá de descubrir técnicas ya usadas en el aprendizaje por refuerzo, pero que gracias a su funcionalidad y lógica se pueden deducir sin tener total conocimiento de los algoritmos en su totalidad.

Aunque se han encontrado dificultades, como por ejemplo la implementación de la estrategia ε-greedy que no arrojaba los resultados esperados, y dificultó el aprendizaje, finalmente siguiendo las recomendaciones de los tutores se terminó por añadir otra técnica de exploración y realizando las modificaciones necesarias para el correcto funcionamiento. No ha sido un camino sencillo, por la naturaleza que tiene el

aprendizaje, a largo plazo y que requiere de tiempo para comprender y analizar los resultados obtenidos, pero, sin embargo, sí que deja unos resultados satisfactorios.

El objetivo será que el robot LEGO MINDSTORMS consiga aprender a desplazarse en un entorno de interiores delimitado por paredes. Para ello se ha desarrollado una arquitectura de control para que el robot, con capacidades limitadas (solo contamos con un sensor lateral de proximidad y un sensor de posición) sea capaz de adaptarse a cualquier entorno de interior. Por tanto, una vez ejecutado, el principal objetivo ha sido analizar la capacidad que tiene el aprendizaje por refuerzo tabular para conseguir aprender a resolver esta tarea.

Se puede concluir con que el learning rate es un parámetro que afecta de manera directa en la política final aprendida y que es importante escoger una buena técnica de actualización para que el agente aprenda satisfactoriamente. Además, se han visto las diferencias entre dos estrategias de selección de acción, sobre todo ha sido especialmente ilustrativo en la forma de converger a la política final.

7.2 Trabajos futuros

El programa de seguimiento de paredes es obviamente una tarea simple y que por sí sola limita mucho la funcionalidad del robot, para trabajos futuros se podría partir de esta base para diseñar el problema de manera que el robot se desplace por un entorno cerrado delimitado por paredes De esta manera se puede obtener más información sobre las distintas versiones de SARSA implementadas.

Por otro lado, también se podría aumentar el estudio realizando una implementación de algún algoritmo clásico de control, que era uno de los objetivos del proyecto pero que finalmente se descartó porque no se podía modelar como un sistema SISO y escapaba de la dimensión de un TFG.

Referencias

[1] R. Hossain, “Control System By Norman nise Sixth Ed.” Accessed: Jan. 30, 2021. [Online]. Available: https://www.academia.edu/35425584/Control_System_By_Norman_nise_Sixth_Ed.

[2] “Applications of Reinforcement Learning in Real World | by garychl | Towards Data Science.” https://towardsdatascience.com/applications-of-reinforcement-learning-in-real-world-1a94955bcd12 (accessed Jan. 29, 2021).

[3] “Ladrillo Inteligente EV3 45500 | MINDSTORMS® | Oficial LEGO® Shop ES.” https://www.lego.com/es-es/product/ev3-intelligent-brick-45500 (accessed Jan. 30, 2021).

[4] Dr. J. A. F. Madrigal, “Asignatura: Control por Computador (2019-20, Todos los grupos).” https://informatica.cv.uma.es/course/view.php?id=3816 (accessed Feb. 06, 2021).

[5] “EV3 Medium Servo Motor 45503 | MINDSTORMS® | Buy online at the Official LEGO® Shop US.” https://www.lego.com/en-us/product/ev3-intelligent-brick-45500 (accessed Jan. 30, 2021).

[6] “LEGO MINDSTORMS EV3 — ev3dev-stretch Linux kernel drivers 19 documentation.” http://docs.ev3dev.org/projects/lego-linux-drivers/en/ev3dev-stretch/ev3.html#input-ports (accessed Feb. 06, 2021).

[7] MathWorks, “MATLAB - El lenguaje del cálculo técnico - MATLAB & Simulink,” *Natick, Massachusetts*, 2018. https://es.mathworks.com/products/matlab.html (accessed Jan. 30, 2021).

[8] Dr. J. A. F. Madrigal, “Lego Mindstorms Minimalist Matlab Library for mobile robot simulation | Dr. Juan-Antonio Fernández-Madrigal.” https://babel.isa.uma.es/jafma/index.php/2020/03/31/lmml/ (accessed Jan. 30, 2021).

[9] L. Pack Kaelbling, M. L. Littman, A. W. Moore, and S. Hall, “Reinforcement Learning: A Survey,” 1996. Accessed: Jan. 30, 2021. [Online].

[10] A. Martínez-Tenor, J. A. Fernández-Madrigal, A. Cruz-Martín, and J. González-Jiménez, “Towards a common implementation of reinforcement learning for multiple robotic tasks,” *Expert Systems with Applications*, vol. 100, pp. 246–259, Feb. 2018, doi: 10.1016/j.eswa.2017.11.011.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction Second edition, in progress*. .

[12] “(PDF) Programación Dinámica y Aprendizaje por Refuerzo-Simulación y aplicación a sistemas electromecánicos.” https://www.researchgate.net/publication/322092684_Programacion_Dinamica_y_Aprendizaje_por_Refuerzo-Simulacion_y_aplicacion_a_sistemas_electromecanicos (accessed Jan. 31, 2021).

[13] “Bellman, R. (1957) Dynamic Programming. Princeton University Press, Princeton, NJ. - References - Scientific Research Publishing.” https://www.scirp.org/reference/ReferencesPapers.aspx?ReferenceID=2187052 (accessed Feb. 06, 2021).

[14] M. Tokic, “Adaptive ε-greedy exploration in reinforcement learning based on value differences,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010, vol. 6359 LNAI, pp. 203–210, doi: 10.1007/978-3-642-16111-7_23.

Apéndice A

Gráficas de distancia euclídea V(s) de los experimentos para cada versión implementada

En este apéndice se recogen todas las gráficas que representan la distancia euclídea entre dos V(s) consecutivas que son usadas en el capítulo 5 de la memoria., para comparar la variación del parámetro learning rate.

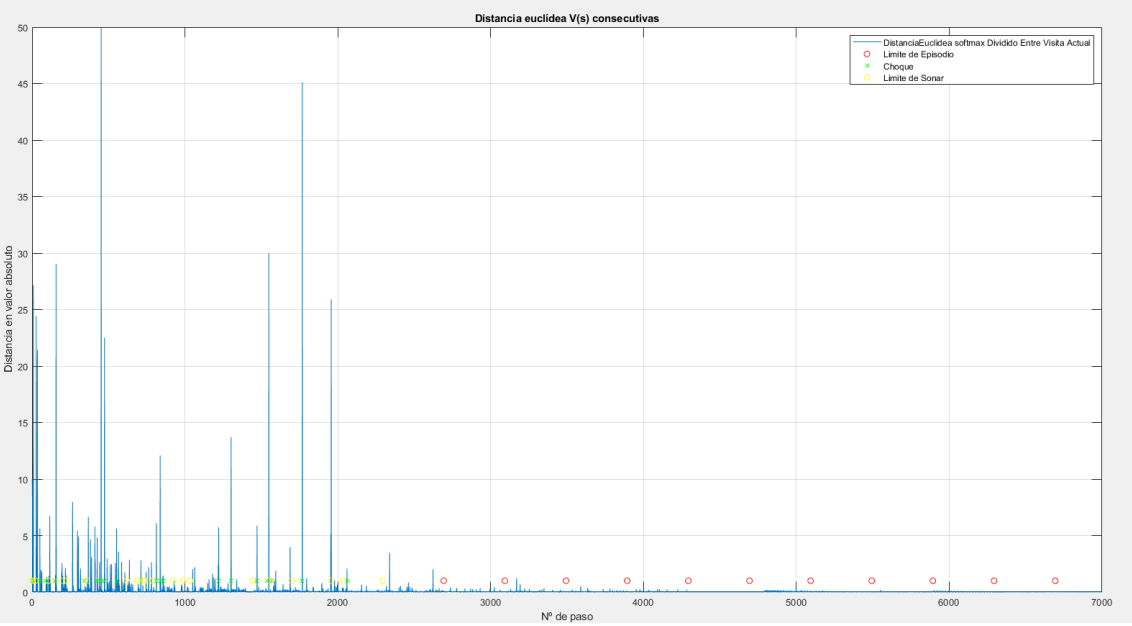

Figura A.1 Distancia euclídea entre V(s) consecutivas para cada paso con estrategia SoftMax y actualización de learning rate por el número de visitas de cada estado.

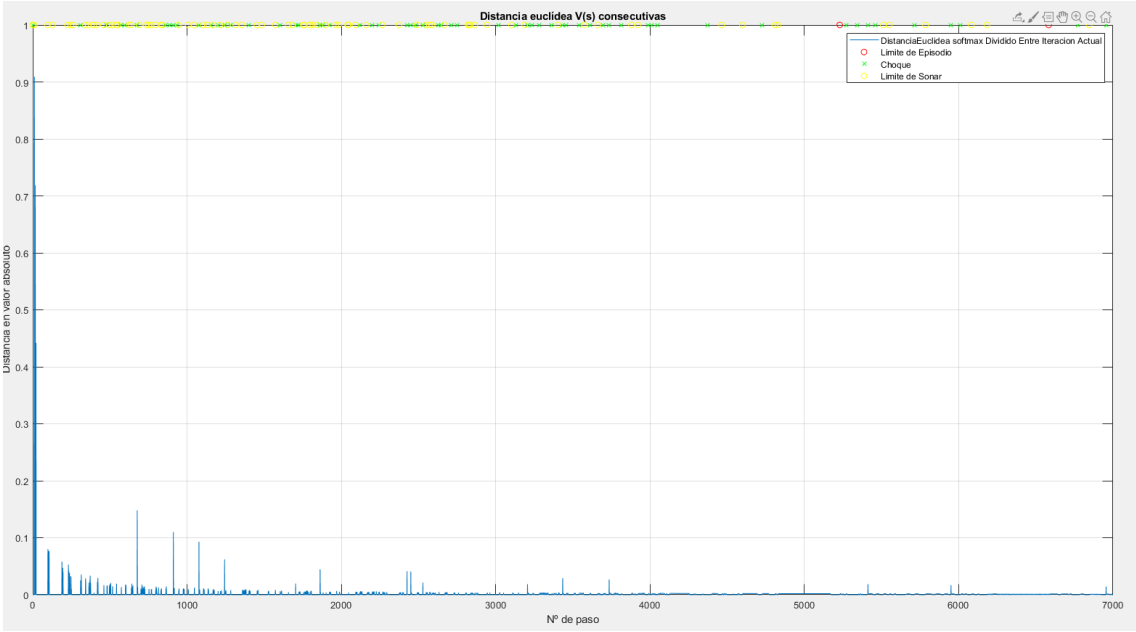

Figura A.2. Distancia euclídea entre V(s) consecutivas para cada paso con estrategia SoftMax y actualización de learning rate por el número de pasos total.

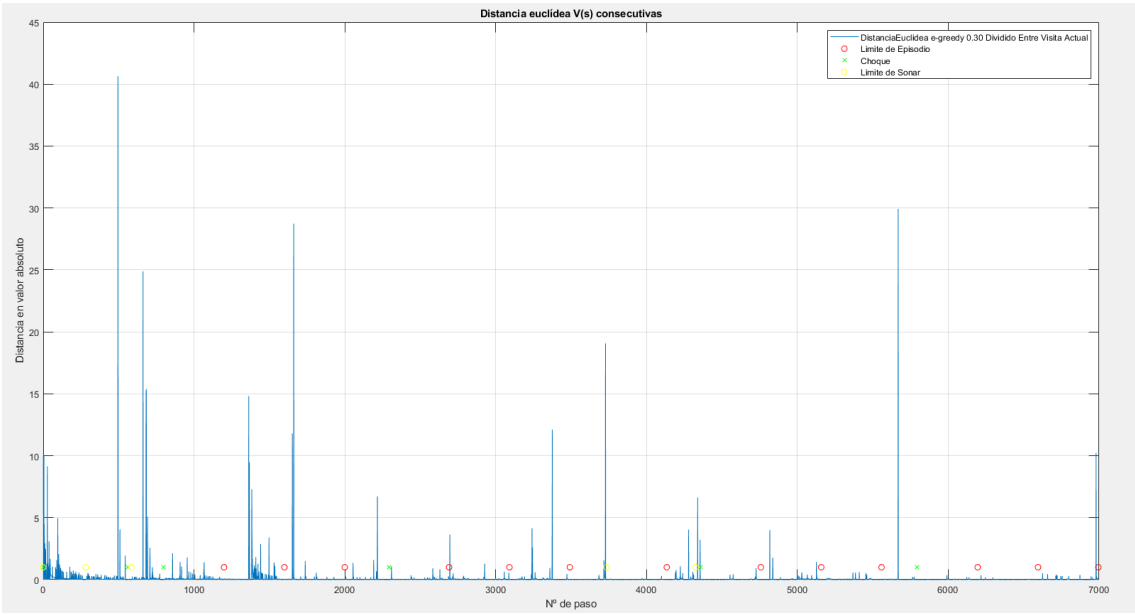

Figura A.3 Distancia euclídea entre V(s) consecutivas para cada paso con estrategia ε-greedy con e=0.30 y actualización de learning rate por el número de visitas de cada estado.

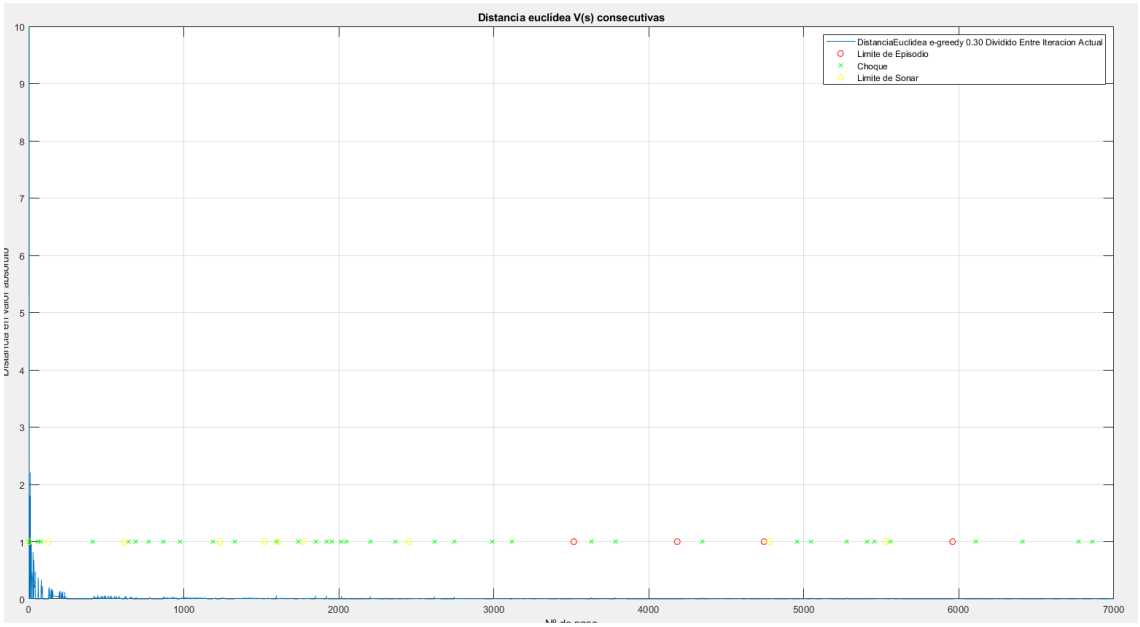

Figura A.4 . Distancia euclídea entre V(s) consecutivas para cada paso con estrategia ε-greedy con e=0.30 y actualización de learning rate por el número de pasos total.

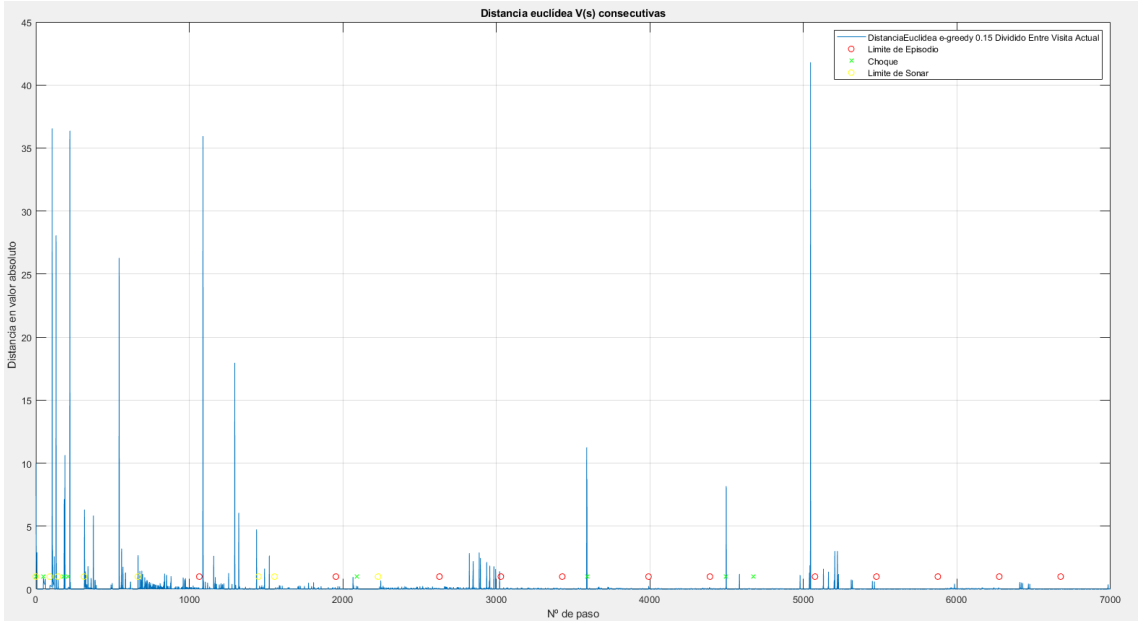

Figura A.5 Distancia euclídea entre V(s) consecutivas para cada paso con estrategia ε-greedy con e=0.15 y actualización de learning rate por el número de visitas de cada estado.

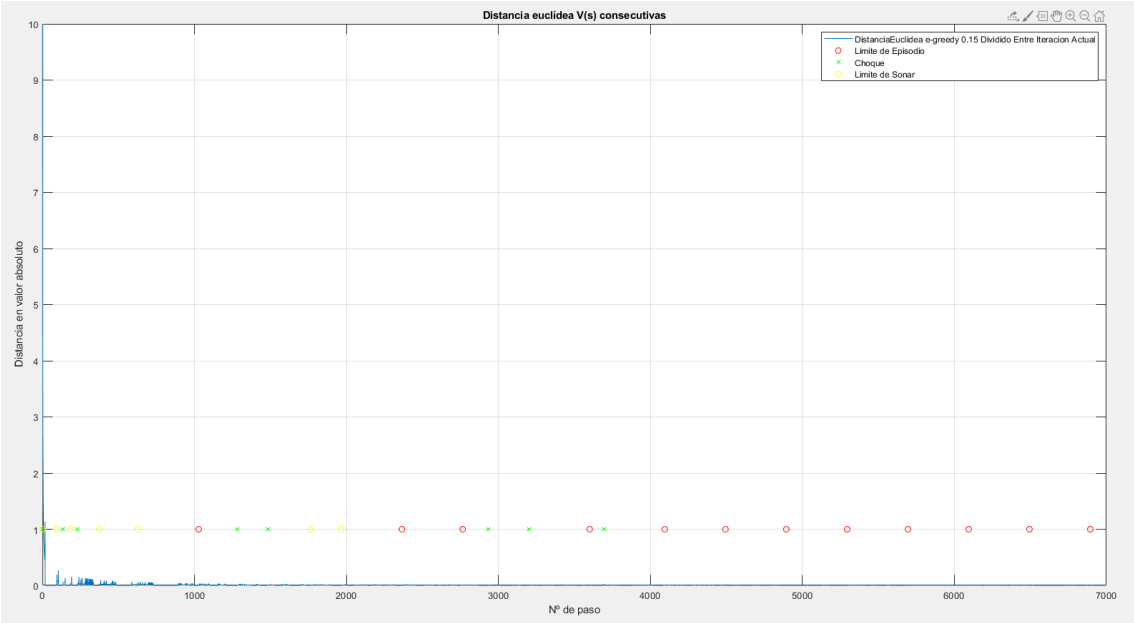

Figura A.6. Distancia euclídea entre V(s) consecutivas para cada paso con estrategia ε-greedy con e=0.15 y actualización de learning rate por el número de pasos total.

Apéndice B

Recompensa acumulada por episodio para todas las configuraciones

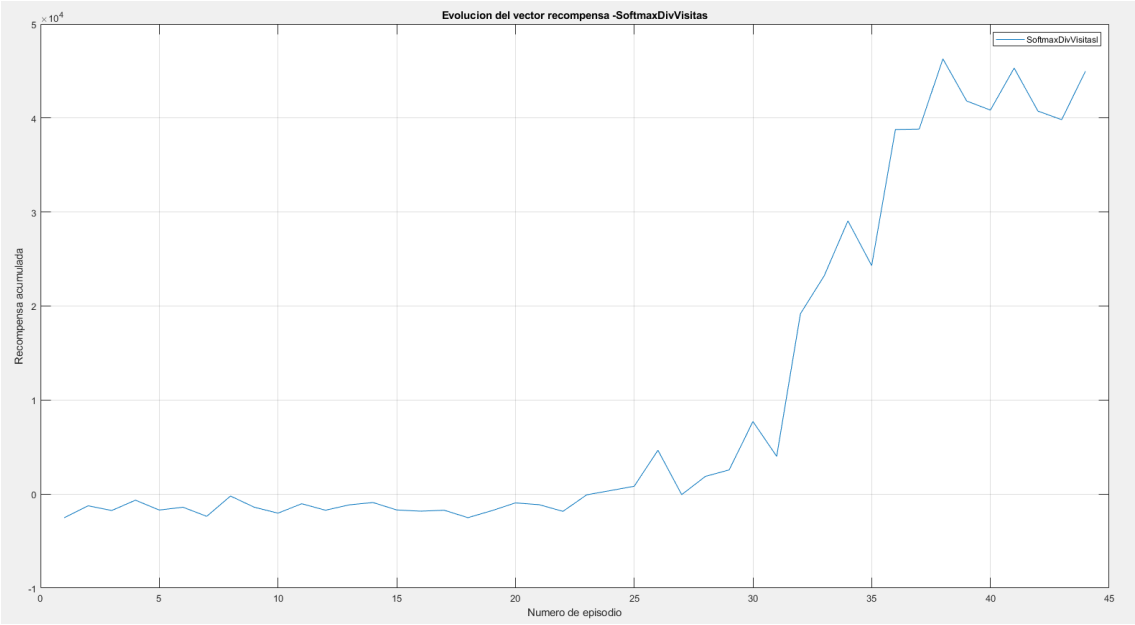

Figura B.1. Recompensa acumulada por episodio para SARSA con Softmax y learning rate actualizado por el número de visitas del estado.

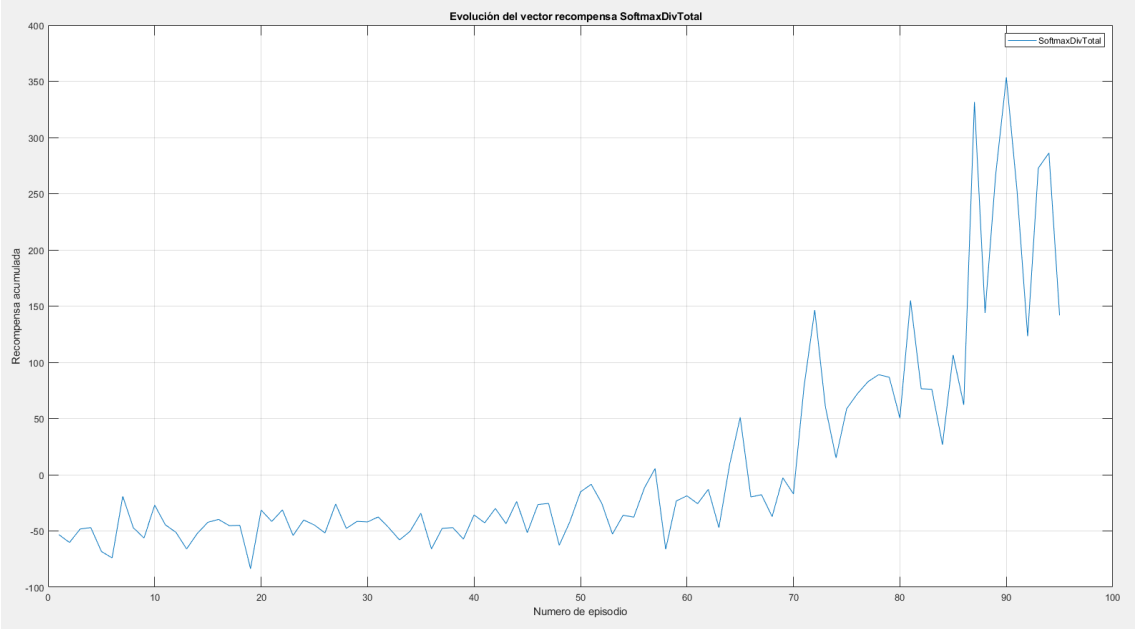

Figura B.2. Recompensa acumulada por episodio para SARSA con Softmax y learning rate actualizado por el número de iteración actual.

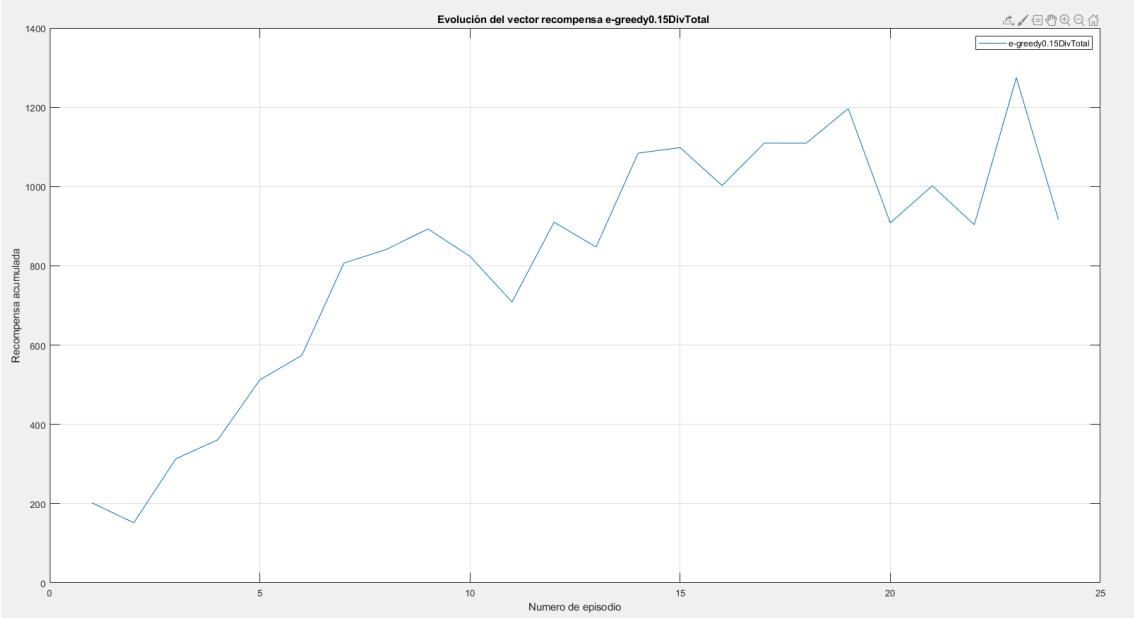

Figura B.3. Recompensa acumulada por episodio para SARSA con ε-greedy 0.15 y learning rate actualizado por el número de iteración actual.

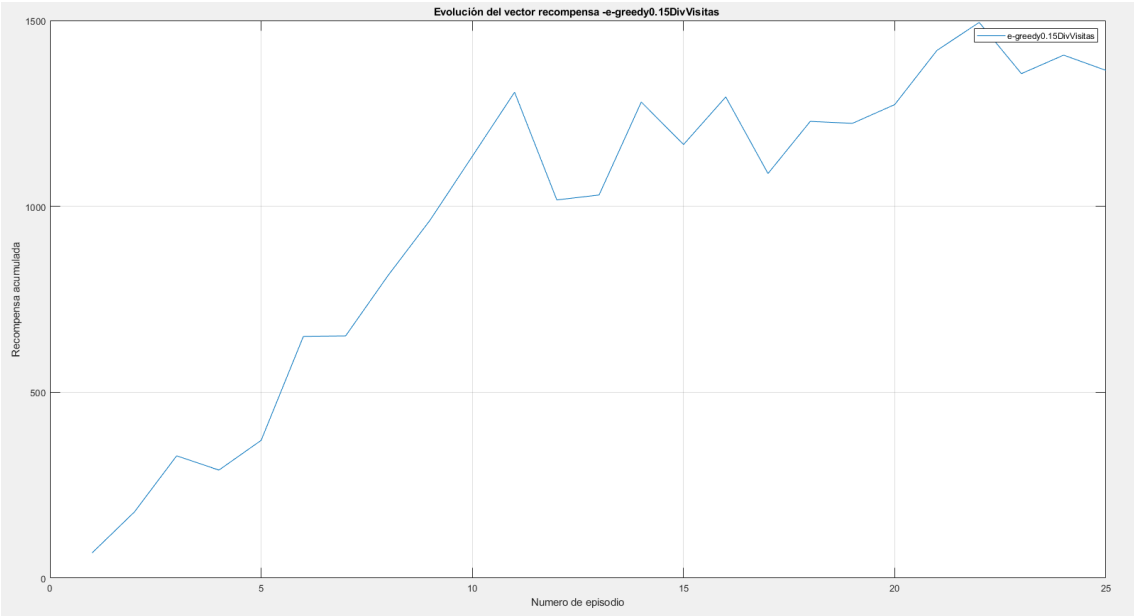

Figura B.4. Recompensa acumulada por episodio para SARSA con ε-greedy 0.15 y learning rate actualizado por el número de visitas del estado.

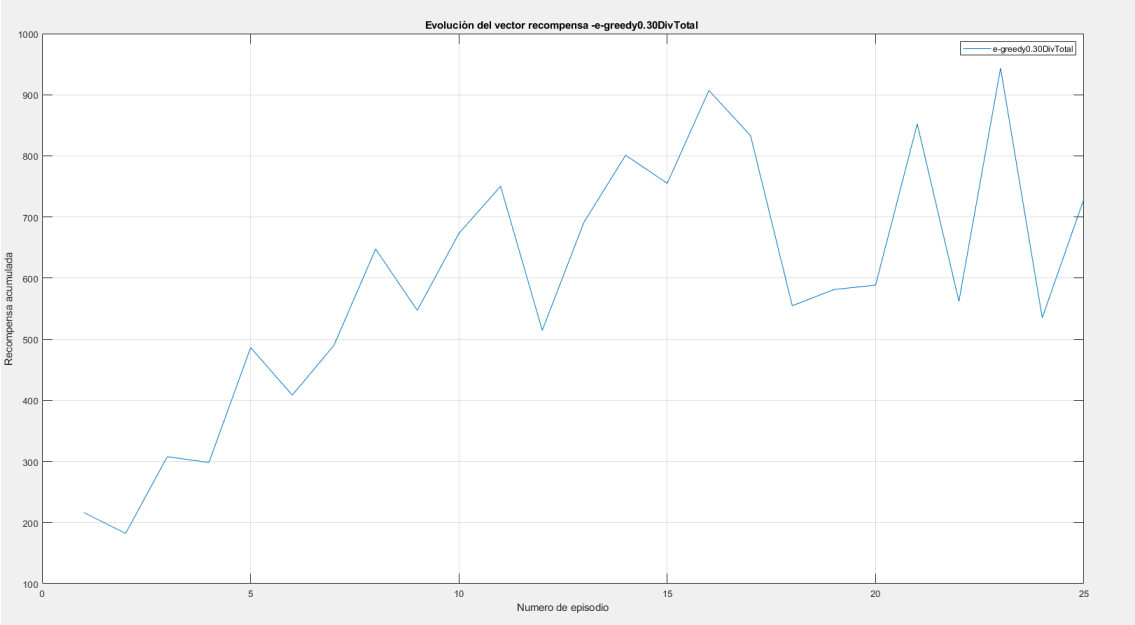

Figura B.520. Recompensa acumulada por episodio para SARSA con ε-greedy 0.30 y learning rate actualizado por el número de iteración actual.

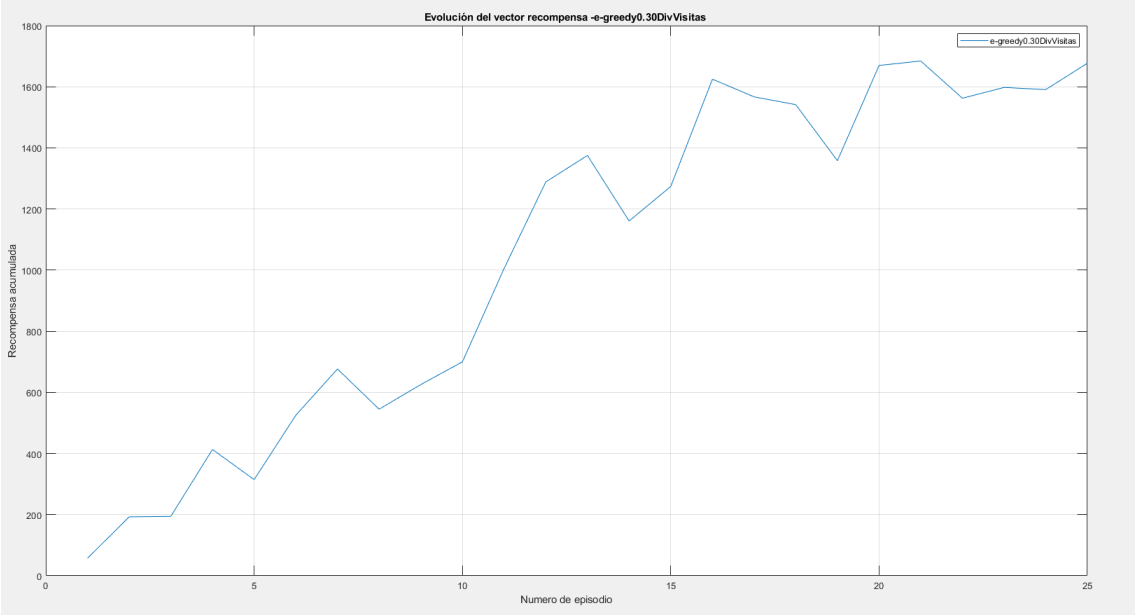

Figura B.6. Recompensa acumulada por episodio para SARSA con ε-greedy 0.30 y learning rate actualizado por el número de visitas del estado.

Apéndice C

Recompensa acumulada por episodio con ruido para todas las configuraciones

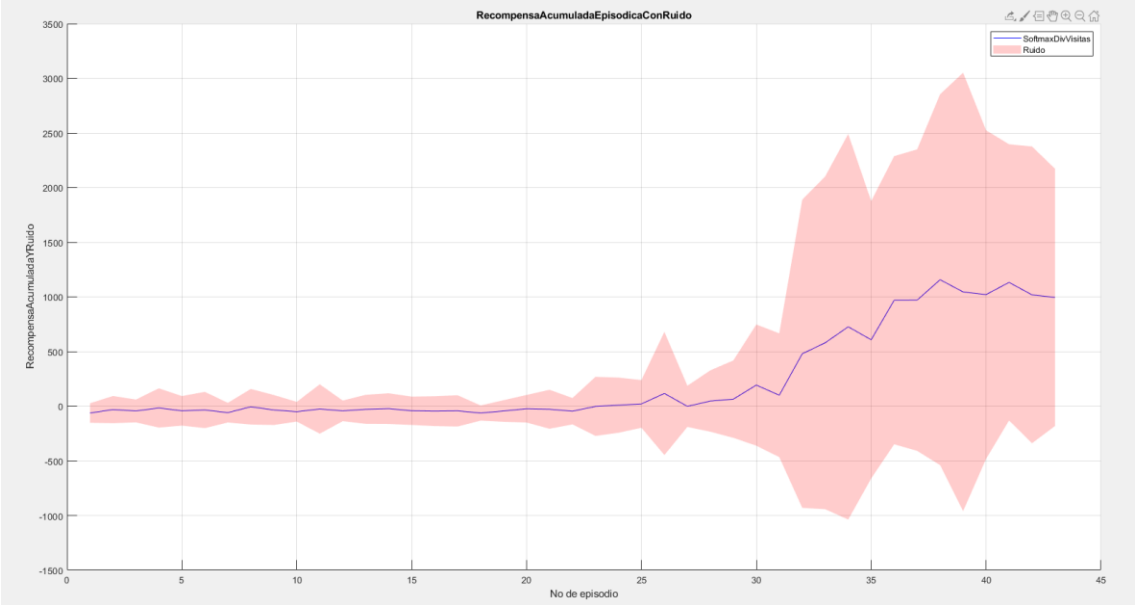

Figura C.1. Recompensa acumulada por episodio para SARSA con Softmax y learning rate actualizado por el número de visitas del estado con ruido.

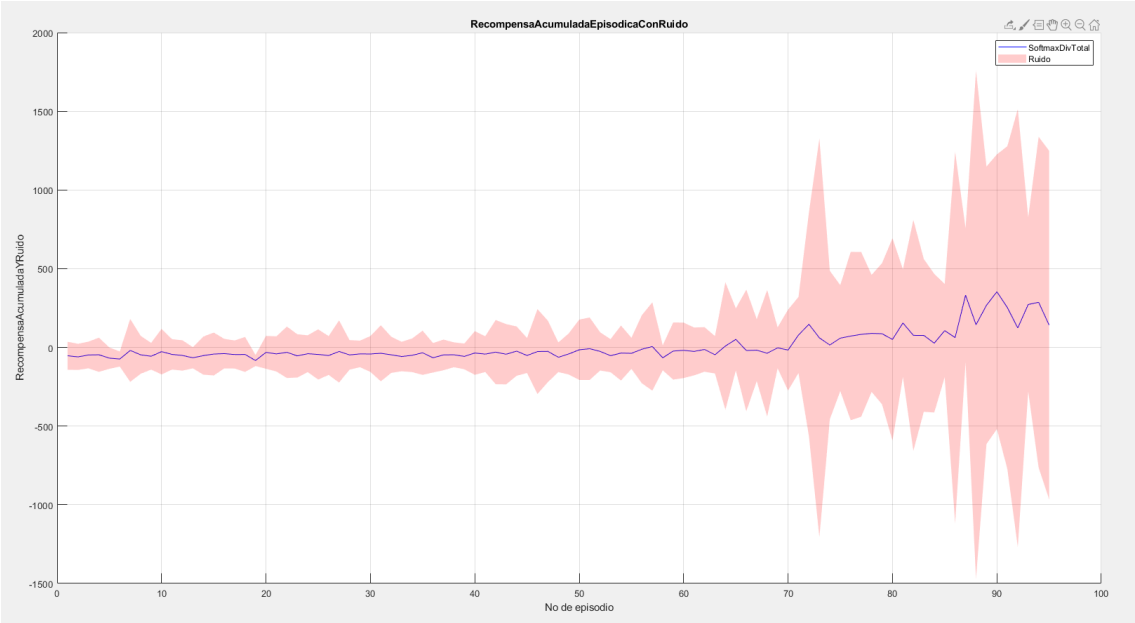

Figura C.2 Recompensa acumulada por episodio para SARSA con Softmax y learning rate actualizado por el número de iteración actual con ruido.

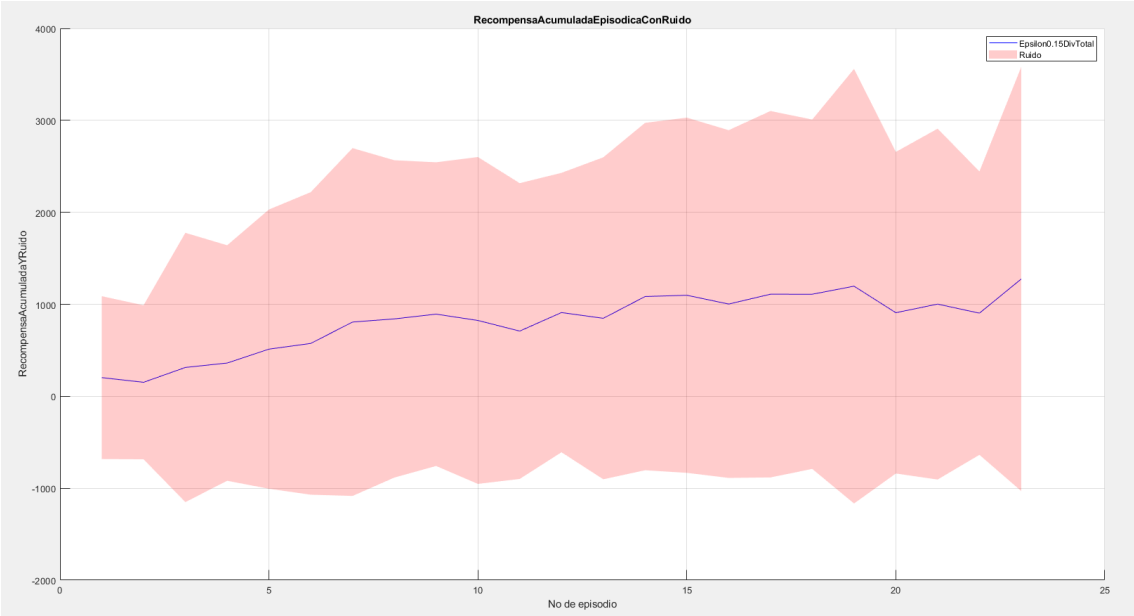

Figura C.3. Recompensa acumulada por episodio para SARSA con ε-greedy 0.15 y learning rate actualizado por el número de iteración actual con ruido.

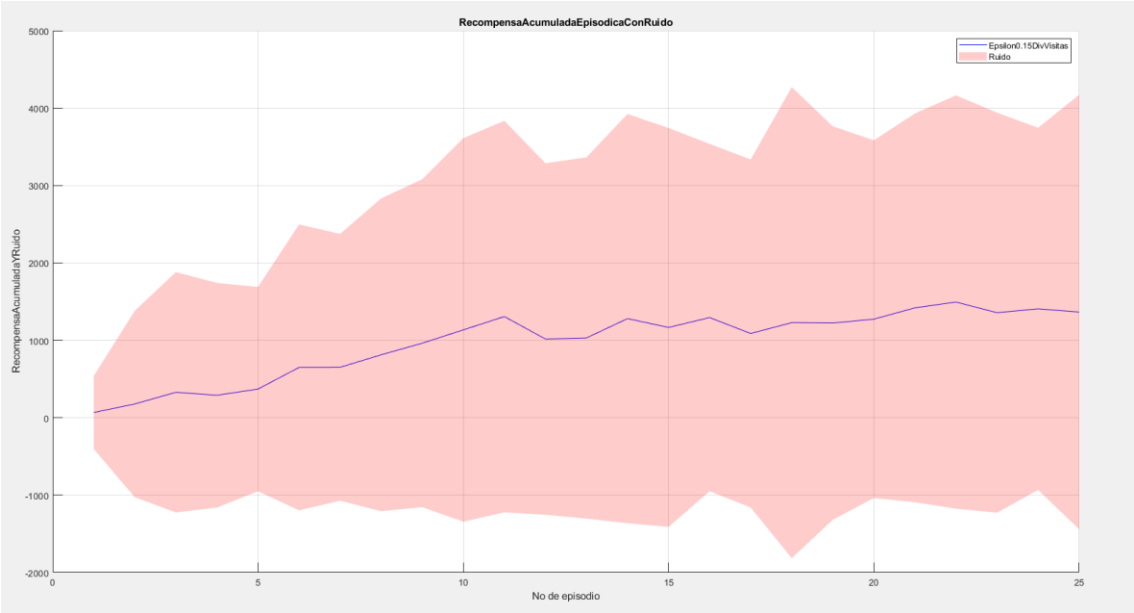

Figura C.4. Recompensa acumulada por episodio para SARSA con ε-greedy 0.15 y learning rate actualizado por el número de visitas del estado con ruido.

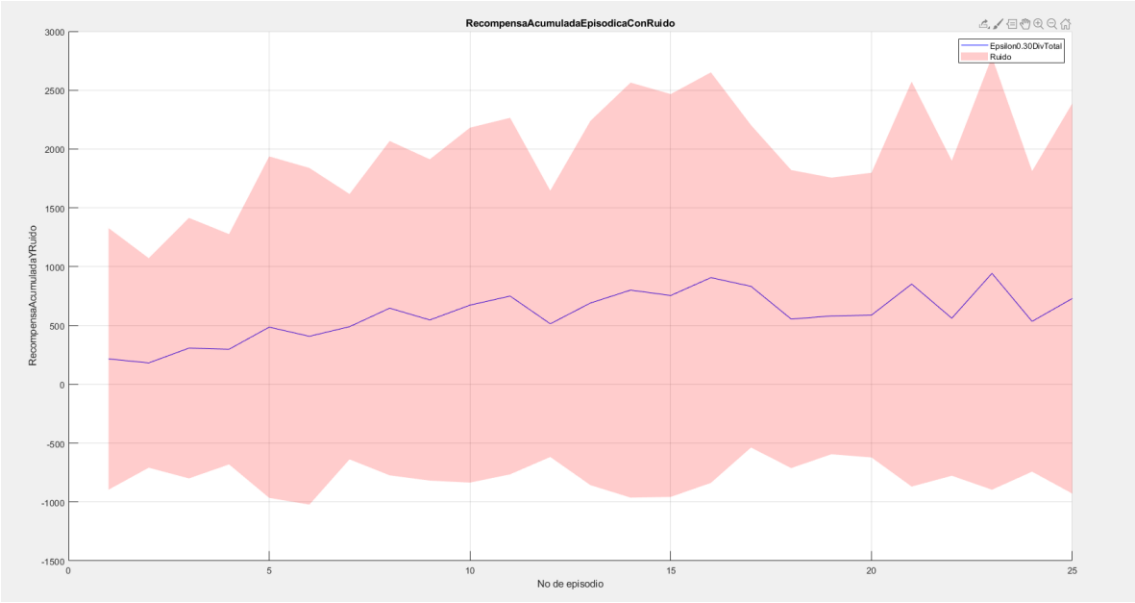

Figura C.5. Recompensa acumulada por episodio para SARSA con ε-greedy 0.30 y learning rate actualizado por el número de iteración actua con ruido.

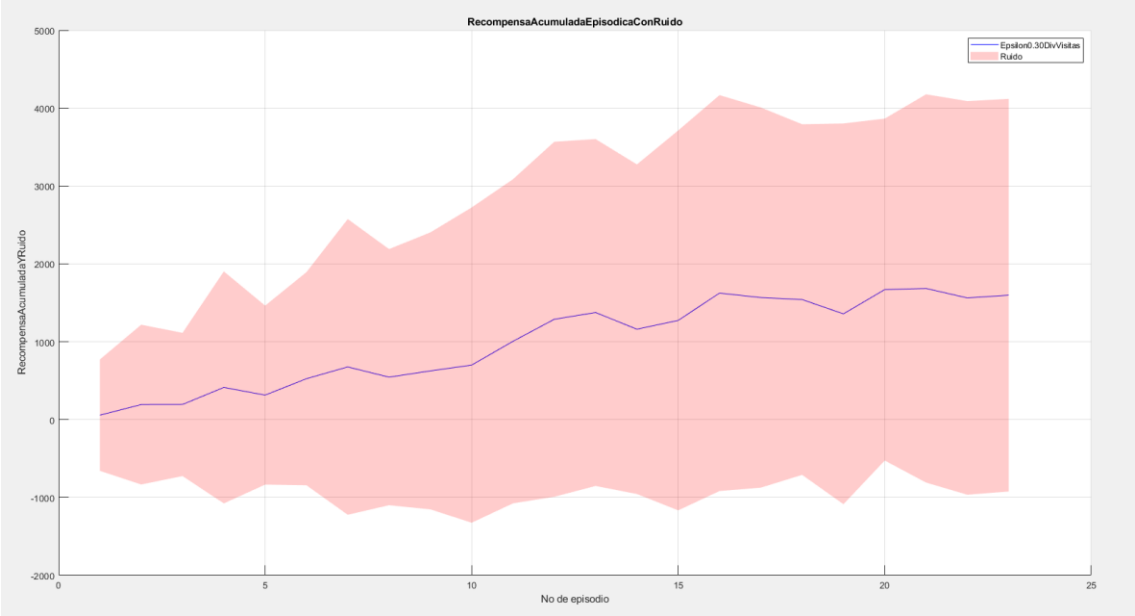

Figura C.6. Recompensa acumulada por episodio para SARSA con ε-greedy 0.30 y learning rate actualizado por el número de visitas del estado con ruido.

Apéndice D

Tabla Q(S,A) media de cada experimento ejecutado.

Nd	na	Estado	Acción		
			Avance	Giro horario	Giro antihorario
1	1	1	-0.0148	-0.2052	-0.0419
	2	2	-0.3566	-0.1930	-0.2941
	3	3	-0.5660	-0.5313	-0.4239
	4	4	-0.4957	-0.4785	-0.4604
	5	5	0	0	0
	6	6	0	0	0
	7	7	3.3729	1.7576	3.1877
	8	8	7.4026	6.4226	-0.0001
2	1	9	0.2666	0.0063	0.0475
	2	10	-0.0001	0.0027	0.0089
	3	11	-0.0015	-0.0031	-0.0025
	4	12	-0.1024	-0.2157	-0.2051
	5	13	0	0	0
	6	14	-3.0588	-5.9248	1.9367
	7	15	1.2077e-06	1.9283e-06	4.9076e-06
	8	16	0.0021	0.0016	0.0121
3	1	17	9.8874	2.4911	2.5614
	2	18	0.2487	2.6647	0.0450
	3	19	0.0008	0.0620	-0.0036
	4	20	-0.0090	-0.0415	-0.7959
	5	21	0	0	0
	6	22	-1.2356e-06	-0.0002	8.9225e-07
	7	23	4.2319e-05	-8.2048e-05	0.0094
	8	24	0.2203	0.0253	1.9511
4	1	25	0.0659	0.0442	0.0024
	2	26	0.0062	0.2244	0.0006
	3	27	0.0002	0.0071	-0.0031
	4	28	-0.0020	-0.0044	-0.7514
	5	29	0	0	0
	6	30	-0.0003	-0.0387	-1.0417
	7	31	-0.0035	-0.0269	0.0064
	8	32	0.0018	-0.0125	0.0973

Tabla D.1 Tabla Q(S,A) media para estrategia Softmax y actualización de learning rate mediante el número de iteración (parte 1).

Nd	Na	Estado	Acción		
			Avance	Giro horario	Giro antihorario
5	1	33	7.2679	-5.4910	-9.8287
	2	34	1.3129e-05	0.0001	-0.0002
	3	35	-0.0018	8.5761e-05	-0.1420
	4	36	-0.0097	-0.0251	-1.5827
	5	37	0	0	0
	6	38	-0.0056	-1.3937	-0.0007
	7	39	-0.0197	-2.4573	-0.0061
	8	40	-0.0004	-0.0158	5.8127e-05
6	1	41	-1.2557e-06	-3.1530e-05	-0.0029
	2	42	-7.6659e-06	-8.8953e-07	-0.0325
	3	43	-9.4573e-05	-4.1925e-05	-0.3358
	4	44	-0.0012	-0.0042	-1.1008
	5	45	0	0	0
	6	46	-0.35504	-2.3990	-0.0033
	7	47	-0.2918	-2.2328	-0.0049
	8	48	-0.0118	-0.1642	-4.1190e-05
7	1	49	0	0	0
	2	50	-1.2717e-10	0	0
	3	51	0	0	-0.0026
	4	52	-5.1738e-09	-3.0642e-10	-0.0373
	5	53	0	0	0
	6	54	-0.0500	-0.0145	-6.5554e-05
	7	55	-0.0170	-0.0066	-1.0670
	8	56	0	0	0

Tabla D.2 Tabla Q(S,A) media para estrategia Softmax y actualización de learning rate mediante el número de iteración (parte 2)..

nd	na	Estado	Acción		
			Avance	Giro horario	Giro antihorario
1	1	1	-4.73163	-5.4911	-8.2414
	2	2	-12.8145	-11.4551	-27.1092
	3	3	-35.9465	-22.00972	-45.5827
	4	4	-65.2603	-91.1316	-66.2377
	5	5	0	0	0
	6	6	0	0	0
	7	7	-0.1030	-0.3124	-0.0990
	8	8	0.2486	0.2579	-2.8683
2	1	9	3.8105	3.9081	3.5852
	2	10	0.0117	1.3332	-1.6207
	3	11	-7.0040	-5.4214	-5.8665
	4	12	-32.5909	-36.3034	-23.2125
	5	13	0	0	0
	6	14	-2.0175	-2.6358	0.2620
	7	15	1.4063	-0.0948	2.0825
	8	16	5.5415	4.0863	4.0766
3	1	17	33.0326	18.2224	18.0286
	2	18	14.5502	20.4595	4.5792
	3	19	0.0472	2.3390	-4.2205
	4	20	-10.0398	-13.5871	-20.6754
	5	21	0	0	0
	6	22	-0.3387	-27.6020	1.06755
	7	23	4.5052	0.03574	12.8041
	8	24	16.8923	6.3174	23.7770
4	1	25	3.4379	2.0942	3.1134
	2	26	8.1207	12.3284	4.4320
	3	27	1.8543	7.8970	-6.5431
	4	28	-4.7702	-3.4180	-52.3062
	5	29	0	0	0
	6	30	-9.2075	-22.4924	0.70217
	7	31	-2.4931	-10.4526	2.3520
	8	32	1.0845	-3.3021	3.9442

Tabla D.3. Tabla Q(S,A) media con estrategia SoftMax y actualización de learning rate por el número de visitas de cada estado (parte 1).

Nd	Na	Estado	Acción		
			Avance	Giro horario	Giro antihorario
5	1	33	-1.8540	-4.1093	-2.2612
	2	34	0.3672	0.4988	-4.5100
	3	35	0.2912	2.7907	-15.0777
	4	36	-3.3644	-17.8784	-67.8651
	5	37	0	0	0
	6	38	-10.5271	-102.5833	-3.2852
	7	39	-6.8913	-36.4919	-1.3661
	8	40	-2.3964	-10.7854	-1.0454
6	1	41	-0.8319	-5.6815	-3.4436
	2	42	-0.2335	-0.3337	-28.4427
	3	43	-1.5077	-0.8553	-90.9163
	4	44	-12.1905	-0.0816	-92.5000
	5	45	0	0	0
	6	46	-54.6958	-75	-8.3863
	7	47	-14.5489	-103.71250	-2.7012
	8	48	-2.5164	-49.1905	-2.2261
7	1	49	-11.8750	-2.5000	0
	2	50	0	0	0
	3	51	0	-0.0003	0
	4	52	0	0	-15
	5	53	0	0	0
	6	54	-10	-5	0.0293
	7	55	-12.5000	-5	-9
	8	56	-5	0	-0.4551

Tabla D.4.. Tabla Q(S,A) media con estrategia SoftMax y actualización de learning rate por el número de visitas de cada estado (parte 2).

Nd	na	Estado	Acción		
			Avance	Giro horario	Giro antihorario
1	1	1	-0.0388	-0.0249	-0.0431
	2	2	-0.4370	-0.0351	-0.1514
	3	3	-0.7238	-0.3801	-0.0815
	4	4	-0.3669	-0.2077	-0.0646
	5	5	0	0	0
	6	6	0	0	0
	7	7	1.2958e-09	2.1808e-09	2.2723e-06
	8	8	8.9422e-05	1.1191e-05	0.0005
2	1	9	0.7244	0.0148	0.1001
	2	10	0.0003	0.0293	0.0034
	3	11	-0.0006	-0.0001	-2.3168
	4	12	-0.0649	-0.0218	-0.0287
	5	13	0	0	0
	6	14	0	0	0
	7	15	2.9175	6.0341e-07	3.26035e-05
	8	16	0.01251	0.0023	0.0206
3	1	17	20.5856	3.0499	2.7138
	2	18	0.0979	3.4881	0.0216
	3	19	0.0001	0.0044	2.1622
	4	20	-0.0117	-0.0036	-0.00863
	5	21	0	0	0
	6	22	3.6323e-11	-2.3370e-06	7.9947e-07
	7	23	7.9170e-06	7.7221e-07	0.0062
	8	24	0.1017	0.0163	3.3723
4	1	25	0.2819	0.0492	0.0090
	2	26	0.0095	0.22580	0.0004
	3	27	5.2492	0.00572	4.0947
	4	28	-0.0042	-0.001	-0.0147
	5	29	0	0	0
	6	30	-5.2902e-07	-0.0020	1.7018e-07
	7	31	-0.0004	-2.0378e-06	0.0008
	8	32	0.0003	-0.0009	0.0657

Tabla 5. Tabla Q(S,A) media con estrategia de ε-greedy 0.30 y actualización por el número iteración actual (parte 1).

Nd	Na	Estado	Acción		
			Avance	Giro horario	Giro antihorario
5	1	33	6.8573e-05	-0.0001	2.1468e-05
	2	34	0.0003	0.0011	-1.6696e-06
	3	35	3.3930e-07	0.0001	-0.0054
	4	36	-0.0014	-0.0009	-0.0252
	5	37	0	0	0
	6	38	-7.2091e-06	-0.0187	3.5613e-08
	7	39	-0.0018	-0.6087	4.1571e-05
	8	40	-2.0952e-05	-0.0004	0.0003
6	1	41	-2.1868e-05	-0.0125	-0.00316
	2	42	-8.8466e-06	-2.8557e-06	-0.0461
	3	43	-1.6504e-05	1.9054e-07	-0.0685
	4	44	-0.0035	7.9671e-11	-0.0112
	5	45	0	0	0
	6	46	-0.0782	-0.1157	-4.4445e-10
	7	47	-0.6711	-0.9687	-0.0001
	8	48	-0.1000	-0.26014	-3.81055e-05
7	1	49	0	0	0
	2	50	0	0	0
	3	51	0	0	0
	4	52	0	0	-0.0004
	5	53	0	0	0
	6	54	-0.1823	0	0
	7	55	-0.04942	0	0
	8	56	-0.0183	0	0

Tabla D.6. Tabla Q(S,A) media con estrategia de ε-greedy 0.30 y actualización por el número iteración actual (parte 2).

Nd	na	Estado	Acción		
			Avance	Giro horario	Giro antihorario
1	1	1	-2.1526	-2.2217	-4.5320
	2	2	-18.7702	-8.11780	-23.1732
	3	3	-38.3465	-31.1642	-38.2384
	4	4	-80.6730	-77.7536	-50.6745
	5	5	0	0	0
	6	6	0	0	0.0641
	7	7	1.1252	0.2769	0.5965
	8	8	2.3890	1.7785	1.2096
2	1	9	5.9352	12.0512	10.4634
	2	10	1.6147	5.9190	3.6658
	3	11	-2.1232	-0.7555	-1.0846
	4	12	-25.7730	-22.9395	-15.0574
	5	13	0	0	0
	6	14	0.25308	0.0467	0.26606
	7	15	4.3917	3.24753	5.29314
	8	16	10.5517	11.5092	9.33690
3	1	17	42.6732	33.0752	31.5971
	2	18	21.3514	36.34114	18.3391
	3	19	1.6377	13.49587	2.2609
	4	20	-5.9289	-6.8969	-7.5415
	5	21	0	0	0
	6	22	-0.0490	-3.5649	2.0920
	7	23	4.8442	6.32014	26.74608
	8	24	24.8620	18.7966	41.2806
4	1	25	9.1077	10.0457	16.1653
	2	26	14.7690	28.3697	12.9537
	3	27	3.37160	20.4827	4.11844
	4	28	-4.0206	-1.7556	-8.3333
	5	29	0	0	0
	6	30	-3.1485	-15.9393	2.3620
	7	31	0.9866	-0.0174	11.3957
	8	32	2.72402	4.4855	12.82336

Tabla D.7 Tabla Q(S,A) media con estrategia de ε-greedy 0.30 y actualización por el númerode visita (parte 1).

Nd	Na	Estado	Acción		
			Avance	Giro horario	Giro antihorario
5	1	33	0.2931	-0.0424	0.8765
	2	34	1.4523	3.1489	0.8225
	3	35	2.6917	7.2849	-1.9807
	4	36	-0.8554	1.48561	-3.75
	5	37	0	0	0
	6	38	-2.0730	-42.6474	0.40243
	7	39	-2.0776	-23.5206	1.3072
	8	40	-0.2583	-1.8259	0.66453
6	1	41	-0.28029	-5.4855	-3.7457
	2	42	0.1488	-0.0186	-4.2730
	3	43	-0.6689	-0.2569	-30.0495
	4	44	0.3198	0.1617	-7.5
	5	45	0	0	0
	6	46	-40.5245	-72.5	-1.5568
	7	47	-28.9437	-97.2252	-1.6633
	8	48	-6.7805	-40.6623	-1.0400
7	1	49	-5	0	-2.5
	2	50	-0.0015	0	0
	3	51	0	0	0
	4	52	0	0	0
	5	53	0	0	0
	6	54	-5	-2.5	0
	7	55	-15	0	0
	8	56	-15	-7.5	0

Tabla D.8. Tabla Q(S,A) media con estrategia de ε-greedy 0.30 y actualización por el número de visita (parte 2).

Nd	na	Estado	Acción		
			Avance	Giro horario	Giro antihorario
1	1	1	-3.3772	-1.5003	-2.9434
	2	2	-13.4950	-3.9049	-15.3429
	3	3	-29.8569	-31.4921	-33.6500
	4	4	-75.9661	-69.9124	-53.3704
	5	5	0	0	0
	6	6	0	0	0
	7	7	1.5939	0.1387¡	0.8551
	8	8	3.3493	2.7372	1.7492
2	1	9	5.0058	11.0343	10.8337
	2	10	2.34878	5.97234	3.80575
	3	11	-0.7325	0.4027	-0.3845
	4	12	-15.3089	-20.3912	-9.7267
	5	13	0	0	0
	6	14	0	0	0.1719
	7	15	1.9574	3.0255	4.6088
	8	16	6.7913	9.83625	10.0971
3	1	17	46.8736	35.2353	35.0667
	2	18	13.2343	39.1904	18.1702
	3	19	0.7305	6.4482	0.4773
	4	20	-2.9938	-3.0847	-4.7916
	5	21	0	0	0
	6	22	0.0862	0.0230	1.3173
	7	23	2.0774	1.0600	21.2123
	8	24	14.0885	18.26034	45.97831
4	1	25	7.20532	11.1121	19.0273
	2	26	13.0014	31.8042	14.1892
	3	27	1.1821	18.6557	1.5403
	4	28	-8.0792	-6.5391	-2.5000
	5	29	0	0	0
	6	30	-0.1828	-4.1146	0.60381
	7	31	0.3076	0.26220	5.8817
	8	32	1.3328	4.0452	14.8158

Tabla D.9 Tabla Q(S,A) media con estrategia de ε-greedy 0.15 y actualización por el número de visita (parte 1).

Nd	Na	Estado	Acción		
			Avance	Giro horario	Giro antihorario
5	1	33	0.0788	0.2019	0.9168
	2	34	1.2396	2.5943	0.1556
	3	35	0.4771	3.2196	-1.1104
	4	36	-7.7045	-4.7500	0
	5	37	0	0	0
	6	38	-0.5408	-20	0.1235
	7	39	-1.1076	-16.9642	1.6608
	8	40	-0.1569	-2.3671	0.6650
6	1	41	-0.5962	-6.0931	-4.4623
	2	42	0.1056	0.1298	-6.6290
	3	43	0.1571	0.3984	-20
	4	44	-5.1619	-4.1948	0
	5	45	0	0	0
	6	46	-26.2082	-40	-0.1384
	7	47	-22.6781	-86.6666	-1.0118
	8	48	-5.86498	-44.11184	-0.0803
7	1	49	-12.5000	0	0
	2	50	0	0	0
	3	51	0.0518	0.0054	0
	4	52	0	0	0
	5	53	0	0	0
	6	54	-5	0	0
	7	55	-27.5	-2.5	0
	8	56	-27.5	-2.5	0

Tabla D.10 Tabla Q(S,A) media con estrategia de ε-greedy 0.15 y actualización por el número de visita (parte 2).

Nd	na	Estado	Acción		
			Avance	Giro horario	Giro antihorario
1	1	1	-0.1324	-0.0067	-0.0102
	2	2	-0.5436	-0.0594	-0.0600
	3	3	-0.5581	-0.0740	-0.0514
	4	4	-0.2238	-0.1016	-0.0452
	5	5	0	0	0
	6	6	0	0	0
	7	7	2.0117	0	0
	8	8	3.4297	1.41043	8.9521
2	1	9	0.2468	0.0103	0.0390
	2	10	4.7395e-05	0.0041	0.0007
	3	11	-6.7033e-05	-3.8658e-05	-3.0020e-06
	4	12	-0.0220	-0.0012	-0.0095
	5	13	0	0	0
	6	14	0	0	0
	7	15	1.2995	4.6058	1.5545
	8	16	0.0097	0.0017	0.0177
3	1	17	27.2781	1.8638	1.4311
	2	18	0.0122	1.8411	0.0017
	3	19	6.3717e-10	0.0002	-7.3892-07
	4	20	-0.0040	-0.0022	-0.0051
	5	21	0	0	0
	6	22	3.8870e-16	0	0
	7	23	5.8742e-07	2.3152e-07	0.0016
	8	24	0.0162	0.0025	2.0424
4	1	25	0.3413	0.0292	0.0085
	2	26	0.0121	0.1493	8.3984e-05
	3	27	2.0100e-08	0.0002	3.6680e-12
	4	28	-0.0070	-0.0024	-0.0005
	5	29	0	0	0
	6	30	2.4710e-12	-0.0007	4.2356e-12
	7	31	2.0889e-06	-4.8497e-07	0.0005
	8	32	-0.1324	-0.00670	-0.0102

Tabla D.11. Tabla Q(S,A) media con estrategia de ε-greedy 0.15 y actualización por la iteracion actual (parte 1).

Nd	Na	Estado	Acción		
			Avance	Giro horario	Giro antihorario
5	1	33	0.0004	6.1159e-06	1.4529e-05
	2	34	0.0002	0.0008	6.2280
	3	35	4.8486	1.5310e-05	1.0094e-09
	4	36	-0.0026	-8.3182e-10	-0.0031
	5	37	0	0	0
	6	38	-1.6293e-07	-0.0855	1.3263e-08
	7	39	-0.0001	-0.0320	1.7990e-05
	8	40	-0.0001	-0.0001	0.0003
6	1	41	-0.0040	-0.0180	-0.02141
	2	42	-2.3360e-06	1.4651e-09	-0.0195
	3	43	1.3311e-10	5.1095e-08	-0.0144
	4	44	0	8.4593e-13	-0.0005
	5	45	0	0	0
	6	46	-0.1839	-0.0306	4.9907e-09
	7	47	-0.8501	-0.3881	-2.4710e-05
	8	48	-0.2914	-0.30818	-2.1946e-05
7	1	49	-0.0382	-7.3082e-08	-0.0005
	2	50	0	0	0
	3	51	0	2.4734e-15	0
	4	52	0	0	0
	5	53	0	0	0
	6	54	0	0	0
	7	55	-0.0596	0	0
	8	56	-0.0976	-0.0014	1.19397e-15

Tabla D.12. Tabla Q(S,A) media con estrategia de ε-greedy 0.15 y actualización por la iteracion actual (parte 2).

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA