Robots capaces de aprender y adaptarse al entorno a partir de sus propias experiencias

Pablo Quintía Vidal

DEPARTAMENTO DE ELECTRÓNICA E COMPUTACIÓN

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Departamento de Electrónica e Computación

Tesis doctoral

ROBOTS CAPACES DE APRENDER Y ADAPTARSE AL ENTORNO A PARTIR DE SUS PROPIAS EXPERIENCIAS

Presentada por:

Pablo Quintía Vidal

Dirigida por:

Roberto Iglesias Rodríguez

Carlos Vázquez Regueiro

Mayo 2013

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Roberto Iglesias Rodríguez, Profesor Titular de Universidad del Área de Ciencia de la Computación e Inteligencia Artificial de la Universidade de Santiago de Compostela e investigador adscrito al Centro Singular de Investigación en Tecnoloxías da Información (CITIUS)

Carlos Vázquez Regueiro, Profesor Titular de Universidad del Área de Arquitectura y Tecnología de Computadores de la Universidade da Coruña

HACEN CONSTAR:

Que la memoria titulada **ROBOTS CAPACES DE APRENDER Y ADAPTARSE AL ENTORNO A PARTIR DE SUS PROPIAS EXPERIENCIAS** ha sido realizada por **D. Pablo Quintía Vidal** bajo nuestra dirección en el Departamento de Electrónica e Computación de la Universidade de Santiago de Compostela, y constituye la Tesis que presenta para optar al título de Doctor.

Mayo 2013

Roberto Iglesias Rodríguez	**Carlos Vázquez Regueiro**	**Pablo Quintía Vidal**
Director de la tesis	Director de la tesis	Autor de la tesis

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Agradecimentos

Esta tesis es el resultado del trabajo de muchas personas, a las cuales quiero agradecer su ayuda y apoyo durante todo este tiempo:

En primer lugar a mis directores de tesis, Roberto Iglesias Rodríguez y Carlos Vázquez Regueiro, por su ayuda, sus consejos y la confianza que depositaron en mí.

A Miguel Rodríguez por su inestimable ayuda y al que considero como un codirector más de la tesis. También a Theocharis Kyriacou por la colaboración prestada y a Eva Cernadas por sus consejos.

Al Departamento de Electrónica e Computación y al CITIUS de la Universidade de Santiago de Compostela, por proporcionar los recursos necesarios para la realización de esta tesis.

Al Departamento de Electrónica e Sistemas de la Universidade da Coruña, por darme la oportunidad de iniciarme como docente y por todo el apoyo y ayuda que me ha proporcionado.

A todos los precarios de Coruña y Santiago con los que compartí las incertidumbres de un estudiante de doctorado y gracias a los que los días en el laboratorio eran más agradables.

A la Xunta de Galicia, por proporcionar mediante su programa de becas predoctorales la financiación con la que pude dedicarme a tiempo completo a este trabajo.

En el último y más importante lugar, gracias a mis padres, familia y amigos, por hacerme olvidar todo en los ratos libres (excepto cuando preguntaban ¿pero aún no acabaste?). Especialmente a Carolina, por apoyarme todo este tiempo y el que queda.

Mayo 2013

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Índice general

Introducción

Objetivo de la tesis

Los intentos de dotar a los robots móviles con conocimiento y la capacidad de realizar tareas de forma autónoma se vienen haciendo desde prácticamente el principio de la robótica [35]. Inicialmente se asumía que se poseía suficiente información sobre el robot, la tarea y el entorno, pero dicha asunción se demostró falsa ya que a la hora de desarrollar un *software* de control aparecen diversas limitaciones al conocimiento que poseemos. El comportamiento de un robot es en general impredecible, no hay nada que garantice que un robot que funciona perfectamente en el entorno en el que ha sido diseñado o probado vaya a ejecutar la tarea de manera robusta y fiable en cualquier tipo de entorno. Esto se debe a que el comportamiento de un robot está afectado por la combinación de tres factores: el propio robot – el *hardware* –, el controlador – el *software* – y el entorno. Por lo tanto un mismo robot con un mismo programa de control puede que se comporte de manera diferente en entornos diferentes. Aunque pueda parecer factible tener un modelo exacto del robot, es posible que la forma en la que se mueve pueda verse alterada a pesar de recibir las mismas consignas – debido a cambios en la carga de las baterías, la presión de las ruedas, posibles averías, etc. – por lo que el modelo en ningún caso podrá ser perfecto. Por otro lado, para ambientes cotidianos será inviable obtener un modelo del entorno, ya que éste será altamente dinámico, con frecuentes cambios en la posición de obstáculos u objetos relevantes. También, debido a las diferentes propiedades de los materiales, las lecturas sensoriales pueden verse afectadas, con el consecuente cambio en el comportamiento del robot.

Si bien ya a principios de los años 80 se vio la necesidad de obtener robots con la capacidad de aprender a partir de sus propias experiencias [115, 16, 118, 62], esta capacidad en los robots es ahora más necesaria que nunca, ya que actualmente se observa un claro desplaza-

miento hacia el sector de servicios. Tal y como se recoge en el Libro Blanco de la Robótica [30], se espera que la robótica de servicios se introduzca de forma masiva en la sociedad en la próxima década. Se pretende que los robots formen parte de la vida diaria como electrodomésticos, asistentes del hogar, ayudantes y cuidadores de personas dependientes, compañeros de trabajo, etc. Sin embargo, para que los robots operen fuera de los centros de investigación y sin la supervisión de ingenieros o expertos en robótica, es necesario afrontar ciertos retos tecnológicos, entre los cuales está la autonomía avanzada: los robots deben poder actuar en entornos muy diferentes y dinámicos, realizando una amplia variedad de tareas y operando de diferente manera en función de la situación. Una de las principales características de estos robots será la capacidad de adaptación a los cambios.

Existen numerosas evidencias que avalan la importancia de alcanzar una verdadera capacidad adaptativa en los robots actuales. A modo de ejemplo podemos mencionar aquí la agenda estratégica de investigación de la *European Robotics Technology Platform*, EUROP [38], donde como reto específico aparece la necesidad de robots con la capacidad de aprendizaje continuo. La agenda elaborada por EUROP muestra que será necesario que los futuros robots adapten tanto su *hardware* como su *software* para poder desenvolver con éxito las tareas que se espera que realicen. Por otra parte, también nos parece interesante mencionar la propuesta RoboCom (*Robot Companions for Citizens* [31]), desarrollada por un consorcio de numerosos grupos de investigación europeos, y que fue una de las tres propuestas finalistas a ser uno de los *Flagship* FET (*Future and Emerging Technologies*) de la Unión Europea (resolución definitiva del año 2013). Esta propuesta se centra en la idea de que para mantener el bienestar de la sociedad europea en el futuro será necesario disponer de una nueva generación de robots. Estos robots serán un motor tanto económico como social, y desarrollarán tareas como la manufactura de productos, lo que evitaría la deslocalización de la producción industrial a países con mano de obra extremadamente barata; o el cuidado de ancianos y dependientes, ya que se estima que dentro de 50 años más de un tercio de la población europea será mayor de 60 años. En el propio documento que resume las líneas generales de esta propuesta se describe cómo uno de los conceptos ambiciosos y claves para posibilitar el logro de este tipo de robots es la adaptabilidad. En este sentido se afirma que todavía es necesario alcanzar un mayor grado de adaptación en los robots.

Los documentos anteriores dejan ver que la principal dificultad para la introducción en la sociedad de la robótica de servicios son las limitaciones tecnológicas de los robots actuales, y no a la ausencia de posibles compradores. Un ejemplo de esto es el éxito de los robots aspira-

dores. Este tipo de robots poseen una "inteligencia" y capacidad sensorial muy limitadas, lo cual se suple en muchas ocasiones con la presencia de una persona o la modificación del entorno. Esto nos deja ver que aunque el aprendizaje se puede aplicar a cualquier tipo de robot, en el contexto de la robótica de servicios la capacidad de adaptación automática se hace más patente, debido a los entornos en los que operan y a los usuarios que los utilizan. Si se desea que los robots de servicio puedan realizar tareas más complejas y de manera más automática es necesario que tengan la capacidad de aprender por sí mismos.

En este contexto, todas aquellas estrategias que permiten al robot aprender a partir de su interacción con el entorno adquieren especial relevancia. De entre esas técnicas queremos destacar el aprendizaje por refuerzo, donde todo lo que se necesita para que el robot alcance un comportamiento interesante es una respuesta – función de refuerzo – que indica al robot si sus acciones son correctas o no. Ahora bien, pese a que este paradigma parece adecuado para conseguir la adaptación de los robots a su entorno, no ha tenido el impacto esperado en sistemas reales. Esto se debe principalmente a dos razones: la lentitud del proceso de aprendizaje, y el elevado número de errores que comete el robot durante el mismo. Esto hace que en muchos casos el aprendizaje se traslade a un simulador, sin embargo, no siempre habrá un simulador donde poder aprender el controlador, y en caso de haberlo, éste puede ser incompleto o limitado. Por esta razón necesitamos algoritmos que permitan que el robot se adapte en intervalos de tiempo más cortos y con el mínimo número de fallos. El robot debe tener la capacidad de adaptarse y aprender de manera rápida tanto en aprendizajes partiendo de cero, sin ningún conocimiento previo, como para el refinamiento de controladores donde es necesario adaptarlos a alguna nueva situación.

Otra característica necesaria es que el proceso de adaptación de los robots al entorno en el que se mueven sea lo más sencillo posible, ya que lo más probable es que el usuario final de un robot de servicio no sea un experto en robótica. Por este motivo se requieren algoritmos con un reducido número de parámetros cuyo valor sea sencillo de establecer, esto hará que los algoritmos de aprendizaje sean más fáciles de usar y más polivalentes.

En estos dos últimos párrafos se condensan los objetivos de la tesis, esto es, el desarrollo de un sistema de aprendizaje basado en refuerzo que permita una rápida y sencilla adaptación de los robots cometiendo el mínimo número de errores. En el pasado se han realizado esfuerzos dentro del Grupo de Sistemas Inteligentes de la Universidade de Santiago de Compostela para acelerar el proceso de aprendizaje, utilizando información sobre la tarea [81, 82] o combinando el aprendizaje por refuerzo con la programación genética [52, 109]. Estas estra-

tegias requerían del ajuste de numerosos parámetros y la velocidad de aprendizaje conseguida con ellas no permitía su uso en robots reales. En esta tesis se presenta una nueva estrategia donde no será necesario disponer de información previa sobre la tarea y donde el número de parámetros a ajustar se reduce al mínimo. Ahora bien, con esta tesis también se pretenden dar los primeros pasos para lograr el aprendizaje continuo. Queremos romper la propuesta clásica de dos etapas claramente diferenciadas: una primera etapa de aprendizaje a partir de la interacción robot-entorno, y una segunda etapa donde el robot pone en uso lo aprendido anteriormente. No se pretende obtener un controlador y aplicarlo para un robot y un entorno sin tocarlo jamás, sino que por contra, buscamos flexibilidad y pretendemos que los futuros robots dispongan de controladores que siempre estén sujetos a posibles cambios, adaptándose a las diferentes situaciones que pueda encontrar el robot en cualquier momento.

Estructura de la tesis

En esta tesis vamos a describir el trabajo realizado para conseguir los objetivos mencionados anteriormente. En el **capítulo 1** se contextualiza la investigación, introduciendo conceptos generales sobre la robótica y el por qué es necesario el aprendizaje en robots. Para finalizar este capítulo se mostrarán distintas estrategias de aprendizaje de uso habitual en robótica de servicios, en concreto el aprendizaje por demostración y el aprendizaje a partir de la experiencia.

El **capítulo 2** aborda en detalle el aprendizaje por refuerzo, sus fundamentos matemáticos y la importancia de las funciones de valor. Se hace una reflexión de la revisión bibliográfica, donde no solo se muestran los algoritmos clásicos que aportan las bases a partir de las cuales se debe hacer cualquier propuesta, sino que también se mostrarán algunos algoritmos de aprendizaje por refuerzo más novedosos.

En el **capítulo 3** presentamos nuestra propuesta para hacer frente al primer reto que nos marcamos: aumentar la interpretabilidad de los algoritmos de aprendizaje por refuerzo. Al utilizar un algoritmo debemos ser capaces de interpretar de manera rápida e intuitiva si el aprendizaje progresa de la manera esperada o si por el contrario hay algún problema. La mayor parte de los algoritmos son muy poco transparentes, en ellos normalmente se acaba con grandes matrices dispersas que no permiten saber si el aprendizaje está funcionando bien o mal. Para resolverlo presentamos una nueva formulación de la función objetivo, la cual se basa en predecir el tiempo que transcurrirá hasta que el robot cometa un error. A partir de esta

nueva función objetivo construimos el algoritmo de aprendizaje *Increasing the time before failure* (*I_tbf*). Este nuevo algoritmo es la base sobre la que se sustentan una buena parte de las soluciones aportadas en esta tesis.

El **capítulo 4** se centra en el problema de la representación de estados o situaciones representativas por las que pasa el robot. Una característica de los algoritmos de aprendizaje por refuerzo en los que se apoya esta tesis es la necesidad de clasificar el entorno en un conjunto finito de estados a partir de la información sensorial que recibe el robot. Este conjunto debe ser lo suficientemente grande como para representar inequívocamente las diferentes situaciones que encuentra el robot, pero no puede ser excesivamente grande ya que eso incrementaría el tiempo de aprendizaje de manera exponencial. También, si como se comentó en los objetivos de la tesis, se desea que el robot se adapte a nuevas circunstancias, la representación de estados no puede ser estática o prediseñada *ad hoc*. El robot debe ser capaz de incorporar nuevo conocimiento a su representación de estados sin que ello suponga olvidar lo ya aprendido. Para lograr que el robot sea capaz de encontrar las situaciones relevantes durante el aprendizaje decidimos usar una red *Fuzzy ART* [26]. Este tipo de redes son capaces de generar de forma dinámica y no supervisada una clasificación de estados, además tienen la característica de ser robustas ante situaciones poco frecuentes. En esta tesis hemos adaptado la red *Fuzzy ART* para su uso en nuestro sistema de aprendizaje y mostramos cómo es posible generar un espacio de estados a medida que avanza el aprendizaje.

Llegados a este punto disponemos de un sistema capaz de aprender a "ver y hacer". Si como se expuso en los objetivos de la tesis, se desea aprender a partir del mínimo número de ejemplos, se nos presenta un dilema, común a todos los procesos de aprendizaje y clasificación, que es el equilibrio entre *bias* y *varianza*. Esto es, llegar a un equilibrio entre cómo de bien debe el algoritmo de aprendizaje ajustarse a los datos procedentes de la interacción robot-entorno, y al mismo tiempo mantener la capacidad de generalizar correctamente ante nuevas situaciones. El **capítulo 5** muestra una propuesta basada en comités de aprendedores para resolver el dilema *bias-varianza*. Esta estrategia se basa en un conjunto de aprendedores donde cada uno de ellos aprende de manera independiente y los errores que cometen no están correlacionados. Con esta propuesta alcanzamos una solución que nos permite trasladar los procesos de aprendizaje a un robot operando en el entorno real, lo cual era uno de los objetivos de la tesis. A su vez, tal y como se reflejará en este capítulo 5, el uso de estos comités abre nuevas vías interesantes de cara al logro de procesos de aprendizaje continuos.

Otro problema que enlaza con la representación del entorno es el gran volumen de información sensorial disponible. Una parte crítica de cualquier sistema autónomo es la percepción sensorial. Al final del capítulo 4 alcanzábamos un sistema capaz de aprender a "ver y hacer". Los comités del capítulo 5 nos han permitido acelerar estos procesos de aprendizaje, ahora bien, la existencia de estos comités también facilitará que la percepción del entorno se pueda condicionar aún más a la tarea que el robot está tratando de resolver. Por esto en el **capítulo 6** se mostrará cómo analizar las señales sensoriales para extraer la información relevante e identificar los sensores más importantes para la tarea que el robot está realizando. Cada vez los robots vienen equipados con más sensores que proporcionan más información y con mayor frecuencia de actualización. Este volumen creciente de información hace que los algoritmos de aprendizaje tarden más en converger si la información no es relevante para lo que el robot quiere hacer en cada momento. Para evitarlo aplicamos a los datos sensoriales técnicas de la teoría de la información con el fin de detectar aquellos sensores más relevantes para la tarea. De esta manera el algoritmo de aprendizaje dispondrá de una mayor calidad en la información sensorial a medida que avance el proceso de aprendizaje.

Por último, se cierra la memoria con el **capítulo 7**, donde se realiza una reflexión sobre las principales conclusiones que se derivan del trabajo descrito esta tesis, y se proponen nuevas vías de avance.

Capítulo 1

Contexto y motivación

Las historias de autómatas humanoides son recurrentes en la mitología de varias civilizaciones (China, Grecia, el imperio islámico). La palabra robot forma parte de la fantasía popular desde que el escritor checo Karel Čapek la introdujera en su obra *R.U.R.* (*Robots Universales Rossum*) en 1920, para referirse a la gente artificial creada para trabajar en el lugar de humanos. Desde entonces los robots son un elemento recurrente en las historias de ciencia ficción, donde se muestran robots tanto o más inteligentes y capaces que los humanos e incluso robots con sentimientos. La realidad es que la robótica todavía está lejos de alcanzar lo que se muestra en la ficción.

En este capítulo se muestra una introducción a los diferentes tipos de robots existentes, con especial detalle en los robots de servicio. A continuación se introduce el concepto de autonomía y su necesidad en la robótica de servicio personal. Por último se presentan dos estrategias de aprendizaje de gran utilidad: el aprendizaje por demostración y el aprendizaje a partir de la interacción con el entorno.

1.1. Tipos de robots

Según la *Robotics Industry Association* un robot es un manipulador reprogramable y multifuncional diseñado para mover materiales o dispositivos con movimientos programados y con el objetivo de realizar una tarea. Esta puede ser una buena definición para los robots industriales aunque es muy restrictiva para lo que hoy se conoce como robots de servicio. Arkin [7] propone una definición en la que dice que un robot es una máquina capaz de extraer in-

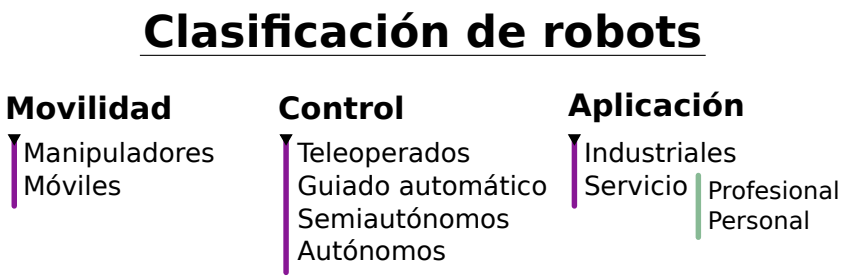


Figura 1.1: Clasificación de los robots en función de su movilidad, el tipo de control y su aplicación.

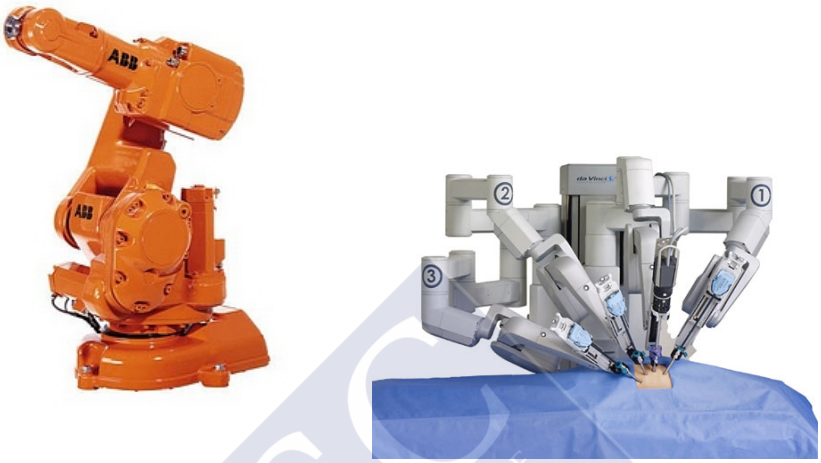
formación del entorno y utilizar el conocimiento que posee sobre el mundo para moverse de forma segura y con un propósito.

Existen muchos tipos de robots capaces de realizar diferentes tareas. En la figura 1.1 se muestra un esquema general de diferentes clasificaciones de los robots en función de varios criterios: su movilidad, su sistema de control, y por último su aplicación.

En función de su movilidad podemos clasificar a los robots como robots manipuladores o robots móviles. Los robots manipuladores (figura 1.2), son aquellos que realizan, en general, tareas repetitivas y de gran precisión. Suelen ser brazos articulados con un efector u herramienta en su extremo, están anclados en una posición y operan en un entorno muy delimitado y estructurado. Su uso más habitual es en cadenas de ensamblaje para la realización de tareas como soldadura, pintado, perforación o desplazamiento de objetos. En la actualidad se están introduciendo robots manipuladores teleoperados en quirófanos para la realización de ciertas intervenciones.

Los robots móviles (figura 1.3) se caracterizan por la capacidad de desplazarse por el entorno, sea éste tierra, aire o agua. Existen robots móviles que se desplazan por entornos diseñados para facilitar su movimiento, esto sucede principalmente en ámbitos industriales. Pero el lograr que un robot se mueva de manera segura y autónoma por un entorno cotidiano, no estructurado y que pueda estar densamente poblado, es aún un reto. Esta memoria pretende ser un avance en esta dirección.

Antes de exponer la clasificación de robots en función de su autonomía, conviene proporcionar una definición de dicho término. Según Webster [79] existen dos definiciones de autonomía o, lo que es lo mismo, se pueden establecer dos grandes categorías: actuación sin control externo y capacidad de auto-gobierno.



(a) (b)

Figura 1.2: Dos ejemplos de robots manipuladores: a) brazo articulado de uso en instalaciones industriales, y b) robot quirúrgico.

En el primer caso, cualquier sistema que transporte su propia fuente de energía y su propio sistema de control se dice que es autónomo, ya que no depende del exterior, es decir, es autosuficiente. Sin embargo, este tipo de autonomía se considera "débil", ya que cualquier alteración del entorno o situación no especificada generalmente conlleva un funcionamiento incorrecto del sistema.

La segunda definición implica un mayor grado de autonomía que podemos denominar "autonomía fuerte". En este nivel un sistema es autónomo cuando tiene capacidad para decidir, a partir de sus propios procesos de razonamiento, las acciones que ha de tomar en vez de seguir y ejecutar una secuencia fija e inmutable de pasos o instrucciones. La autonomía fuerte requiere la habilidad para construir representaciones internas del mundo, para planear, y para aprender a partir de la experiencia [84].

La característica más importante de un sistema autónomo es su capacidad para operar durante largos periodos de tiempo en un entorno "natural" (es decir, sin alteraciones que faciliten su funcionamiento) y sin ningún tipo de intervención humana directa. Esta cualidad conlleva, inevitablemente, la adaptación a los cambios que se produzcan en el entorno, una actuación

aceptable ante situaciones no previstas, el aprendizaje y modificación del comportamiento a partir de la propia experiencia, y la construcción de representaciones internas del mundo para utilizarlas en los procesos internos de razonamiento, como por ejemplo la navegación.

Según Pfeifer y Scheier [91] la autonomía completa no existe, ya que todos los sistemas dependen, de uno u otro modo, de su entorno. La autonomía es, por tanto, una cuestión de grado.

Una palabra asociada a los robots y a su autonomía es la inteligencia. El término inteligencia es ampliamente utilizado en la bibliografía, aunque hasta ahora se ha resistido a una definición clara, útil y ampliamente aceptada por toda la comunidad científica. Por este motivo, establecer si un sistema se comporta de forma inteligente no deja de ser, hasta cierto punto, subjetivo y sujeto a la propia interpretación del observador y, por lo tanto, a su educación, entendimiento y punto de vista. Por todo ello, en este trabajo se evitará considerar si un sistema es o no inteligente, y nos centraremos más en su grado de autonomía.

Una vez explicado qué es la autonomía podemos establecer una nueva clasificación de robots en función de ella (figura 1.1):

- Robots teleoperados: son aquellos robots donde todo el control lo realiza un operador humano. Ejemplos de este tipo de robot son los usados para la desactivación de explosivos, o los vehículos aéreos no tripulados (UAV).

- Vehículos de guiado automático (AGV): los AGV son vehículos que se desplazan por entornos adaptados con raíles, balizas u otro tipo de marcadores que permitan que estos robots sigan rutas preestablecidas. Habitualmente este tipo de robots se encargan de tareas de transporte de mercancías en plantas industriales. Su autonomía es muy limitada ya que, aunque no necesitan ningún humano que los maneje, su uso está restringido a entornos controlados.

- Robots semiautónomos: los robots semiautónomos son robots teleoperados con un grado de autonomía débil. Generalmente los robots semiautónomos poseen controladores reactivos, capaces de realizar tareas sencillas de bajo nivel, lo que les otorga un grado de autonomía bajo. Un ejemplo son los *rovers* de exploración espacial usados en Marte. Debido al retardo de las comunicaciones es imposible controlar en tiempo real los robots desde la Tierra, por ello los controladores de misión mandan al robot las consignas relativas a la misión que debe realizar, y el robot las ejecuta gracias a su capacidad limitada de navegación y adquisición de datos autónomas.

(a) (b)

(c) (d)

Figura 1.3: Robots móviles con diversos grados de autonomía: a) UAV teleguiado Predator en uso por el ejército de EE.UU., b) AGV para la carga de mercancías, c) Mars Rover Spirit, robot semiautónomo de exploración marciana de la NASA, y d) aspirador autónomo Roomba de iRobot.

– Robots autónomos: en [12] se define a los sistemas autónomos como aquellos capaces de operar en el mundo real, sin ningún tipo de control externo, por largos períodos de tiempo. La frontera entre robots autónomos y semiautónomos es difusa en tanto que un mismo robot puede ser autónomo en unos aspectos y requerir ayuda humana para otros.

En esta memoria nos centraremos en este último tipo. La investigación con robots autónomos se dirige a la obtención de sistemas de control para la realización correcta de tareas y capaces de gestionar la incertidumbre asociada al entorno en el que el robot se encuentra.

1.2. Robótica de servicio

La última clasificación de las mostradas en la figura 1.1 es la clasificación atendiendo al uso del robot. Según su propósito se puede establecer una diferenciación entre robots industriales y robots de servicio. Éstos a su vez se dividen en robots de servicio personal y robots de servicio profesional.

La norma ISO 8373 define un robot industrial como: un manipulador controlado automáticamente, reprogramable y multipropósito, programable en tres o más ejes, que puede estar tanto anclado en un lugar como ser móvil, y de uso en aplicaciones de automatización industrial.

La definición de robot de servicio es más compleja y no hay un consenso sobre qué se puede definir como robot de servicio y qué no. Según la Federación Internacional de Robótica un robot de servicio es aquel que opera de manera autónoma o semiautónoma para realizar servicios útiles al bienestar de los humanos o su equipamiento, excluyendo las operaciones de fabricación. Actualmente una comisión de expertos está trabajando para actualizar la norma ISO 8373, dicha actualización incluirá una definición de robots de servicio.

Dentro de los robots de servicio se puede diferenciar entre robots de servicio de uso profesional o aquellos de uso personal. Nos centraremos en estos últimos. Los robots de servicio de uso personal son aquellos con capacidades de convivir con las personas y de realizar tareas que influyan directamente en su forma de vida. Éstos son robots domésticos, de vigilancia, de entretenimiento, guías en bibliotecas y museos, etc. (figura 1.4). Los robots de servicio de uso profesional son aquellos usados en aplicaciones profesionales, como puede ser agricultura, transporte, uso militar, etc. (figura 1.5).

Al contrario que la robótica industrial, que ya está bien establecida en todas las fases de la manufactura de productos, la robótica de servicio es todavía emergente (figura 1.6). Según las previsiones tanto de la Federación Internacional de Robótica como del gobierno japonés la robótica de servicio crecerá de manera exponencial en los próximos años, tanto a corto plazo (figura 1.7) como a medio-largo plazo (figura 1.6). Según el Libro Blanco de la Robótica en España [30] nos encontramos al inicio de una revolución que nos llevará a un mercado robótico de consumo, mediante la utilización en el hogar de nuevos robots, auténticos electrodomésticos móviles, que complementarán a los actuales estáticos. Es por esto que toda la investigación dedicada a la robótica, y especialmente a la robótica de servicio, será crucial para cumplir estas previsiones y tendrá un gran impacto social y económico.

(a) (b)

(c) (d)

Figura 1.4: Robots de servicio personal: a) Nao, robot humanoide utilizado en la RoboCup, b) robot limpia piscinas Dolphin, c) Paro, una foca robótica usada en Japón para la compañía de ancianos, y d) Care-O-bot un robot asistente del hogar.

1.3. Autonomía en los robots de servicio personal

Los entornos donde operan los robots de servicio personal son entornos cotidianos, donde la gente vive y trabaja, y sus usuarios no serán expertos en robótica. Esto hace necesario que estos robots sean lo más autónomos que sea posible, descargando al usuario final de tareas como la configuración del robot o el entorno, y la programación de las tareas a realizar.

Las principales características que se le piden a un robot de servicio personal son las siguientes:

(a) (b)

(c)

Figura 1.5: Robots de servicio profesional: a) robot de seguridad, b) robot portacontenedores, y c) robot fumigador.

– Puede funcionar de forma independiente durante un tiempo significativo.

– Toma sus propias decisiones, sin necesidad de control humano.

– Posee sus propias fuentes de energía y computación: es autosuficiente.

– Puede percibir y actuar en el mundo real que le rodea.

– Se adapta a los cambios y retos presentes en un entorno dinámico.

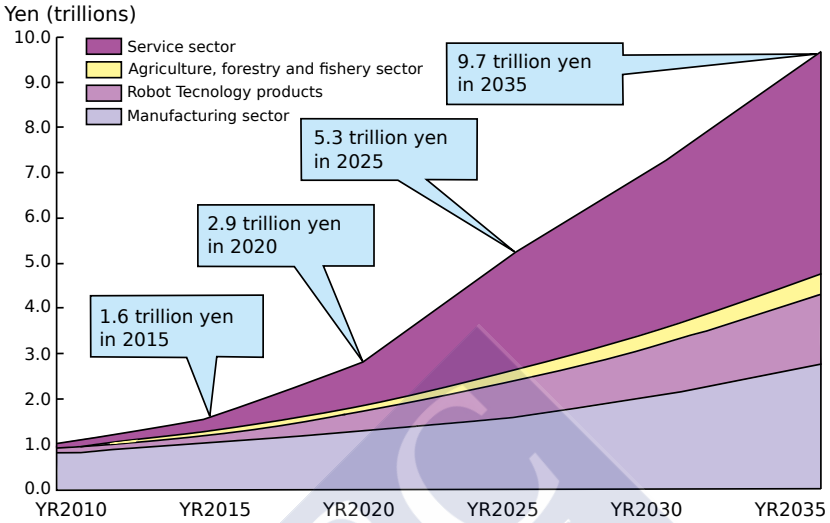


Figura 1.6: Estimación del gobierno japonés de la evolución del mercado de robots hasta el año 2035. Fuente: comunicado de prensa del Ministerio de Economía, Comercio e Industria del Gobierno Japonés, 23 de abril de 2010.

Para lograr las características anteriores debemos tener presente los tres elementos que afectan al comportamiento de un robot: la configuración física del propio robot (sensores, actuadores, forma, materiales, etc.), el programa de control, y por último el entorno en el que se desarrolla la tarea (figura 1.9). Dentro del entorno estará el usuario final del robot.

Los componentes físicos de un robot se pueden estructurar en dos grupos: sensores y actuadores. Los sensores constituyen el sistema perceptivo del robot y se encargan de medir una serie de magnitudes físicas como son: contacto (sensores táctiles), distancia (ultrasonidos, infrarrojos, láser), luminosidad (cámaras), posición (GPS), etc. La información proporcionada por los sensores debe ser procesada en mayor o menor medida dependiendo de su complejidad. Finalmente, los actuadores son cualquier dispositivo que permite al robot actuar sobre el entorno, y pueden ser desde medios de locomoción como patas o ruedas hasta brazos o dedos.

El programa de control de un robot tiene como misión procesar los datos proporcionados por los sensores y seleccionar en cada instante la acción más adecuada para su envío a los actuadores. El objetivo del controlador será que estos actuadores produzcan el resultado deseado sobre el entorno teniendo en cuenta la tarea que el robot está llevando a cabo.

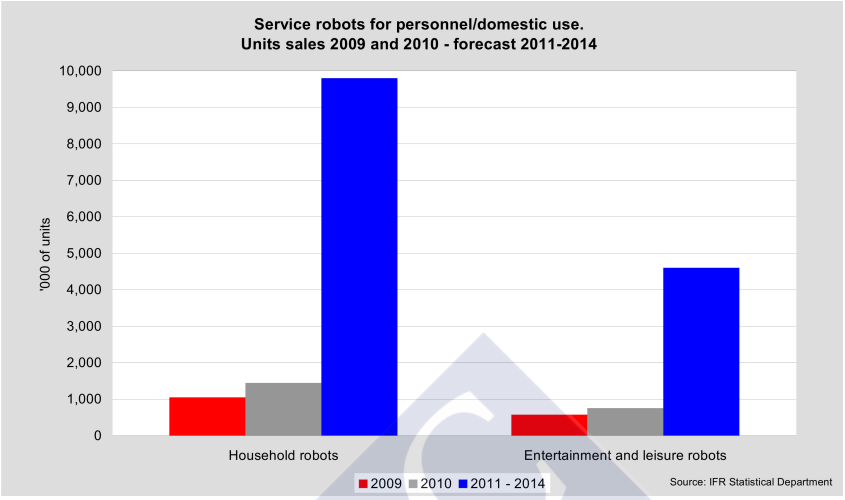


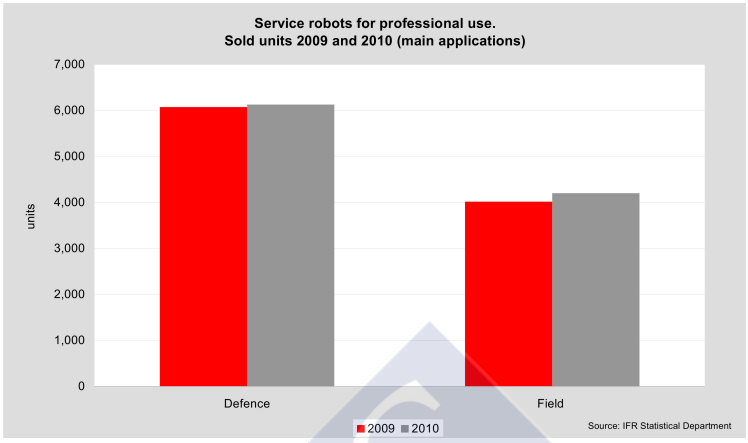
Figura 1.7: Ventas de robots de servicio para uso personal y previsión de ventas para las próximos años. Fuente: informe "World Robotics 2011" del Departamento de Estadística de la Federación Internacional de Robótica.

Tabla 1.1: Principales diferencias entre los controladores reactivos y los deliberativos.

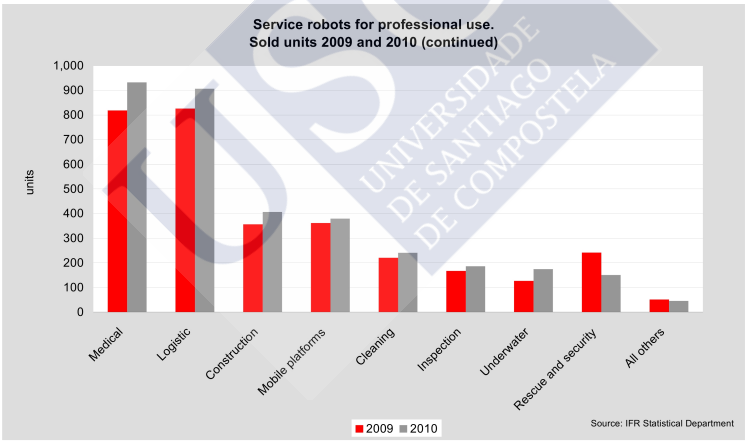
	Deliberativo	**Reactivo**
Tiempo de respuesta	Lento	Tiempo real
Conocimiento sobre el mundo	Modelos complejos del mundo	Modelos básicos
Estructura	Jerárquica	Lineal
Representación del modelo del mundo	Simbólica	Bajo nivel
Mecanismo de razonamiento	Contiene capas de razonamiento entre percepción y acción	Alto acoplamiento entre percepción y acción
Tipo de inteligencia	Alto nivel	Bajo nivel
Qué es posible aprender	Tareas complejas	Tareas muy simples

Las funciones de control de un robot se han clasificado históricamente en dos grupos, según el tipo de información que utilicen para decidir las acciones a ejecutar: control reactivo o deliberativo. En la tabla 1.1 se puede ver una comparación entre el control reactivo y el control deliberativo.

De acuerdo a lo que se describe en Handbook of Robotics [117], el control reactivo se basa en la noción biológica de estímulo-respuesta. Este tipo de control utiliza únicamente la información sensorial recibida en el instante actual a la hora de decidir la acción, y no usa



(a)



(b)

Figura 1.8: Ventas de robots de servicio de uso profesional: a) ventas de robots militares, y b) ventas de robots de servicio para uso profesional exceptuando aplicaciones militares. Fuente: informe "World Robotics 2011" del Departamento de Estadística de la Federación Internacional de Robótica.

ninguna representación interna del entorno. Tiene en general un tiempo de reacción muy bajo con el fin de que la reacción sea lo más rápida posible. Es por tanto un control de bajo nivel adecuado para entornos dinámicos y no estructurados.

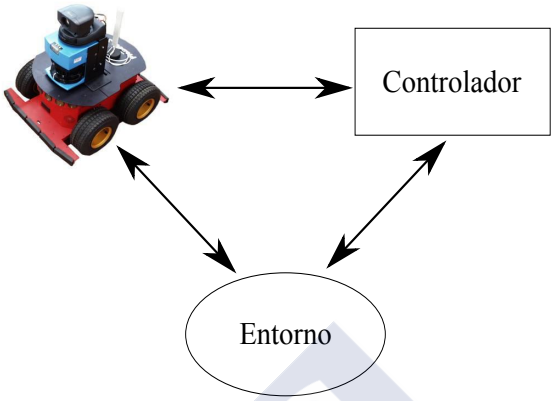


Figura 1.9: La autonomía de un robot depende de la configuración física del robot, su programa de control y el entorno en el que opera.

En el mismo estudio [117] se considera al control deliberativo más complejo que el reactivo. En el control deliberativo generalmente se utiliza toda la información sensorial disponible, además de conocimiento almacenado internamente, para razonar qué acciones ejecutar a continuación. Este razonamiento es generalmente el planificar una serie de acciones y los resultados de las mismas, con el fin de obtener una secuencia de acciones con la que se alcance el resultado deseado. La planificación requiere la existencia de un modelo interno del mundo, y para ser eficiente dicho modelo debe ser preciso y actualizado. Debido a esto los autores opinan que a día de hoy no existen robots puramente deliberativos.

Para resolver tareas complejas normalmente se recurre a soluciones híbridas que pretenden combinar las mejores características del control reactivo y deliberativo: la respuesta en tiempo real del control reactivo y el razonamiento del deliberativo. Los controladores híbridos utilizan una arquitectura de tres capas, la cual consiste en una capa reactiva (ejecución), capa intermedia (coordinación), y deliberativa (organización/planificación) [105, 117]. Para este tipo de tareas de alta complejidad suele ser necesario contar con una arquitectura de control, con el fin de determinar y coordinar las tareas que un robot debe llevar a cabo. Una arquitectura de control proporciona un conjunto de principios para organizar el *software* de control, indicando la estructura e imponiendo una serie de restricciones en la forma en que el problema de control puede ser solucionado [78].

A la hora de programar un controlador de los mencionados anteriormente, el planteamiento clásico de la robótica asumía que las tareas se podrían resolver de forma analítica. Teniendo un modelo del robot, el entorno y la tarea se puede programar al robot para que resuelva la tarea, pero en la realidad esta aproximación es irrealizable. Para que un robot sea capaz de realizar una tarea útil debe poseer un mapeo entre sus percepciones (el estado del entorno) y las acciones que realizará, es lo que se denomina una política de control. Programar una política de control para que un robot realice tareas en entornos no estructurados es complicado, costoso y propenso a errores. Esto es debido a que no se puede tener un modelo completo, fiable y duradero del robot y/o el entorno. Dicha afirmación cobra especial relevancia cuando el objetivo es que el robot opere entre humanos y en entornos no estructurados ni adaptables. Siguiendo a Thrun [129], clasificamos las limitaciones al diseño manual de controladores como diferentes cuellos de botella:

– *Cuello de botella del conocimiento:* un diseñador humano debe proporcionar modelos precisos del mundo y el robot.

– *Cuello de botella de la ingeniería:* incluso si se dispone de información detallada sobre el mundo y el robot, hacer esta información accesible al sistema puede requerir una cantidad inasumible de tiempo de programación.

– *Cuello de botella de la tratabilidad:* pronto se reconoció que muchos dominios robóticos realistas son demasiado complejos para ser manejados de manera eficiente. La tratabilidad computacional mostró ser un severo obstáculo para el diseño de estructuras de control para robots complejos en entornos complejos.

– *Cuello de botella de la precisión:* el dispositivo robótico debe ser lo suficientemente exacto como para ejecutar de manera precisa los planes que fueron generados utilizando los modelos internos del mundo.

– *Cuello de botella de la percepción:* los sensores del robot a menudo tienen una información parcial e incompleta del entorno que le rodea.

El desarrollo clásico de controladores implica procesos cíclicos de diseño y prueba del controlador. Esto es lento, costoso y propenso a fallos, además requiere que lo realice un experto en robótica. En un entorno de robótica de servicio personal es necesario que el robot se adapte por sí mismo. Al igual que las personas, este tipo de robots deben ser capaces de

Tabla 1.2: Características básicas de la programación de robots, el aprendizaje por experiencia y el aprendizaje por demostración.

Aprendizaje	**Diseño *ad hoc***	**Experiencia**	**Demostración**
Cómo	Programando	Exploración	Ejemplos
Qué	Sin aprendizaje	Una política de control	Una política de control
De dónde	El diseñador	Prueba y error	El profesor
Cuándo	Durante el diseño	Durante y después del diseño	Durante y después del diseño
Puntos críticos	Se requiere mucho conocimiento experto. Diseño mediante prueba y error	La función de refuerzo requiere conocimiento experto	Rendimiento del profesor y el mapeo entre los ejemplos y el robot

aprender de sus propios errores o de las indicaciones que les proporciona un humano. No se va a conseguir un avance definitivo de la robótica de servicio personal hasta que los robots dispongan de la capacidad de adaptarse. La adaptación es lo que hace que el comportamiento del robot se pueda adecuar a los cambios durante el funcionamiento del mismo. La adaptación puede tener lugar en diferentes escalas de tiempo (corto y largo plazo), y puede afectar a cualquier nivel del sistema. Esta necesidad de adaptación es uno de los requerimientos que instituciones como la *European Robotics Technology Platform* (EUROP), reconocen como claves para lograr el éxito y la difusión de la robótica de servicio [38].

1.4. Estrategias de aprendizaje

Una vez vista la necesidad de que el robot adquiera conocimiento durante su funcionamiento vamos a ver en este apartado dos de las estrategias más utilizadas para la adaptación y el aprendizaje de los robots.

Como se ha visto en la sección anterior, para lograr robots que hagan tareas útiles en entornos complejos será necesario dotarlos de la capacidad de aprendizaje. Existen diversas estrategias de aprendizaje, siendo las más comunes para robótica el aprendizaje por demostración y el aprendizaje por interacción con el entorno. La tabla 1.2 muestra algunas de las características y diferencias básicas entre ambas estrategias de aprendizaje y la programación de controladores. A continuación se dará una breve introduccion a las técnicas de aprendizaje por demostración y aprendizaje por interacción con el entorno.

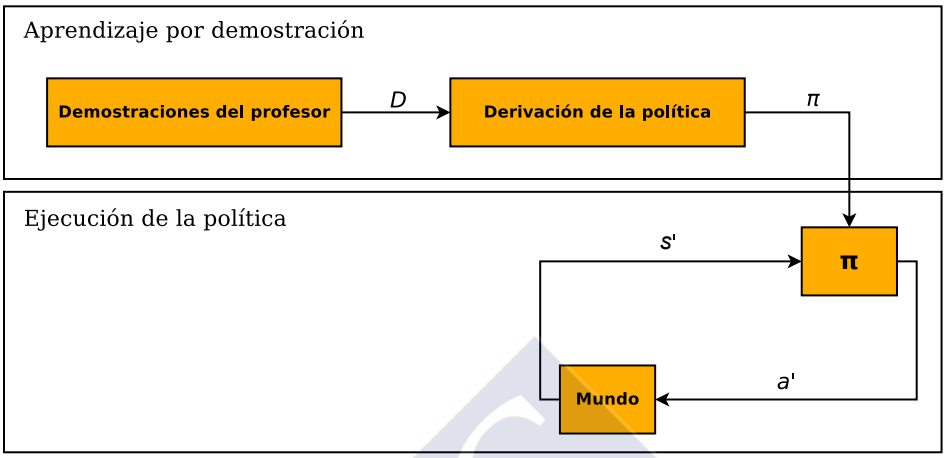


Figura 1.10: Esquema general del proceso de aprendizaje por demostración. Primero se observa al profesor y se deriva una política de control. A continuación esa política es utilizada para ejecutar la tarea.

1.4.1. Aprendizaje por demostración

En el aprendizaje por demostración una política de control es aprendida mediante ejemplos o demostraciones proporcionados por un profesor. Un ejemplo es una serie de pares estado-acción que son grabados mientras el profesor muestra el comportamiento deseado del robot. Dentro del aprendizaje por demostración existen diferentes aproximaciones, si bien no existe una terminología estándar para referirse a ellas. En nuestro caso usaremos la terminología utilizada en [6].

En el aprendizaje por demostración es necesario abordar varios problemas. El primero de ellos es cómo conseguir un conjunto de ejemplos válido para que el robot pueda aprender a partir de ellos. A continuación se debe calcular una política de control que permita reproducir el comportamiento observado en los ejemplos. Estos pasos se pueden ver en el esquema mostrado en la figura 1.10.

Construir el conjunto de ejemplos es un proceso en dos etapas: el primer paso es la grabación de la tarea mientras la ejecuta un profesor. El segundo paso es traducir dicha grabación a un conjunto de entrenamiento para el aprendizaje de un controlador. Durante el primer paso es necesario recopilar información relativa a lo que el profesor percibe en cada instante ($estado_{profesor}$), y lo que hace ($acción_{profesor}$). La proyección de la información grabada durante el proceso de demostración en un conjunto de pares estado-acción del profesor, es lo que

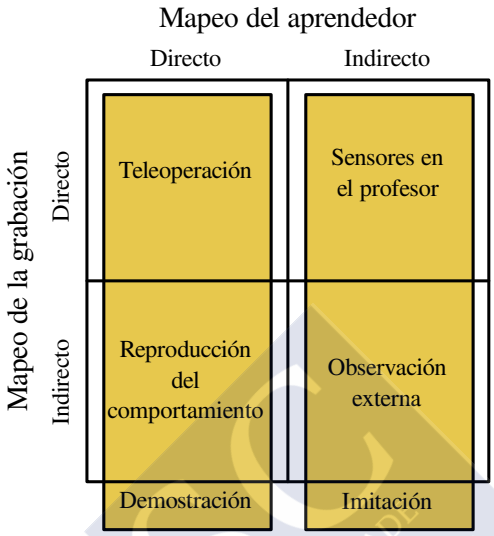


Figura 1.11: Tipos de aprendizaje por demostración dependiendo de cómo se haga el mapeo de la grabación y el mapeo del aprendedor.

se denomina *mapeo de la grabación*. Es importante destacar que estos pares estado-acción están referidos al profesor, y será necesario traducirlos a lo que en cada caso tendría que hacer el robot: $estado_{robot} - acción_{robot}$. Es precisamente este segundo conjunto de datos el que permitirá el aprendizaje de un controlador. Esta proyección de información, desde el punto de vista del profesor al punto de vista del robot que aprenderá la tarea, es lo que denominamos *mapeo del aprendedor*. En función de cómo se realicen estos dos procesos de mapeo se hablará de diferentes tipos de aprendizaje por demostración, que son los que se muestran en la figura 1.11: *teleoperación*, *sensores en el profesor*, *reproducción del comportamiento* y *observación externa*.

En el caso de la *teleoperación*, un operador humano controlará de forma remota el movimiento de un robot mientras éste reproduce la tarea que se pretende aprender. Tal y como se refleja en la figura 1.11, los procesos de mapeo son directos dado que durante el proceso de demostración se graba lo que el profesor percibe y hace. Ahora bien, considerando que la demostración la hace el propio robot los pares estado-acción grabados representan en sí mismos la información necesaria para el aprendizaje del controlador.

En el aprendizaje de *sensores en el profesor* se colocan sensores en el demostrador (habitualmente un humano) que ejecuta la tarea que se pretende aprender. De esta manera durante la demostración se obtiene información de lo que el profesor percibe y hace, razón por la que tal y como se muestra en la figura 1.11 el mapeo de la grabación es un proceso directo. No obstante, lo que el profesor percibe no tiene por qué coincidir con lo que vería el robot. A su vez, alguna de las acciones ejecutadas por el profesor pueden no ser realizables por el robot debido a sus limitaciones físicas. Esto hace que el mapeo del aprendedor sea más complejo (indirecto en la figura 1.11).

Para evitar la complejidad del mapeo del aprendedor de esta segunda opción, una alternativa consiste en lograr que un robot observe lo que hace el humano durante el proceso de demostración y lo reproduzca, es el aprendizaje mostrado en la figura 1.11 como *reproducción del comportamiento*. Con esta alternativa, durante la demostración se graba lo que el robot percibe y hace, razón por la que el mapeo del aprendedor es directo. Sin embargo, es necesario algún mecanismo que permita que el robot reproduzca *lo que ve* durante el proceso de demostración. Esta es la razón por la que en la figura 1.11 el mapeo de la grabación se describe como indirecto.

Finalmente, en el cuarto y último caso, *observación externa* (figura 1.11), un humano (profesor) ejecuta la tarea mientras se graba lo que sucede desde agentes externos (por ejemplo cámaras ubicadas en el entorno). Esta opción es la más simple desde el punto de vista del profesor (dado que no tendría que portar sensores específicos o teleoperar un robot), pero ofrece la mayor complejidad desde el punto de vista de los mapeos. En este caso es preciso diseñar alguna estrategia para obtener lo que el humano percibe del entorno y lo que hace en cada instante a partir de la información grabada (mapeo de la grabación indirecto). Por otra parte también es necesario traducir esta información a lo que el robot debe ejecutar en cada uno de estos instantes (mapeo del aprendedor indirecto).

Es importante destacar que en las opciones de *teleoperación* y *reproducción del comportamiento*, el robot que ejecutará la tarea participa en el proceso de demostración, mientras que en las dos opciones restantes, la demostración se restringe a un operador humano. En consecuencia la grabación de la demostración en los dos primeros casos se hace en el propio espacio de estados y acciones del robot. Es por esta razón por la que en la figura 1.11 las dos primeras opciones se encierran en lo que se llama “proceso de demostración”, mientras que las dos opciones restantes se consideran como opciones de “imitación”.

Una vez se tiene un conjunto de ejemplos preparados para su uso por el aprendedor, el siguiente paso es derivar la política de control. Existen tres aproximaciones principales a la hora de obtener la política de control [6]:

– Creación de una proyección percepción-acción: con esta técnica se aprende una función que aproxima el mapeo de estados a acciones observado en los ejemplos. Las técnicas más comunes para realizar esta aproximación son las técnicas de clasificación y de regresión.

– Modelo del sistema: se aprende un modelo de la dinámica del mundo, por lo que se conocen las transiciones entre estados. Gracias a esta información se puede derivar una política coherente con el modelo que resuelva la tarea.

– Planes: una alternativa es representar el comportamiento deseado del robot como un plan, es decir, una secuencia de acciones que llevan al sistema de un estado inicial a un estado final. Las acciones generalmente se definen mediante una serie de precondiciones y postcondiciones. A diferencia de otras aproximaciones, en el aprendizaje de planes no sólo interviene la secuencia de ejemplos, sino que se incluye cierta información sobre las intenciones del profesor.

Tras la grabación de los ejemplos y la derivación de la política de control se tiene un controlador que en el mejor de los casos servirá para ejecutar la tarea tal y como el profesor la enseñó. La calidad de la política aprendida depende enormemente de la calidad de las demostraciones. En general se asume que el conjunto de datos de ejemplo contiene demostraciones de calidad, sin embargo, las demostraciones del profesor pueden ser ambiguas, subóptimas o fallidas. Incluso en el caso de tener demostraciones perfectas, el comportamiento del robot sólo estará aprendido para aquellos estados que el profesor haya enseñado. Por lo tanto, se hace necesario que los ejemplos hagan un repaso exhaustivo por todo el espacio de estados, o la implementación de algún método que generalice lo aprendido a partir de los ejemplos grabados ante nuevas situaciones.

Debido a los problemas recién mencionados es muy probable que usando directamente aprendizaje por demostración el robot no consiga adquirir una buena política de control. Una estrategia que se está siguiendo actualmente para mejorar la política obtenida mediante aprendizaje por demostración es el utilizar aprendizaje a partir de la experiencia para mejorarla. Un ejemplo de esto se puede ver en [2], donde se enseña a un helicóptero de aeromodelismo autónomo a ejecutar maniobras acrobáticas. Primero un piloto experto ejecuta las maniobras,

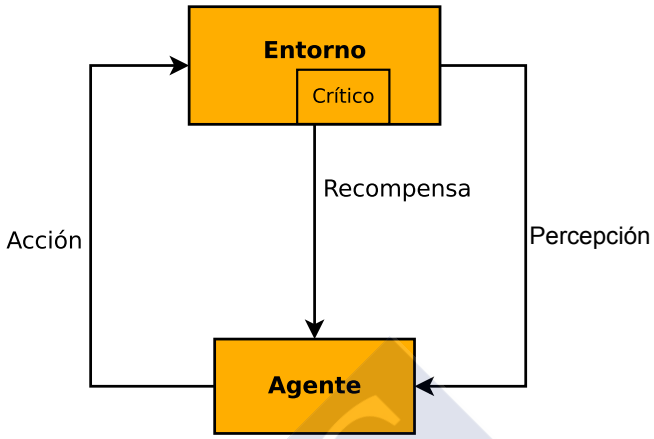


Figura 1.12: Esquema general del proceso de aprendizaje a partir de la experiencia: el robot interactúa con el entorno y recibe una recompensa a sus acciones. El objetivo del aprendizaje es maximizar dicha recompensa.

a partir de estos datos se obtiene un controlador que a continuación es mejorado mediante aprendizaje por refuerzo. En [73] un brazo robótico aprende a dar la vuelta a una tortilla. El aprendizaje por demostración se hace mediante un humano que mueve el brazo para ejecutar la tarea. Tras la demostración el robot no es capaz de dar la vuelta a la tortilla, pero tras diversos ciclos de prueba y error consigue un comportamiento exitoso. Una aplicación similar es la mostrada en [67], donde un brazo robótico aprende a jugar al boliche, donde el objetivo es introducir una pequeña bola en una recipiente sostenido en la mano. Al igual que en el ejemplo anterior, primero un humano mueve el brazo mientras ejecuta la tarea y a continuación se refina el comportamiento mediante aprendizaje por refuerzo.

1.4.2. Aprendizaje a partir de la experiencia

El aprendizaje a partir de la experiencia es una estrategia de aprendizaje donde el robot aprende una política mediante prueba y error al interactuar con el entorno. Desde el punto de vista del usuario este aprendizaje tiene la ventaja de que se puede percibir una mejora en el comportamiento del robot, ya que el robot aprende de sus fallos y evita los errores cometidos en el pasado. La mayor virtud de este tipo de aprendizaje es que no es necesario que un demostrador defina o ejecute el comportamiento deseado, únicamente se necesita proporcionar una recompensa en función de la calidad de la tarea ejecutada. Así el robot irá probando diferen-

tes acciones, y según el resultado de la ejecución recibirá una recompensa positiva o negativa. El algoritmo de aprendizaje irá recordando qué acciones son las que proporcionan mayor recompensa para así construir una política de control que maximice el refuerzo recibido a largo plazo (figura 1.12).

Tener una función que le indique al robot cuándo lo está haciendo bien o mal nos libera de tener que proporcionar lo que sería la acción correcta en cada caso, pero obtener esa función no es trivial, y una función de refuerzo errónea provocará que el comportamiento obtenido no sea el deseado o incluso que el robot sea incapaz de aprender.

Una opción para evitar este problema es la formulación automática de la función de refuerzo. Una forma de conseguirlo es mediante el aprendizaje por refuerzo inverso, con esta estrategia el objetivo es encontrar una función de refuerzo que explique el comportamiento observado [86]. Otra opción es la generación de refuerzo a partir de la observación del comportamiento humano. En el Grupo de Sistemas Intelixentes de la Universidade de Santiago de Compostela hemos trabajado en esta línea. En [100, 103] se genera automáticamente una señal de refuerzo que hace que el robot aprenda a moverse por las mismas áreas por las que transitan habitualmente las personas que trabajan en ese entorno.

Otra opción para evitar el diseño de una función de refuerzo es que sea un humano el que le indique al robot cuándo su comportamiento no es correcto. De esta manera, un observador humano puede proporcionar la señal de refuerzo sin necesidad de conocimientos de robótica, únicamente observando al robot y juzgando su comportamiento. En esta tesis se mostrará una implementación de esta estrategia, la cual se puede considerar como una aproximación entre el aprendizaje por refuerzo y el aprendizaje por demostración.

En el Capítulo 2 se dará una explicación pormenorizada del aprendizaje por refuerzo, así como de los diferentes algoritmos desarrollados para lograr este tipo de aprendizaje.

1.5. Discusión

En este capítulo se ha expuesto una clasificación de los diferentes tipos de robots, y dentro de ella dónde se engloba la robótica de servicios personales. Se espera que los robots personales sean tan comunes en el futuro como lo son hoy los teléfonos móviles, pero para lograr esto uno de los principales retos es el de dotar a estos robots de autonomía y adaptación.

Una característica deseable y esperada de los robots de servicio personales es la autonomía avanzada, esto es, el poder operar correctamente durante un determinado período de tiempo

en un entorno no estructurado y sin la ayuda de un operador humano. Para alcanzar dicha autonomía no se pueden aplicar las técnicas clásicas de programación de autómatas, ya que el entorno será cambiante, las condiciones de la tarea pueden alterarse o el robot puede sufrir cambios. Existe una gran incertidumbre que impide modelar cualquiera de estos tres aspectos, por lo que se hace necesario encontrar técnicas alternativas para dotar a los robots de la capacidad de adaptación y la posibilidad de que una persona no experta pueda condicionar el comportamiento del robot.

Para un robot de servicio una buena aproximación es el aprendizaje por demostración. Se ha visto cómo existen técnicas que permiten a un robot adquirir un comportamiento a partir de los ejemplos que le proporciona un humano. Sería deseable que un usuario de un robot, que no tiene por qué poseer conocimientos de robótica ni de programación pueda enseñar a su robot a ejecutar las tareas que quiera que éste haga.

Otra estrategia de aprendizaje válida para el ámbito de la robótica de servicios es el aprendizaje a partir de la experiencia. En este tipo de aprendizaje será el robot el que se desplace por el entorno y a partir de observar el resultado de sus acciones irá mejorando el comportamiento hasta lograr resolver la tarea.

El aprendizaje por demostración y el aprendizaje a partir de la experiencia no son técnicas incompatibles, y tal y como se mostró en la Sección 1.4.1 es común el combinarlas para conseguir un controlador satisfactorio. Ambas estrategias están abiertas a la investigación y en el futuro se realizarán importantes avances tanto en el aprendizaje por refuerzo como en el aprendizaje por demostración. Desde nuestro punto de vista consideramos que el aprendizaje por interacción con el entorno proporciona la capacidad de entrenar a un robot desde cero hasta la obtención de un controlador que resuelve la tarea, además de permitir la adaptación de dicho controlador durante todo el ciclo de vida del robot. Por ello la investigación mostrada en esta tesis está orientada a la adquisición de comportamientos mediante el aprendizaje por refuerzo.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Capítulo 2

Aprendizaje por refuerzo en robótica

En la sección 1.4.2 se introdujo el aprendizaje por refuerzo como un paradigma capaz de permitir que un robot aprenda a partir de su interacción con el entorno. En este capítulo se explicará más detalladamente este paradigma de aprendizaje, prestando especial atención a su aplicación en robótica. Este capítulo no pretende ser un análisis exhaustivo de los algoritmos de aprendizaje por refuerzo, sino que se introducirán las diferentes alternativas existentes en la actualidad.

2.1. Antecedentes

Los primeros estudios sobre el efecto del refuerzo en el aprendizaje se remontan al principio del siglo XX. Los experimentos llevados a cabo por Edward L. Thorndike [127, 128], donde se observaba el tiempo requerido por unos gatos para escapar de cajas-puzle, demostraron que en base a la experiencia acumulada las acciones poco efectivas iban siendo cada vez menos frecuentes. En particular, cuando los gatos eran encerrados por primera vez necesitaban mucho tiempo para escapar, a medida que las acciones correctas iban siendo más frecuentes los gatos eran capaces de escapar en un menor tiempo. De las varias acciones que había para una misma situación, se fortalecían aquellas a las que inmediatamente o en un corto período de tiempo seguía una respuesta satisfactoria para el animal. Por el contrario, se debilitaban aquellas acciones donde la respuesta era negativa. Cuanto mayor fuese la respues-

ta, positiva o negativa, reportada por una acción, la asociación entre situación y acción era reforzada o debilitada en mayor medida. Esta es la Ley de Efecto teorizada por Thorndike, piedra angular del condicionamiento operante (experimentos de Burrhus F. Skinner [119]): un comportamiento se ve fortalecido si al estímulo le sigue una recompensa – refuerzo positivo – y se ve debilitado si le sigue un castigo – refuerzo negativo.

Inspirados por estas teorías, Sutton y Barto [123] desarrollaron el aprendizaje por refuerzo como un paradigma de aprendizaje en máquinas que determina cómo un agente puede aprender a tomar las acciones en un entorno con el fin de maximizar un refuerzo a largo plazo. Lo que hace más atractivo a este paradigma de aprendizaje es el hecho de que el sistema aprende por sí mismo mediante prueba y error, dependiendo únicamente de sus propias experiencias y una señal de retroalimentación – el refuerzo– que estimula o coarta las diferentes secuencias de acciones.

El aprendizaje por refuerzo es particularmente adecuado para aquellas aplicaciones donde disponemos de algún tipo de información sobre el rendimiento del robot y donde no es fácil obtener un conjunto representativo de ejemplos. El aprendizaje por refuerzo utiliza una medida de rendimiento global (el refuerzo) para controlar el proceso de aprendizaje. En este aspecto es diferente a los paradigmas de aprendizaje supervisado [112], los cuales utilizan para el aprendizaje un conjunto de entrenamiento con entradas y las salidas deseadas para cada una de ellas. Evitar este tipo de conjuntos de entrenamiento puede ser especialmente útil en robótica, donde a menudo únicamente se conoce el comportamiento global deseado del robot.

En el aprendizaje por refuerzo se busca emplear el valor de recompensa para lograr un mapeo de percepciones a acciones que resuelva la tarea. En términos generales se puede decir que se tiene a un agente que debe aprender interactuando con el entorno para lograr un objetivo. Dicho agente debe ser capaz de percibir de alguna manera el estado del entorno y debe poder ejecutar acciones que afectan al estado (figura 2.1). Al agente aprendedor no se le comunicará qué acciones debería haber ejecutado y lo que hará será explorar diferentes combinaciones tratando de alcanzar aquellas que maximizan el refuerzo. En muchas ocasiones, las acciones no solo afectarán al refuerzo inmediato, sino que también afectarán a la siguiente situación y por consiguiente a todos los refuerzos siguientes. Estas dos características, prueba y error y refuerzos retardados, son las más importantes del aprendizaje por refuerzo.

Un aspecto muy importante es que el término *aprendizaje por refuerzo* representa un paradigma de aprendizaje, esto es, una forma de adquirir conocimiento, pero no representa en sí mismo ningún algoritmo particular. Cualquier estrategia adecuada para resolver el proble-

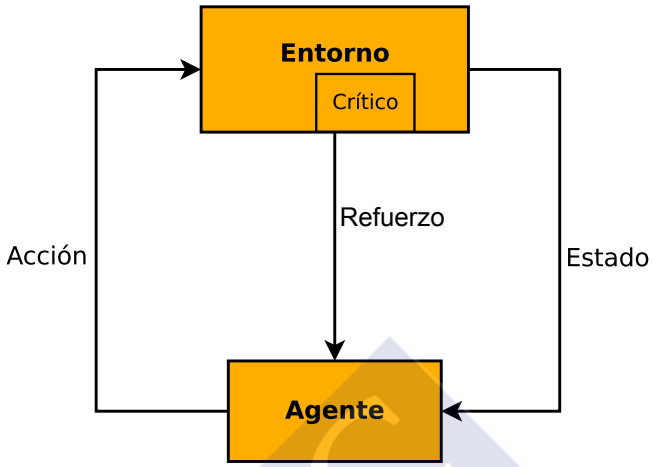


Figura 2.1: Esquema general del proceso de aprendizaje por refuerzo.

ma de aprendizaje a partir de la información sobre el rendimiento puede ser considerado un método de aprendizaje por refuerzo.

2.2. El problema de aprendizaje por refuerzo

Un sistema de aprendizaje por refuerzo se compone, aparte del agente y el entorno (figura 2.1), de cuatro elementos principales: una política de control, una función de refuerzo, una función de valor y, opcionalmente, un modelo del entorno. Dado que nos centraremos en el aprendizaje por refuerzo en robots, de aquí en adelante nos referiremos al agente aprendedor indistintamente como robot o como agente.

La política de control define la acción que el robot va a ejecutar en cada uno de los estados que visite. Esta política es lo que se pretende aprender, e irá evolucionando durante el proceso de aprendizaje hasta encontrar aquella que maximiza el refuerzo. En esta tesis entendemos como estado aquella situación del entorno que el robot considera relevante y que se puede percibir a través de sus sensores.

La función de refuerzo define el objetivo en un problema de aprendizaje por refuerzo ya que el refuerzo indica si el robot lo está haciendo bien o mal, aunque no indica qué acciones debería haber ejecutado para hacerlo bien. La única finalidad del robot es encontrar una política de control que maximiza el refuerzo que recibe a largo plazo. Desde este punto de

vista el aprendizaje por refuerzo puede ser visto como un problema de optimización, donde el objetivo es maximizar el refuerzo recibido.

Mientras que la función de refuerzo indica de una manera más inmediata el carácter negativo o positivo de las últimas acciones ejecutadas, la función de valor especifica lo que es bueno a largo plazo. La función de valor es una estimación de los refuerzos que recibirá el robot siguiendo una política de control determinada. Para ello se debe tener en cuenta no sólo el estado actual, sino también los estados que probablemente le sigan y los refuerzos asociados a dichos estados.

El cuarto y último elemento de algunos sistemas de aprendizaje por refuerzo es un modelo del entorno. Un modelo permite predecir, dados un estado y una acción, cuales serán el estado y el refuerzo siguientes. La ventaja de usar un modelo es que facilita el evaluar y mejorar políticas de control sin necesidad de desplazar físicamente al robot por el entorno. La contrapartida es que su uso implica mayor capacidad computacional y de memoria, aparte también sería necesario un modelo que represente de manera precisa o fiable el entorno, lo que no siempre es factible.

De forma clásica, para representaciones discretas de estados, existen dos alternativas a la hora de representar un sistema de aprendizaje por refuerzo: Procesos de Decisión de Markov y Procesos de Decisión de Markov Parcialmente Observables.

2.2.1. Procesos de Decisión de Markov

Un Proceso de Decisión Markov (MDP) [29] es una secuencia de variables $(x_1, x_2, ..., x_n)$ con la propiedad de que cualquier predicción de x_n se puede hacer a partir del conocimiento de x_{n-1}. En otras palabras, cualquier futuro valor de una variable depende únicamente del valor actual de dicha variable, y no en la secuencia de valores pasados.

En el caso que nos ocupa en esta tesis, un proceso de Decisión de Markov viene definido por la tupla $< S, A, P, R >$, dada por:

- Un espacio de estados S.
- Un espacio de acciones A.
- Una matriz de probabilidades de transición de estados P que indica, dados un estado, una acción y otro estado, la probabilidad de pasar del primer al segundo estado ejecutando la acción dada. $P(s, a, s') : S \times A \times S \rightarrow \Re$. Dados un estado s y una acción a, la

probabilidad de transición a cada posible estado siguiente es:

$$\mathscr{P}^a_{ss'} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}. \tag{2.1}$$

A partir de P y de la acción ejecutada en un instante dado, se puede conocer la distribución de probabilidad del estado siguiente en el que se encontrará el robot.

– Una función de refuerzo R que especifica el refuerzo que el robot recibe al tomar la acción $a \in A$ en el estado $s \in S$ y realizar una transición al estado $s' \in S$. $R(s,a,s') : S \times A \times S \to \Re$. El refuerzo esperado al pasar de un estado s a un estado s' mediante la acción a es:

$$\mathscr{R}^a_{ss'} = E\{r_t \mid s_t = s, a_t = a, s_{t+1} = s'\}. \tag{2.2}$$

2.2.2. Procesos de Decisión de Markov Parcialmente Observables

Un problema común en las aplicaciones de robótica es que el estado puede no ser completamente observable, por ejemplo debido al ruido sensorial, por lo que no se conocerá con exactitud el estado actual del sistema. Es lo que se conoce como Procesos de Decisión de Markov Parcialmente Observables (POMDP) [58]. En este caso se debe añadir un modelo de las observaciones que especifique la probabilidad de hacer la observación o tras haber ejecutado la acción a en el estado s.

Formalmente un POMDP es una tupla $< S,A,P,R,O,Z >$, donde S,A,P,R describen un MDP, O es el conjunto de observaciones acerca del mundo que el agente experimenta, y Z define la distribución de probabilidad de las observaciones en función del estado actual y la acción ejecutada anteriormente $\mathscr{Z}^a_{s'o'} = \Pr\{o_{t+1} = o' \mid s_{t+1} = s', a_t = a\}$. El agente mantendrá una distribución de probabilidad sobre S, llamada *belief state* que representa, para cada estado $s \in S$ la probabilidad de que el robot se encuentre en dicho estado en función de la distribución de probabilidad de estar en cada uno de los estados en el instante anterior b, la acción ejecutada a, y la observación actual o': $b'(s') = \Pr\{s_{t+1} = s' | o_{t+1} = o', a_t = a, b_t = b\}$.

Debido a la mayor incertidumbre asociada a la falta de información sobre el estado, la estimación del estado siguiente implica una multiplicación de las matrices de probabilidad que incluya toda la información disponible [27]:

$$b'(s') = \frac{\mathscr{Z}^a_{s'o'} \sum_{s \in S} \mathscr{P}^a_{ss'} b(s)}{\Pr\{o'|a,b\}} \tag{2.3}$$

Donde $\Pr\{o'|a,b\} = \sum_{s \in S} \sum_{s' \in S} b(s) \mathscr{P}^a_{ss'} \mathscr{Z}^a_{s'o'}$ es un factor de normalización.

De aquí en adelante asumiremos que nuestro robot opera en un Proceso de Decisión de Markov (MDP), por lo que no se incluirá el modelo de las observaciones en las ecuaciones. No obstante, en [48] se puede consultar la extensión de la formulación que se mostrará a continuación para el caso de POMDP.

2.2.3. Política de control

Tal y como se dijo anteriormente, una política de control es un mapeo entre estados y acciones. La política de control dirige el comportamiento del robot ya que proporciona para cada estado la acción a ejecutar. Durante el proceso de aprendizaje por refuerzo el robot desarrolla una política π_t, donde $\pi_t(s) \in A$ indica la acción que se ejecutará en el instante t en el estado s. El objetivo del aprendizaje por refuerzo es encontrar una política de control que maximiza el refuerzo que recibirá el robot a partir del instante actual, R_t:

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \tag{2.4}$$

donde $0 \leq \gamma \leq 1$ es lo que se conoce como coeficiente de descuento. Su función es evitar que el valor de R_t pueda ser infinito. Si $\gamma = 0$, entonces únicamente se valora el refuerzo inmediato. A medida que γ se aproxima a 1, los refuerzos futuros tendrán más importancia que el actual [4].

Dentro de la teoría de control, las políticas pueden ser estocásticas o deterministas:

– En el caso de políticas de control estocásticas no se conocerá con exactitud qué acción será ejecutada en cada estado, sino que cada acción tendrá una cierta probabilidad de ser ejecutada. Se define una política de control estocástica $\pi(s,a)$ como la probabilidad de ejecutar la acción a en el estado s y el instante t:

$$\begin{aligned} \pi : S, A \quad &\rightarrow \Re \\ s \in S, a \in A &\rightarrow \pi(s,a) \in [0,1] \end{aligned} \tag{2.5}$$

– En una política de control determinista siempre se ejecutará la misma acción para un estado dado. Una política de control determinista $\pi(s)$ es la acción ejecutada en el estado s y el instante t. En este caso la política no representa una probabilidad, sino la

acción que será ejecutada:

$$\begin{aligned} \pi : S &\rightarrow A \\ s \in S &\rightarrow \pi(s) \in A \end{aligned} \tag{2.6}$$

En esta tesis trabajaremos con políticas de control deterministas dado que es el caso más habitual. En cualquier caso, quisiéramos destacar que toda la formulación presentada en esta tesis se puede adaptar fácilmente para políticas de control estocásticas.

2.2.4. Funciones de valor

Casi todos los algoritmos de aprendizaje por refuerzo se basan en estimar funciones de valor que pueden estar referidas a un estado $V^{\pi}(s)$ o a un par estado-acción $Q^{\pi}(s,a)$, que estiman cómo de bueno será para el agente estar en dicho estado (o ejecutar la acción en el estado). La idea de "cómo de bueno" se define en términos del valor de refuerzo esperado en el futuro. $V^{\pi}(s)$ representa el refuerzo esperado si a partir del estado s el robot sigue la política π. En el segundo caso $Q^{\pi}(s,a)$, usualmente llamado Q-valor de a en s, representa el refuerzo esperado si en el estado s se ejecuta la acción a y a continuación se sigue la política π.

Las funciones de valor deben ser estimadas y actualizados a partir de la secuencia de observaciones y refuerzos que un agente realiza durante todo el aprendizaje. De hecho, el componente más importante de prácticamente cualquier algoritmo de aprendizaje por refuerzo es el método para estimar de manera eficiente estas valoraciones.

Dado que el refuerzo que el robot recibirá en el futuro depende de las acciones que ejecuta, las funciones de valor se definen respecto a una política. Por lo tanto, el valor del estado s bajo la política π, representado como $V^{\pi}(s)$, es el refuerzo esperado empezando en s y siguiendo la política π. Para un Proceso de Decisión de Markov se define $V^{\pi}(s)$ como:

$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s\right\}, \tag{2.7}$$

donde $E_{\pi}\{\}$ es el valor esperado si el agente sigue la política π. V^{π} es la función de valor de estado de la política π.

De manera similar $Q^{\pi}(s,a)$ define el valor de ejecutar la acción a en el estado s y siguiendo a continuación la política π:

$$Q^{\pi}(s,a) = E_{\pi}\{R_t \mid s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, a_t = a\right\}. \tag{2.8}$$

Una propiedad fundamental de las funciones de valor es que satisfacen relaciones recursivas particulares. Para cualquier política π y cualquier estado s, la siguiente condición se mantiene entre el valor de s y el valor de sus posibles estados sucesores:

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\{R_t \mid s_t = s\} \\ &= E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \;\middle|\; s_t = s \right\} \\ &= E_{\pi}\left\{ r_t + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \;\middle|\; s_t = s \right\} \\ &= \sum_{s'} \mathscr{P}_{ss'}^{\pi(s)} \left[\mathscr{R}_{ss'}^{\pi(s)} + \gamma E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \;\middle|\; s_{t+1} = s' \right\} \right] \\ &= \sum_{s'} \mathscr{P}_{ss'}^{\pi(s)} \left[\mathscr{R}_{ss'}^{\pi(s)} + \gamma V^{\pi}(s') \right]. \end{aligned} \tag{2.9}$$

La ecuación 2.9 es la ecuación de Bellman para V^{π}. Expresa la relación entre el valor de un estado y los valores de sus estados sucesores. La ecuación de Bellman es la base para calcular, aproximar y aprender V^{π} [135].

De la misma manera, se puede obtener la ecuación de Bellman para Q^{π}:

$$\begin{aligned} Q^{\pi}(s,a) &= E_{\pi}\{R_t \mid s_t = s, a_t = a\} \\ &= E_{\pi}\left\{ r_t + \gamma V^{\pi}(s_{t+1}) \mid s_t = s, a_t = a \right\} \\ &= \sum_{s'} \mathscr{P}_{ss'}^{a} \left[\mathscr{R}_{ss'}^{a} + \gamma V^{\pi}(s') \right], \end{aligned} \tag{2.10}$$

Una forma muy común de representar las funciones de valoración es en función de la política de control óptima en lugar de una política determinada. La política de control óptima es aquella que maximiza las funciones de valor para cada estado, y por lo tanto no existe ninguna política mejor que ella. Dado que durante el aprendizaje no se conoce la política óptima, se toma como mejor política hasta el momento la política egoísta o política *greedy*. La política egoísta es aquella que para cada estado selecciona la acción con mayor valoración de acuerdo a lo aprendido hasta ese momento, por lo tanto toda política egoísta será egoísta

con respecto a una función de valor:

$$\begin{aligned}\pi^*(s) &= \arg\max_a Q^\pi(s,a) \\ &= \arg\max_a E\left\{r_t + \gamma V^\pi(s_{t+1}) \mid s_t = t, a_t = a\right\} \\ &= \arg\max_a \sum_{s'} \mathscr{P}^a_{ss'}\left[\mathscr{R}^a_{ss'} + \gamma V^\pi(s')\right],\end{aligned} \tag{2.11}$$

Las funciones de valor de la política óptima se denominan V^* y Q^*. Las ecuaciones de Bellman para las funciones de valor óptimas V^* y Q^* son:

$$\begin{aligned}V^*(s) &= \max_a E\left\{r_t + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a\right\} \\ &= \max_a \sum_{s'} \mathscr{P}^a_{ss'}\left[\mathscr{R}^a_{ss'} + \gamma V^*(s')\right]\end{aligned} \tag{2.12}$$

$$\begin{aligned}Q^*(s,a) &= E\left\{r_t + \gamma \max_{a'} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a\right\} \\ &= \sum_{s'} \mathscr{P}^a_{ss'}\left[\mathscr{R}^a_{ss'} + \gamma \max_{a'} Q^*(s', a')\right]\end{aligned} \tag{2.13}$$

Las ecuaciones anteriores permiten conocer cómo de bueno es un estado o un par estado-acción. Entonces, para conseguir una nueva política que sea mejor, o al menos igual de buena, que la mejor de las políticas actuales se elegirá para cada estado la acción con mejor valoración, la acción egoísta.

2.3. Algoritmos de aprendizaje por refuerzo

Los algoritmos de aprendizaje por refuerzo comparten muchos aspectos comunes, pero presentan ciertas características que permiten establecer una diferenciación entre ellos.

Una primera división que haremos respecto a los algoritmos de aprendizaje por refuerzo será en base a si éstos necesitan un modelo del entorno o no. Un modelo del entorno consiste en la información sobre la probabilidad de transición entre estados, el refuerzo que se recibirá en cada estado o en cada transición entre estados, y cualquier otra información que permita predecir el estado futuro del robot en base al estado y acción actuales. Aquellos algoritmos que utilicen un modelo será más complicado que puedan ser implementados en un robot real, esto es debido a la imposibilidad de obtener un modelo fiable de un entorno real (ver sección

1.3). Por otra parte, el disponer de un modelo del sistema permite aplicar técnicas que aceleran el aprendizaje, ya que no es necesaria una fase de exploración para conocer el resultado de la ejecución de las acciones.

Los algoritmos que no utilizan un modelo del entorno se basarán en el refuerzo obtenido tras cada iteración para calcular las funciones de valor. Estos modelos no necesitan almacenar una matriz de probabilidades de transición ni la matriz de refuerzos (sección 2.2.1). En esta sección se mostrarán algunos de los algoritmos utilizados en ambos tipos de aprendizaje por refuerzo: la programación dinámica, la cual utiliza un modelo, los métodos de Monte Carlo y los métodos de diferencias temporales. Estos dos últimos métodos no requieren el uso de un modelo.

2.3.1. Programación dinámica

La programación dinámica es una técnica de optimización útil en la toma de una serie de decisiones interrelacionadas [13], ya que proporciona un procedimiento sistemático para determinar la combinación de decisiones que maximiza el refuerzo recibido. El aprendizaje por refuerzo se puede entender como un problema de optimización, y por lo tanto es posible aplicar programación dinámica para resolverlo. En este caso se debe encontrar la política de control que maximiza las funciones de valor, y la interrelación de las decisiones viene dada por el hecho de que exista una cadena de estados y acciones, lo cual implica que las decisiones tomadas en un estado afectarán a los demás. Una propiedad de la programación dinámica es que, dada la solución óptima a un problema, ésta contiene la solución óptima a cualquier subproblema del mismo. Gracias a esto, la política de control obtenida mediante programación dinámica será la política de control óptima para cada uno de los estados.

Dado un modelo perfecto del entorno la programación dinámica puede usarse para calcular la política óptima. Si se conoce a la perfección la dinámica del entorno, la ecuación 2.9 es un sistema de $|S|$ ecuaciones lineales con $|S|$ incógnitas (los valores $V^{\pi}(s), s \in S$). Aunque la solución es directa, requiere mucha computación. Otra solución es usar un método iterativo que vaya aproximando la función de valoración en función del valor en la iteración anterior. Dentro del contexto del aprendizaje por refuerzo, la programación dinámica se divide en dos estrategias: iteración de política e iteración de valor [123]. Estos algoritmos se obtienen adaptando las ecuaciones de Bellman a reglas de actualización para mejorar las funciones de valor.

Algoritmo 2.1 Iteración de política

Inicializar aleatoriamente $V(s) \in \Re$ y $\pi(s) \in A \ \forall s \in S$
repetir

Evaluación de política
repetir
$\Delta \leftarrow 0$
para todo $s \in S$ **hacer**
$v \leftarrow V(s)$
$V(s) \leftarrow \sum_{s'} \mathscr{P}_{ss'}^{\pi(s)}[\mathscr{R}_{ss'}^{\pi(s)} + \gamma V(s')]$
$\Delta \leftarrow \max(\Delta, |v - V(s)|)$
fin para
hasta que $\Delta < \theta$ (un número positivo pequeño)

Mejora de política
$politica\_estable \leftarrow$ verdadero
para todo $s \in S$ **hacer**
$b \leftarrow \pi(s)$
$\pi(s) \leftarrow \arg\max_a \sum_{s'} \mathscr{P}_{ss'}^{a}[\mathscr{R}_{ss'}^{a} + \gamma V(s')]$
si $b \neq \pi(s)$ **entonces**
$politica\_estable \leftarrow$ falso
fin si
fin para
hasta que $politica\_estable =$ verdadero

Iteración de política

El proceso de iteración de política se basa en ir alternando etapas de evaluación y mejora de la política de control hasta encontrar una solución satisfactoria. En la etapa de evaluación de política se calcula V^{π} de una política π constante, y en la etapa de mejora de política se modifica la política π con el fin de buscar otra alternativa π' que proporcione mayor valoración, $V^{\pi'} > V^{\pi}$. El algoritmo 2.1 muestra el algoritmo de iteración de política para políticas deterministas.

Considerando una secuencia de funciones de valor aproximadas $V_0, V_1, V_2, ...$, la aproximación inicial V_0 escoge un valor arbitrario para todos los estados, exceptuando el estado terminal, si existe, donde su valor debe ser 0. Siguiendo un proceso iterativo se puede ir afinando el valor de V teniendo en cuenta el valor en el instante anterior. Durante la etapa de

evaluación de política (algoritmo 2.1) cada aproximación sucesiva de V es obtenida usando la ecuación de Bellman como regla de actualización:

$$\begin{aligned} V_{k+1}(s) &= E_\pi\{r_t + \gamma V_k(s_{t+1}) \mid s_t = s\} \\ &= \sum_{s'} \mathscr{P}_{ss'}^{\pi(s)}[\mathscr{R}_{ss'}^{\pi(s)} + \gamma V_k(s')], \end{aligned} \tag{2.14}$$

$\forall s \in S$. La secuencia $\{V_k\}$ converge a V^π cuando $k \to \infty$, en la práctica se asume que el algoritmo alcanza la convergencia cuando el cambio de V es cercano a 0. Este algoritmo es llamado evaluación iterativa de política.

Una vez se tiene una función de valor V^π para una política arbitraria π, lo que interesa es encontrar una política alternativa mejor, es la etapa de mejora de política. Por lo tanto, lo que debemos saber es si la valoración mejoraría al escoger una acción diferente en un estado dado. La forma de hacerlo es considerando la selección de la acción egoísta $\pi^*(s)$ (ecuación 2.11), y siguiendo a continuación la política existente π. Entonces, la nueva política será igual de buena o mejor que π si

$$Q^\pi(s, \pi^*(s)) \geq V^\pi(s). \tag{2.15}$$

La ecuación 2.11 nos permite mejorar una política π a una nueva política π' a partir de V^π. Entonces es posible calcular $V^{\pi'}$ y mejorar otra vez para obtener una política aún mejor π''. Aplicado sucesivas mejoras y evaluaciones se obtiene una secuencia de políticas y funciones de valor:

$$\pi_0 \xrightarrow{\text{E}} V^{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} V^{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi^* \xrightarrow{\text{E}} V^*,$$

donde $\xrightarrow{\text{E}}$ representa una evaluación de política y $\xrightarrow{\text{I}}$ representa una mejora de política. Dado que un MDP tiene un número finito de políticas, este proceso converge a una política óptima y a una función de valor óptima en un número finito de iteraciones.

Iteración de valor

Un problema de la iteración de política es que cada una de sus iteraciones implica un proceso de evaluación de política (algoritmo 2.1), lo que puede acarrear un costo computacional elevado. Una manera de acortar el proceso es combinar la evaluación de política con la mejora de política en un único paso. De esta manera se actualiza V sobre la política egoísta en cada momento, por lo que no habrá una política constante. El algoritmo 2.2 muestra el proceso resultante, donde se actualiza el valor de $V(s)$ en función de la acción que proporciona mayor

Algoritmo 2.2 Iteración de valor

Inicializar aleatoriamente $V(s) \in \Re \ \forall s \in S$
repetir
  $\Delta \leftarrow 0$
  para todo $s \in S$ **hacer**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s'} \mathscr{P}^a_{ss'}[\mathscr{R}^a_{ss'} + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  fin para
hasta que $\Delta < \theta$ (un número positivo pequeño)

devolver Política determinista π tal que $\pi(s) = \arg\max_a \sum_{s'} \mathscr{P}^a_{ss'}[\mathscr{R}^a_{ss'} + \gamma V(s')]$

refuerzo. A esta ecuación se llega transformando la ecuación óptima de Bellman (ecuación 2.12) en una regla de actualización.

Al igual que en la iteración de política, en el caso de la iteración de valor la convergencia a V^* se alcanzaría en el infinito. En la práctica lo que se hace es parar una vez que el cambio en la función de valor entre iteraciones sucesivas alcanza un valor mínimo.

Los algoritmos clásicos de programación dinámica son de una utilidad limitada en el aprendizaje por refuerzo en robótica debido a la asunción de un modelo perfecto. A continuación se presenta una serie de algoritmos que no requieren un modelo del entorno.

2.3.2. Métodos de Monte Carlo

Los métodos de Monte Carlo, al igual que la programación dinámica, pretenden aproximar funciones de valor. Pero a diferencia de la programación dinámica, los métodos de Monte Carlo no necesitan un conocimiento completo del entorno, no requieren un modelo. En lugar de la dinámica del entorno los métodos de Monte Carlo se basan en la experiencia, secuencias de estados, acciones y refuerzos obtenidos mediante interacción directa con el entorno, por lo que es necesario que el robot interactúe con su entorno para obtener estas observaciones a partir de las cuales se calculan las funciones de valor. Se asume que el conjunto de experiencias se divide en episodios, y que todos los episodios terminan en un momento dado. Sólo cuando el episodio termina se cambian las valoraciones y las políticas de control.

Para calcular las funciones de valor para una política de control dada a partir de la experiencia, la idea más simple es promediar los refuerzos observados tras cada visita a un estado. En particular, supongamos que queremos estimar $V^{\pi}(s)$ a partir de un conjunto de episodios en los que se pasa por el estado s mientras se ejecuta la política π. Dado que en un episodio puede aparecer el mismo estado en varias ocasiones, se puede calcular $V^{\pi}(s)$ de dos maneras: si se calcula $V^{\pi}(s)$ como la media de los refuerzos obtenidos tras cada aparición de s se le llama método de Monte Carlo de *cada-visita*. Si en lugar de promediar todas las apariciones de s se utiliza únicamente el refuerzo tras la primera aparición se le llama método de Monte Carlo de *primera-visita*. Ambos métodos convergen a $V^{*}(s)$ en el infinito.

Sin un modelo del entorno disponible se hace más adecuado el cálculo de valores de acción que de valores de estado. Ya que no conocemos la dinámica del entorno no se puede escoger la acción que nos lleve al siguiente estado mejor valorado. Es por eso que se hace necesario conocer el valor de cada acción para poder escoger la mejor valorada en cada estado.

La forma de calcular la función de valor de acción es similar al cálculo de la función de valor de estado. Dada una serie de episodios, para estimar $Q^{\pi}(s,a)$ se promedia entre los refuerzos obtenidos tras cada visita (o tras la primera en el caso de un método de *primera-visita*) a s y la ejecución de a. El problema es que en el caso de una política determinista muchas acciones no se ejecutarán, con lo que no se podrá calcular su función de valor y no se podrá mejorar la política. La solución será o bien obligar a que cada episodio empiece con un par estado-acción diferente, lo que puede ser imposible en una prueba real; o bien utilizar políticas estocásticas, donde cada acción tiene una probabilidad distinta de cero de ser seleccionada en cada estado. Por lo tanto con los métodos de Monte Carlo se debe alcanzar un compromiso entre la explotación de acciones conocidas y la exploración de nuevas acciones aún no ejecutadas.

El algoritmo 2.3 muestra el método de Monte Carlo para valoraciones de pares estado-acción y para *primera-visita*. Como se puede observar en el algoritmo, la mejora de política se realiza calculando la política egoísta respecto a la actual función de valor:

$$\pi_{k+1}(s) \leftarrow \arg\max_{a} Q^{\pi_k}(s,a). \tag{2.16}$$

2.3.3. Diferencias temporales

El aprendizaje basado en diferencias temporales es una combinación de las ideas de la programación dinámica y de los métodos de Monte Carlo. Los métodos de diferencias tem-

Algoritmo 2.3 Método de Monte Carlo de *primera-visita*

para todo $s \in S, a \in A$ **hacer**
  $Q(s,a) \leftarrow$ Inicializar aleatoriamente
  $\pi(s) \leftarrow$ Inicializar aleatoriamente
  $Refuerzo(s,a) \leftarrow$ Lista vacía
fin para
repetir
  Generar un episodio utilizando estados iniciales exploratorios y π
  para todo par s,a en el episodio **hacer**
    $R \leftarrow$ refuerzo tras la primera ocurrencia de s,a
    Añadir R a la lista $Refuerzo(s,a)$
    $Q(s,a) \leftarrow$ promedio $Refuerzo(s,a)$
  fin para
  para todo s en el episodio **hacer**
    $\pi(s) \leftarrow \arg\max_a Q(s,a)$
  fin para
fin repetir

porales actualizan las estimaciones de las funciones de valor en base a otras estimaciones, sin esperar a que el episodio finalice. Y tal y como ocurre con los métodos de Monte Carlo, el aprendizaje por diferencias temporales puede aprender directamente a partir de la experiencia sin un modelo del entorno.

Al igual que los métodos de Monte Carlo, las diferencias temporales utilizan la experiencia para actualizar las funciones de valor. El caso más simple de diferencias temporales se conoce como $TD(0)$. El 0 hace referencia a que la función de valoración de un estado se actualiza en base a la valoración del estado siguiente. Para llevar a cabo dicha actualización se recurre a la ecuación de Bellman:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_t + \gamma V(s_{t+1}) - V(s_t)], \tag{2.17}$$

donde r_t es el refuerzo recibido tras visitar el estado s_t, y α es un parámetro constante que controla el cambio de V.

Al igual que con la programación dinámica y los métodos de Monte Carlo, con las diferencias temporales se sigue un método iterativo para conseguir una política de control óptima. Con el aprendizaje de diferencias temporales será necesario un compromiso entre exploración y explotación, lo que divide los algoritmos de diferencias temporales en *on-policy* y *off-policy*.

Algoritmo 2.4 Algoritmo Sarsa

$Q(s,a) \leftarrow$ valores iniciales aleatorios
repetir
  Observar estado inicial s
  Seleccionar a para s a partir de una política derivada de Q (p.ej. egoísta)
  repetir
    Ejecutar acción a, observar r, s'
    Seleccionar a' para s' a partir de una política derivada de Q (p.ej. egoísta)
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$
    $s \leftarrow s'; a \leftarrow a';$
  hasta que Fin del episodio
hasta que Convergencia alcanzada

En los algoritmos *on-policy* la política que se está ejecutando es la misma que la que se está evaluando. Los algoritmos *off-policy* evalúan una política mientras que la política siendo ejecutada no tiene por qué ser la misma. A continuación vamos a describir algunos de los algoritmos más conocidos de aprendizaje por diferencias temporales.

Sarsa

Sarsa es un algoritmo *on-policy* para encontrar una política de control mediante diferencias temporales. Para un algoritmo de este tipo se debe estimar la función de valor de acción $Q^{\pi}(s,a)$ para la política actual π y para todos los estados s y acciones a. La regla de actualización usando $TD(0)$ es:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_t + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)]. \tag{2.18}$$

Esta actualización se realiza tras cada transición desde un estado no terminal s_t al estado siguiente s_{t+1}. Si s_{t+1} es terminal, entonces $Q(s_{t+1},a_{t+1}) = 0$. Esta regla utiliza la tupla de elementos $(s_t,a_t,r_t,s_{t+1},a_{t+1})$, que es de donde procede el nombre del algoritmo (Sarsa).

Q-Learning

Uno de los avances más importantes en el aprendizaje por refuerzo ha sido el desarrollo del algoritmo de aprendizaje de diferencias temporales *off-policy* conocido como *Q-learning* [135]. *Q-learning* es uno de los algoritmos de aprendizaje por refuerzo más populares. Su

Algoritmo 2.5 Algoritmo *Q-learning*

$Q(s,a) \leftarrow$ valores iniciales aleatorios
repetir
  Observar estado inicial s
  repetir
    Seleccionar a para s a partir de una política derivada de Q (p.ej. egoísta)
    Ejecutar acción a, observar r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
    $s \leftarrow s'$
  hasta que Fin del episodio
hasta que Convergencia alcanzada

forma más simple, Q(0), se define por

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \left[r_t + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t) \right]. \tag{2.19}$$

En este caso, la función de valor de acción aprendida, Q, aproxima directamente Q^*, la función de valor de acción egoísta, independientemente de la política que se esté ejecutando.

Métodos actor-crítico

La principal característica de los métodos actor-crítico [72] es que disponen de una estructura de memoria para almacenar la política de control de manera explícita e independiente de las funciones de valor. Son métodos *on-policy* ya que la política, conocida como actor, se usa para seleccionar las acciones, y las funciones de valor, el crítico, evalúan dicha política. La valoración tendrá la forma de un error de diferencias temporales como el mostrado a continuación:

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t), \tag{2.20}$$

donde V es la función de valor implementada por el crítico. Este error se utiliza para evaluar la acción a_t que se acaba de ejecutar en el estado s_t, de manera que si el error es positivo se debe incrementar la posibilidad de ejecutar a_t en s_t, o disminuirla en caso contrario. Si, por ejemplo, la probabilidad de seleccionar una acción en un estado es:

$$Pr\{a_t = a, s_t = s\} = \frac{e^{p(s,a)}}{\sum_b e^{p(s,b)}}, \tag{2.21}$$

donde $p(s,a)$ indica la preferencia para escoger la acción a en el estado s. Dicha preferencia puede ser modificada en base al error de la siguiente manera:

$$p(s_t,a_t) \leftarrow p(s_t,a_t) + \beta\,\delta_t, \tag{2.22}$$

donde $\beta > 0$ es el coeficiente de aprendizaje.

Trazas de elegibilidad

Uno de los problemas de los algoritmos de aprendizaje por diferencias temporales es su lentitud, ya que en cada iteración únicamente se actualiza la valoración del estado visitado en la iteración anterior. Las trazas de elegibilidad [135] solucionan esto añadiendo más memoria al sistema, permitiendo la actualización de las valoraciones asociadas a varios estados en cada iteración. Las trazas de elegibilidad son uno de los mecanismos básicos de aprendizaje por refuerzo. Prácticamente cualquier algoritmo de diferencias temporales puede ser combinado con trazas de elegibilidad para obtener un método que pueda aprender de una manera más eficiente.

Las trazas de elegibilidad se pueden ver como un puente entre los métodos de Monte Carlo, donde se actualizan las valoraciones en base a todo el episodio, y las diferencias temporales, donde las funciones de valoración se actualizan en base al estado inmediatamente siguiente. Las trazas de elegibilidad seguirán siendo métodos de diferencias temporales, ya que las predicciones de harán en base a otras predicciones, aunque en este caso no serán las inmediatamente siguientes, sino las situadas n pasos después.

La implementación de las trazas de elegibilidad se hace asociando a cada estado una variable, su traza de elegibilidad. La traza de elegibilidad del estado s en el instante t se representa como $e_t(s) \in \Re^+$. Tras cada paso, la traza de elegibilidad para cada estado decae por $\gamma\lambda$, y la traza del estado visitado en ese paso se incrementa en 1:

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & \text{si } s \neq s_t; \\ \gamma\lambda e_{t-1}(s) + 1 & \text{si } s = s_t, \end{cases} \tag{2.23}$$

$\forall\, s \in S$, donde γ es el coeficiente de descuento y λ es el parámetro de decaimiento de la traza.

Lo que hacen las trazas de elegibilidad es indicar en cada momento qué estados se visitaron recientemente, para que únicamente se actualicen esos estados, y de manera proporcional al tiempo que hace que se visitaron. De esta manera, la actualización de $V_t(s)$ es:

$$\Delta V_t(s) = \alpha\,\delta_t e_t(s), \tag{2.24}$$

Algoritmo 2.6 Algoritmo *Watkins's* $Q(\lambda)$

$Q(s,a) \leftarrow$ valores iniciales aleatorios
$e(s,a) \leftarrow 0$
repetir
  Inicializar s,a
  repetir
    Ejecutar acción a, observar r, s'
    Seleccionar a' para s' a partir de una política derivada de Q (p.ej. egoísta)
    $a^* \leftarrow \arg\max_b Q(s',b)$
    $\delta \leftarrow r + \gamma Q(s',a^*) - Q(s,a)$
    $e(s,a) \leftarrow e(s,a) + 1$
    para todo s,a **hacer**
      $Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
      si $a' = a^*$ **entonces**
        $e(s,a) \leftarrow \gamma\lambda e(s,a)$
      si no
        $e(s,a) \leftarrow 0$
      fin si
    fin para
  hasta que Fin del episodio
hasta que Convergencia alcanzada

donde

$$\delta_t = r_t + \gamma V_t(s_{t+1}) - V_t(s_t). \tag{2.25}$$

En el caso del *Q-learning* la aplicación de las trazas de elegibilidad hace que la actualización de los Q-valores sea de la forma:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t e_t(s,a), \tag{2.26}$$

donde

$$\delta_t = r_t + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t). \tag{2.27}$$

Esta actualización se realiza tras cada ciclo para todos los pares estado-acción.

Existen diferencias en la forma de actualizar las trazas. El algoritmo *Watkins's* $Q(\lambda)$ [135] hace una distinción clara entre exploración y explotación en la actualización de la traza asociada a un par estado-acción visitado. En esta actualización si la acción es la acción egoísta para ese estado la traza decae por $\gamma\lambda$, mientras que si es una acción exploratoria la traza se pone a 0.

Esta actualización, aunque garantiza la convergencia del algoritmo, provoca que el aprendizaje sea más lento. Al no actualizar las acciones exploratorias su rendimiento es sólo ligeramente superior al de Q(0). Una opción es no poner a 0 las trazas de las acciones exploratorias (ecuación 2.23). A esta versión del algoritmo se le conoce como *Naive* $Q(\lambda)$ [123].

La distinción a la hora de aplicar las trazas de elegibilidad viene dada por diferentes puntos de vista a la hora de retropropagar el error. En el caso de *Watkins's* $Q(\lambda)$ no se desea que una acción exploratoria fallida desestabilice las valoraciones de la política egoísta, es por ello que se reinician las trazas a 0 al encontrarse una acción exploratoria para impedir que el error se propague. En cambio, en el algoritmo *Naive* $Q(\lambda)$ se desea aprovechar toda la información obtenida en cada experimento, aún a costa de modificar las valoraciones de la política egoísta.

2.4. Continuidad del espacio de estados y acciones

Uno de los primeros problemas a los que se enfrentó el aprendizaje por refuerzo fue el trabajar en entornos continuos. Muchas aplicaciones del mundo real necesitan trabajar con espacios de estados y/o acciones continuos.

Los algoritmos descritos anteriormente trabajan con conjuntos discretos de estados y acciones. La forma más común de representar los valores $V(s)$ o $Q(s,a)$ en este caso es mediante una representación tabular. Esto es factible si tanto el número de estados como el número de acciones son pequeños. A medida que crece el número de estados el tiempo de aprendizaje crece de manera exponencial, y cuantas más y más diferentes acciones disponga un robot más tiempo de exploración necesitará para encontrar la acción adecuada para cada estado.

Cuando el estado depende de un conjunto de variables continuas es necesario aplicar una estrategia que proyecte las medidas continuas en un conjunto discreto de estados. En la mayoría de los casos esta conversión se realiza en la clasificación de los estados. Por ejemplo, en [53] se usa una red de neuronas para clasificar las percepciones continuas en estados discretos, a continuación se aprende la función Q para cada acción en cada uno de dichos estados discretos. En [121] la clasificación se hace mediante *tile coding*, una técnica que consiste en dividir el espacio de entrada en regiones para tener un conjunto discreto de estados.

Aprendizaje mediante aproximación de funciones

Una manera muy común de enfrentarse a dominios continuos es utilizar algoritmos de aprendizaje por refuerzo con algún tipo de aproximador de funciones. Con este tipo de estra-

tegias se usará un conjunto de funciones parametrizadas que aproximarán o bien la función de valor o bien la propia política de control a ser aprendida. En estos algoritmos el aprendizaje consiste en el ajuste de los parámetros que representan dichas funciones, por lo tanto estos algoritmos trabajan directamente en el dominio continuo. Un ejemplo de esta estrategia son los algoritmos basados en el Regulador Lineal Cuadrático (LQR). En [17] tanto los estados como las acciones se representan mediante vectores de valores continuos, el algoritmo presentado aproxima los Q-valores calculando los elementos de la matriz que resuelve el sistema formado por la combinación de los vectores de estado y acción mediante un proceso recursivo de mínimos cuadrados.

No entramos en mayor detalle en la representación continua dado que los algoritmos presentados en esta tesis se aplican al caso de representaciones discretas. A continuación se muestran diversos algoritmos, algunos de los cuales aplican el aprendizaje por refuerzo basado en la utilización de aproximación de funciones.

2.5. Otros algoritmos de aprendizaje por refuerzo

Anteriormente se han descrito los algoritmos más clásicos, a continuación se muestran algunas otras propuestas más recientes que consideramos reseñables, tanto por su éxito y eficacia como por realizar una aproximación diferente a distintos problemas del aprendizaje por refuerzo. Estos algoritmos los hemos agrupado en algoritmos con exploración, algoritmos basados en mínimos cuadrados y algoritmos sobre espacios continuos.

2.5.1. Algoritmos con exploración

Exploración o Explotación Explícita

El algoritmo Exploración o Explotación Explícita (E^3 [61]) utiliza un modelo del entorno para resolver el problema de aprendizaje, pero no es necesario proporcionar un modelo *a priori* ya que a medida que avanza el aprendizaje y se explora el entorno el propio algoritmo construye el modelo. E^3 comienza aplicando una exploración balanceada, lo que significa ejecutar la acción que fue probada un menor número de veces para el estado actual, y mediante esta exploración obtiene información sobre el refuerzo y las transiciones de los estados que visita. Al cabo de un tiempo algunos estados irán siendo más conocidos y el modelo que se cree de ellos se parecerá bastante al modelo real por lo que se podrá usar dicho modelo para escoger una mejor estrategia de exploración o explotación. En caso de que el refuerzo

esperado sea mayor que un umbral se explotará la mejor acción de dicho estado, en caso contrario ejecutará una acción exploratoria, entendiendo como tal una acción que tenga una probabilidad significativa de que tras su ejecución se transite a un estado poco conocido.

Con este algoritmo se obtiene una política cuasi-óptima en tiempo polinómico respecto al número de estados. Los autores no proporcionan una implementación del algoritmo ni muestran resultados experimentales. Como trabajo futuro se plantean el desarrollo de un algoritmo que no requiera la creación de un modelo y el trabajar en técnicas para poder manejar grandes espacios de estados.

Algoritmo de Expertos Exploración-Explotación

Los algoritmos de expertos se basan en una combinación de las recomendaciones proporcionadas por diferentes expertos. Trasladado a nuestro ámbito, un experto proporciona una política de control adquirida anteriormente mediante cualquier estrategia. La propuesta de [39] se centra en encontrar la mejor política de todas las posibles, pero siempre dentro del conjunto de políticas propuestas por los expertos. Esto tiene dos consecuencias, por un lado hace que el aprendizaje no sea desde cero, ya que se parte de un conjunto de políticas obtenidas *a priori*. Por otro lado el refuerzo obtenido siguiendo este método no será mayor que el proporcionado por el mejor experto, lo que impide aprender la tarea si ninguno de los expertos es capaz de ejecutarla correctamente.

Cada vez que se sigue a un experto se actualiza su valoración mediante:

$$M_e = M_e + \frac{n}{S_e}(\tilde{R} - M_e), \tag{2.28}$$

donde S_e es el número de ciclos que se siguió al experto e, y $\tilde{R}$ es el refuerzo medio acumulado durante la fase actual de n ciclos. Al cabo de suficientes pruebas se tendrá una aproximación de la valoración de los expertos M_e. Hay que tener en cuenta que una vez finalizado el proceso de evaluación la importancia de cada experto no será modificada.

El algoritmo ejecutará una fase de exploración con probabilidad p_i y una fase de explotación con probabilidad $1 - p_i$. En este caso la exploración es la elección de un experto aleatorio sin tener en cuenta su valoración. Por otra parte, una fase de explotación consiste en seguir las indicaciones del experto mejor valorado. En el trabajo original [39] los autores presentan diversas opciones, y sus consecuencias teóricas, para el ajuste del valor de p_i, siendo algunas de ellas el aplicar un valor constante, o un decaimiento exponencial de p_i.

2.5.2. Algoritmos basados en mínimos cuadrados

Diferencias Temporales de Mínimos Cuadrados

Este algoritmo, conocido como LSTD [18], surge como un intento por mejorar la eficiencia en el uso de la información al aplicar métodos de mínimos cuadrados a un problema de aprendizaje de diferencias temporales. A diferencia de otros algoritmos, LSTD no proporciona un método para la mejora de política, sino que únicamente proporciona una valoración de la política de control que está siendo ejecutada. Por lo tanto, este algoritmo presenta la desventaja de que no tiene una aplicación directa para problemas donde se debe aprender una buena política de control para ejecutar una tarea.

El algoritmo LSTD aprende la función de valor del estado, V^{π}, mediante la aproximación de un vector de parámetros θ^*. Dicho vector permite calcular la valoración para un estado x como $V(x) = \phi_x^T \theta^*$, donde ϕ_x es el vector de características que representa al estado x.

La ecuación para calcular el vector θ en un instante t es:

$$\theta_t = \left[\frac{1}{t} \sum_{k=1}^{t} \phi_k (\phi_k - \gamma \phi_{k+1})^T \right]^{-1} \left[\frac{1}{t} \sum_{k=1}^{t} \phi_k r_k \right]. \tag{2.29}$$

Iteración de Política de Mínimos cuadrados

El algoritmo LSPI [75] es una evolución del algoritmo LSTD presentado anteriormente. Este algoritmo permite aprender los valores estado-acción y mejorar una política dentro de un esquema de iteración de política. LSPI se basa en el algoritmo LSTDQ, que, a su vez, es una mejora de LSTD. LSTDQ tiene el objetivo de aproximar la función de valor estado-acción, esta información es usada por LSPI para obtener una nueva y mejor política.

Este algoritmo considera la función Q como una combinación lineal de k funciones base ϕ que son ponderadas por los parámetros w:

$$\widehat{Q}^{\pi}(s,a;w) = \sum_{j=1}^{k} \phi_j(s,a) w_j. \tag{2.30}$$

Dichas funciones base son comúnmente polinomios o funciones de base radial (Gaussianas con una media y varianza fijas).

Se define $\phi(s,a)$ como el vector columna de tamaño k donde cada entrada j es el valor de la función ϕ_j evaluado en (s,a):

$$\phi(s,a) = \begin{pmatrix} \phi_1(s,a) \\ \cdots \\ \phi_2(s,a) \\ \cdots \\ \phi_k(s,a) \end{pmatrix}$$

Ahora, $\widehat{Q}^{\pi}$ puede expresarse de manera compacta como $\widehat{Q}^{\pi} = \Phi w^{\pi}$, donde w^{π} es un vector columna de longitud k que contiene los parámetros y Φ es una matriz $(|S| \cdot |A| \times k)$ de la forma

$$\Phi = \begin{pmatrix} \phi(s_1,a_1)^T \\ \cdots \\ \phi(s,a)^T \\ \cdots \\ \phi(s_{|S|},a_{|A|})^T \end{pmatrix}$$

Cada fila de Φ contiene el valor de todas las funciones base para un cierto par (s,a), y cada columna contiene el valor de una función base para todos los pares (s,a).

Asumiendo que las k funciones base son linealmente independientes, el problema de aprender los parámetros w^{π} de $\widehat{Q}^{\pi} = \Phi w^{\pi}$ se resuelve resolviendo el sistema lineal de $(k \times k)$ ecuaciones:

$$\mathbf{A} w^{\pi} = b,$$

donde $\mathbf{A} = \Phi^T \Delta_{\mu} (\Phi - \gamma \mathscr{P} \Pi_{\pi} \Phi)$ y $b = \Phi^T \Delta_{\mu} \mathscr{R}$, siendo Δ_{μ} la matriz diagonal que representa la importancia de la aproximación del error para cada par estado-acción, y Π_{π} la matriz que describe la política π. No se entrará en detalle sobre $\mathbf{A}$ y b, pero dado que contienen las matrices $\mathscr{P}$ y $\mathscr{R}$ es obvio que se necesita un modelo del sistema. Sin embargo, existe una solución iterativa basada en la interacción con el entorno que evita la necesidad de crear y almacenar de forma explícita un modelo (algoritmos 2.7 y 2.8).

Dado un conjunto de ejemplos D de la forma (s,a,r,s'), el número de funciones base k, el conjunto de funciones base ϕ, el coeficiente de descuento γ, y una política π, el algoritmo 2.7 presenta el proceso para calcular una aproximación de los parámetros w.

El algoritmo 2.7 permite aprender la función de valor Q^{π} de los pares estado-acción. Con dicho algoritmo se puede realizar un proceso de iteración de política que encuentre una

Algoritmo 2.7 Algoritmo LSTDQ

$\tilde{\mathbf{A}} \leftarrow 0$ // matriz $(k \times k)$
$\tilde{b} \leftarrow 0$ // matriz $(k \times 1)$
para cada $(s,a,r,s') \in D$ **hacer**
  $\tilde{\mathbf{A}} \leftarrow \tilde{\mathbf{A}} + \phi(s,a)\left(\phi(s,a) - \gamma\phi(s',\pi(s'))\right)^T$
  $\tilde{b} \leftarrow \tilde{b} + \phi(s,a)r$
fin para
$\tilde{w}^\pi \leftarrow \tilde{\mathbf{A}}^{-1}\tilde{b}$
devolver $\tilde{w}^\pi$

Algoritmo 2.8 Algoritmo LSPI

$w' \leftarrow w_0$
repetir
  $w \leftarrow w'$
  $w' \leftarrow \text{LSTD}Q(D,k,\phi,\gamma,w)$
hasta que $(\|w - w'\| < \varepsilon)$
devolver w

aproximación de los parámetros capaz de resolver la tarea. Esta iteración de política da lugar al algoritmo LSPI, mostrado en el algoritmo 2.8. Este algoritmo proporciona un vector w a partir del cual se puede obtener la política egoísta como:

$$\pi(s) = \arg\max_a Q(s,a) = \arg\max_a \phi(s,a)^T w. \tag{2.31}$$

En [75] se muestran resultados para tres tareas en simulación donde el algoritmo muestra un buen rendimiento. Los autores indican que una de las principales desventajas de su método es que si los ejemplos no son suficientemente representativos no se logra un buen aprendizaje, aunque esto es un problema común a todos los algoritmos de aprendizaje por refuerzo. Otro problema es que las funciones de valor pueden favorecer aquellos pares estado-acción más visitados dependiendo de la distribución de los ejemplos. Por último, el rendimiento del algoritmo depende de la elección de las funciones base, algo que debe ser decidido por un experto y que depende del problema a tratar.

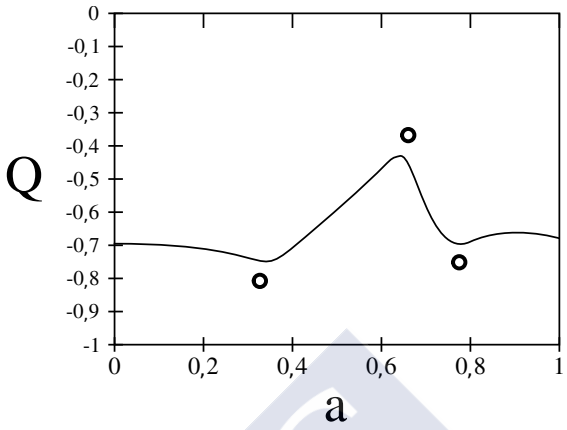


Figura 2.2: Ejemplo de *wire-fitting*. En esta imagen se muestra cómo los Q-valores de una acción unidimensional a son representados mediante una función continua definida mediante tres cables, mostrados como ∘.

2.5.3. Algoritmos sobre espacios continuos

Advantage Learning

En [41] se presenta una variación del *Q-learning* con un mejor rendimiento y capaz de aprender en espacios de estados y acciones continuos. Para aproximar la función Q se utiliza una combinación de una red de neuronas *feedforward* y un interpolador. La entrada de la red de neuronas artificiales será la percepción actual y como salida proporciona un conjunto de acciones y su valoración en la forma de pares acción-valoración $[\mathbf{u}, q]$, donde $\mathbf{u}$ es el vector de acción (la acción en cada una de las dimensiones), y q es la valoración. Esta red de neuronas se ocupa de generalizar entre estados similares, sin embargo, para resolver el problema de la continuidad de las acciones se utiliza el interpolador anteriormente mencionado. En este caso el interpolador es una función de *wire-fitting* [10]. Esta función representa los valores de los pares estado-acción como una superficie multidimensional interpolando entre una serie de puntos, llamados cables. En este caso los cables son los pares acción-valoración proporcionados por la red neuronal. La figura 2.2 muestra un ejemplo en el que una acción unidimensional es interpolada mediante 3 cables. La función de *wire-fitting* interpola entre los cables para permitir una representación continua de la función de valor Q que abarque todo el espacio de acciones.

El algoritmo de aprendizaje consta de los siguientes pasos:

1. Para un estado dado se ejecuta la acción con mayor valoración q de entre las acciones proporcionadas por la red neuronal. A continuación se observa el resultado.

2. Se calcula una nueva estimación de Q para la acción ejecutada a partir del valor actual, el refuerzo y la valoración del estado siguiente. Para ello se sigue la función de actualización de *Q-learning*.

3. La nueva estimación de la función de valor implica modificar la curva o superficie que representa la función Q para todo el espacio de acciones (figura 2.2). Esto supone realizar nuevos cálculos para u y q de cada cable con el fin de aproximar correctamente la curva. A continuación se realiza un entrenamiento de la red neuronal para que produzca como salidas los nuevos valores de u y q.

Esta propuesta mejora de manera significativa el rendimiento en comparación con *Q-learning*. Un problema que tiene al usar redes de neuronas como medio para generalizar los estados es que necesitan inyectar entradas artificialmente para no olvidar eventos pasados. De esta manera, ejemplos de estado-acción-estado son almacenados y se le vuelven a presentar a la red cada cierto tiempo.

Actor-Crítico Natural

Uno de los algoritmos de aprendizaje por refuerzo que más éxito está teniendo en los últimos años es NAC [90]. Como su nombre indica se basa en una arquitectura actor-crítico, por lo tanto es una estrategia de iteración de política. Para tratar el problema de la continuidad el algoritmo utiliza políticas parametrizadas y la actualización de los parámetros se hará en base al gradiente del refuerzo esperado de las políticas. El nombre de Actor-Crítico Natural proviene del uso del gradiente "natural" [5], esto es el ascenso más pronunciado respecto a la métrica de información de Fisher. Para aplicar este algoritmo se asume que se trabaja sobre un Proceso de Decisión de Markov, aunque los autores afirman que se puede aplicar en problemas con información parcial de los estados.

Se presentan dos versiones del algoritmo NAC:

– NAC con LSTD-Q(λ): utilizan una versión mejorada de LSTD para aproximar las funciones de valor. Necesita añadir al conjunto de funciones base unas nuevas funciones

derivadas a partir del gradiente natural. Dichas funciones pueden causar desviaciones en el aprendizaje.

– eNAC: para mejorar el problema de la desviación de la versión anterior, se presenta una versión episódica donde la única función base a mayores es una constante. Al ser un algoritmo episódico se pueden aproximar las funciones de valor a partir de los resultados reales observados al final de cada episodio.

En [90] se presentan resultados tanto en simulación como con un robot real. En simulación el algoritmo se enfrenta a la tarea de sostener un péndulo invertido anclado a una base móvil, conocida como *Cart-Pole Balancing*. El sistema resuelve la tarea en 10 minutos, frente a las 2 horas que tarda el algoritmo analítico contra el que se compara. En este mismo trabajo también se muestran los resultados obtenidos al aplicar el algoritmo a un brazo robótico que debe aprender a batear una pelota. El brazo robótico tiene 7 grados de libertad. En un principio al robot se le enseña mediante aprendizaje supervisado, pero el comportamiento obtenido no es capaz de golpear la pelota. Este comportamiento es mejorado mediante el Actor-Crítico Natural (NAC) y se consigue que golpee la pelota tras unos cientos de episodios.

PoWER

J. Kober y J. Peters desarrollaron el algoritmo PoWER (*Policy learning by Weighting Exploration with the Returns* [66]) con la intención de crear un algoritmo para el aprendizaje de primitivas motoras en problemas con un gran número de dimensiones. Como otros algoritmos presentados anteriormente, PoWER representa las políticas como un conjunto de funciones base parametrizadas, y en este caso se busca optimizar los parámetros de dichas funciones mediante un método de Maximización de la Esperanza (EM) con el fin de obtener el mayor refuerzo. Un aspecto importante de este algoritmo es que se reutilizan los episodios pasados asignándoles a cada uno una importancia en función del resultado obtenido. Para ello representan la ponderación de una variable x en base a la importancia de un episodio $\tau = [s_{1:T+1}, a_{1:T}]$ de $T+1$ estados y T acciones como $\langle x \rangle_{w(\tau)}$. Aquellos episodios de baja importancia se descartan, y el resto serán utilizados para mejorar la política de control ponderando los episodios en función de su importancia. El algoritmo 2.9 muestra el proceso de aprendizaje de los parámetros θ.

Los autores muestran resultados tanto en simulación como con un robot real. En simulación comparan su algoritmo con otras propuestas y PoWER se muestra mejor que todos los

Algoritmo 2.9 Algoritmo PoWER

Dados un conjunto de funciones base ϕ, sus parámetros θ y una función de exploración ε
repetir
Toma de datos: ejecutar episodio(s) utilizando $a = (\theta + \varepsilon_t)^T \phi(s)$ con $[\varepsilon_t]_{ij} \sim \mathcal{N}(0, \sigma_{ij}^2)$ y almacenar las tuplas $(t, s_t, a_t, s_{t+1}, \varepsilon_t, r_t)$ para $t = \{1, 2, ...T+1\}$.
Estimación: $\hat{Q}^\pi(s,a,t) = \sum_{\tilde{t}=t}^{T} r(s_{\tilde{t}}, a_{\tilde{t}}, s_{\tilde{t}+1}, \tilde{t})$.
Ponderación: calcular un conjunto de pesos que ponderen la importancia de los episodios descartando los de poca importancia.
Actualización política: $\theta_{k+1} = \theta_k + \frac{\langle \sum_{t=1}^{T} \varepsilon_t Q^\pi(s,a,t) \rangle_{w(\tau)}}{\langle \sum_{t=1}^{T} Q^\pi(s,a,t) \rangle_{w(\tau)}}$
hasta que $\theta_{k+1} \approx \theta_k$

demás algoritmos. Como resultados en un entorno real muestran cómo un brazo robótico con siete grados de libertad aprende a jugar al boliche. Dado que la tarea es extremadamente compleja para ser aprendida sin ningún conocimiento previo, como paso previo al aprendizaje por refuerzo se realiza un aprendizaje por demostración para inicializar las primitivas de control. Tras la fase de aprendizaje por demostración los autores aplican el algoritmo PoWER y tras unos 75 intentos el robot es capaz de completar el juego con éxito.

2.6. Formulación del refuerzo

Todos los algoritmos mostrados en este capítulo dependen de una señal de refuerzo que indique cuándo el robot está ejecutando el comportamiento de manera correcta. El diseño e implementación de esta señal de refuerzo es una tarea crucial para conseguir que el robot alcance el comportamiento deseado. Por lo tanto, no se puede considerar que el proceso de aprendizaje es totalmente autónomo, y no es factible pedir a personas no expertas en robótica que especifiquen adecuadamente el conjunto de reglas necesarias para indicar cuándo el robot lo está haciendo bien o mal. A continuación se exponen brevemente dos alternativas para la obtención automática de la señal refuerzo: el aprendizaje por refuerzo inverso y la obtención de la señal de refuerzo a partir de la observación del comportamiento humano.

Aprendizaje por refuerzo inverso

El problema del aprendizaje por refuerzo inverso es obtener la función de refuerzo a partir de la observación de un agente ejecutando un comportamiento [86]. La motivación del apren-

dizaje por refuerzo inverso tiene dos vertientes. Por un lado se ideó como un método útil a la hora de obtener la función siendo optimizada por un sistema natural (p.ej. la búsqueda de néctar por parte de abejas). Por otro lado se vio su utilidad para el aprendizaje en robótica, ya que si se dispone de un agente o demostrador capaz de ejecutar correctamente el comportamiento es posible extraer la función de refuerzo para aplicar ese refuerzo en nuevos aprendizajes.

El objetivo del aprendizaje por refuerzo inverso es obtener una función de refuerzo R que maximice el refuerzo obtenido por la política óptima. La función de refuerzo será de la forma

$$R(s) = w_1\phi_1(s) + w_2\phi_2(s) + \cdots + w_N\phi_N(s), \tag{2.32}$$

donde ϕ_n es una función base que representa una componente del refuerzo y w_n realiza la ponderación de cada una de las funciones. Por lo tanto, lo que se desea obtener es el vector $w \in \mathbb{R}^N$. Para ello se debe optimizar la siguiente ecuación:

$$\max_w \sum_{k=1}^{K} f\left(V_w^{\pi^*}(s_0) - V_w^{\pi_k}(s_0)\right). \tag{2.33}$$

El proceso de obtención de w consiste en observar los episodios del agente demostrador ejecutando la política óptima. A partir de esas observaciones se obtiene una aproximación de la función de valor para el estado inicial. Entonces se genera y ejecuta una política y se observa su resultado, a continuación se modifican los pesos w de manera que esta nueva política obtenga una menor valoración que la política óptima. Iterando entre el proceso de generación y valoración de políticas y el proceso de modificación del refuerzo se alcanza una función R que permite aprender una política de valor similar a la óptima.

Obtención del refuerzo observando el comportamiento humano

Como paso previo a la investigación en la mejora del aprendizaje por refuerzo presentada en esta tesis, hemos desarrollado un sistema de obtención automática de refuerzo a partir de la observación del comportamiento humano [100, 103]. Mediante cámaras situadas en el entorno se observó el movimiento de las personas y se establecieron cuáles eran las trayectorias que la gente suele seguir. Con esta información se generó de forma automática una señal de refuerzo que simplemente discrimina cuándo el robot se está moviendo por las mismas áreas por donde se mueve la gente.

El sistema que se implementó consta de cuatro etapas (figura 2.3): detección de movimiento, detección y seguimiento de las personas, posicionamiento en el entorno y segmentación

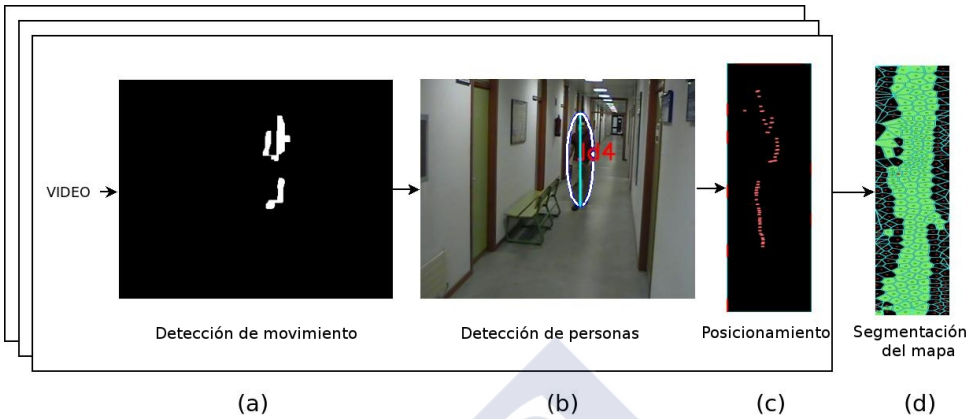


Figura 2.3: Etapas para la generación automática de refuerzo: a) detección de movimiento; b) detección y seguimiento de las personas; c) posicionamiento en el entorno; y d) segmentación del entorno en regiones de Voronoi. Las regiones verdes representan zonas transitables, las negras representan las zonas donde el robot recibe refuerzo negativo.

del entorno. Las etapas de detección de movimiento y seguimiento de personas se pueden realizar partir de cualquiera de las numerosas técnicas existentes [87], en nuestro caso se utilizó la detección de movimiento por diferencia con una imagen de fondo. Para la detección de personas se seleccionaron aquellos fragmentos de la imagen en movimiento que se ajustaban a las proporciones de una persona.

La fase de posicionamiento consistió en calcular la trayectoria de cada persona sobre un mapa del entorno. Como la posición de cada cámara en el entorno era conocida se calibró el sistema para determinar la transformación proyectiva que asocia cada punto del suelo en la imagen con la posición real en el entorno [47]. Para calcular la matriz de transformación sólo es necesario conocer la posición real de cuatro puntos que pertenezcan al plano del suelo y asociarlos con sus respectivas coordenadas en píxeles de la imagen capturada.

Después de un período de observación se obtuvo un conjunto significativo de trayectorias de las personas durante sus quehaceres cotidianos. La última fase de nuestro esquema consistió en la traducción automática de dicha información en un refuerzo válido para ser usado en técnicas de aprendizaje por refuerzo. El objetivo es categorizar todas las posiciones (aunque realmente debería realizarse a nivel de "trayectorias") obtenidas y determinar cuáles son las áreas del entorno más comúnmente utilizadas. Para esta clasificación se han utilizado redes neuronales artificiales del tipo LVQ [69, 68] con el objetivo de conseguir una adecuada ge-

neralización de los datos. La salida es una segmentación del entorno en regiones de Voronoi, cada una de las cuales indica el refuerzo obtenido por el robot al posicionarse dentro de ella.

Se realizaron pruebas experimentales en el Departamento de Electrónica e Computación de la Universidade de Santiago de Compostela. Con los datos obtenidos tras la observación de personas durante un período de 10 minutos se generó una señal de refuerzo que se utilizó para realizar un aprendizaje en simulación. Tras el aprendizaje se trasladó la política de control obtenida a un robot real, el cual logró moverse por el entorno de manera satisfactoria.

Si bien este trabajo es una contribución neta de la tesis, representa un trabajo inicial que no utiliza los algoritmos que se describirán en el resto de los capítulos de esta tesis. En los aprendizajes mostrados en [100, 103] se utilizó una estrategia de aprendizaje basada en la combinación del aprendizaje por refuerzo y algoritmos genéticos.

2.7. Discusión

En este capítulo hemos presentado los fundamentos del aprendizaje por refuerzo. Se ha visto la importancia de las funciones de valor, las cuales permiten estimar el refuerzo futuro que se obtendrá tras la ejecución de una acción o una política en un estado determinado. Se han presentado diversos algoritmos tanto clásicos como más novedosos, haciendo una distinción entre aquellos que construyen un modelo y los que se basan únicamente en la experiencia. El objetivo de esta tesis es desarrollar un sistema de aprendizaje apto para su aplicación en robots, y en particular en el ámbito de la robótica de servicio, ya que creemos que es especialmente interesante por el tipo de trabajos que realizan y los usuarios que los manejan. Para ello será necesario resolver los siguientes problemas que hemos identificado en el aprendizaje por refuerzo:

Lentitud

El mayor problema del aprendizaje por refuerzo es que puede ser muy lento [20]. Se necesitan muchas iteraciones para tener unas valoraciones fiables de cada uno de los estados y acciones posibles. El aprendizaje por refuerzo sufre la llamada maldición de la dimensionalidad: el tiempo de aprendizaje crece exponencialmente con el número de estados. Esto hace que muchos algoritmos sean en la práctica imposibles de aplicar a un robot real. En un robot real cada iteración tiene un alto coste de tiempo y energía, por lo tanto hay que conseguir algoritmos capaces de aprender en un número reducido de iteraciones (es decir, ejemplos o

pruebas). Con el fin de acelerar el proceso de aprendizaje se ha propuesto la combinación del aprendizaje con refuerzo con diferentes técnicas como la programación genética [60] o heurísticas basadas en casos [28].

Problemas de generalización

Lo comentado anteriormente nos lleva a uno de los problemas clásicos del aprendizaje máquina, el dilema *bias-varianza*. Esto es el cómo de bien se ajusta nuestro sistema al conjunto de ejemplos mientras es capaz de generalizar y mostrar un comportamiento correcto ante situaciones diferentes de aquellas que ha visto previamente. El *bias* representa el nivel al que el sistema se ajusta al conjunto de entrenamiento, un bajo *bias* significa que el sistema se ajusta muy bien a estos datos. La *varianza* es la medida de cómo de sensible es el sistema a los datos con los que fue obtenido, ¿se habría obtenido el mismo controlador con un conjunto de datos diferente? Por lo general una baja *varianza* implica una buena generalización.

Si, como se dijo anteriormente, el robot debe aprender de muy pocos ejemplos es probable que se obtenga un sistema que se ajuste bien a esos pocos datos (bajo *bias*) pero que no sea capaz de generalizar a nuevas situaciones (alta *varianza*). Generalmente hay un compromiso entre el *bias* y la *varianza*: si se intenta disminuir uno el otro aumentará.

Estrategias multiparamétricas

En esta tesis asumimos que dado que el robot debe aprender, se cometerán fallos durante el proceso de aprendizaje y puede que sea necesario realizar algún ajuste de los parámetros del algoritmo de aprendizaje. Esta es una tarea que un usuario no experto en robótica no puede hacer, ya que no tiene por qué poseer los conocimientos necesarios para ajustar el algoritmo a las circunstancias particulares del entorno. Es por esta razón que los algoritmos de aprendizaje clásicos son difíciles de usar para quien no tiene una formación adecuada. Son algoritmos que dependen de múltiples parámetros y donde la modificación de sus valores no es algo trivial. Sin embargo, tal y como ya se ha comentado, en la robótica de servicios se necesitan algoritmos capaces de aprender en un gran número de situaciones o tareas diferentes pero aplicando el mínimo número de ajustes a sus parámetros.

Interpretabilidad de lo aprendido

Otro aspecto que tiene que ver con su facilidad de uso es el poder interpretar lo que el algoritmo conoce hasta el momento. Las funciones de valor almacenan un valor numérico que representa el refuerzo descontado esperado, algo que aunque permite diferenciar qué acciones o estados son mejores, hace complicado el conocer la calidad absoluta de una acción. Por ejemplo, una valoración para un par estado-acción de -0.9 prácticamente nos asegura que el robot recibirá refuerzo negativo tras la ejecución de dicha acción, pero no indica cuándo. Un algoritmo donde la valoración de las acciones sea más interpretable ayudará a monitorizar el proceso de aprendizaje permitiendo detectar fallos durante la evolución del mismo.

Dilema exploración/explotación

Uno de los retos que aparece en el aprendizaje por refuerzo, y no en otros tipos de aprendizaje, es el compromiso entre explotación y exploración. Para maximizar el refuerzo recibido el agente debe tender a tomar aquellas acciones que ya probó en el pasado y que se mostraron efectivas para incrementar el refuerzo. Sin embargo, para descubrir las mejores acciones para cada estado o situación el agente debe probar acciones distintas a las que ya ha seleccionado. Se está en una situación donde el agente debe explotar lo que ya conoce para maximizar el refuerzo, pero también debe explorar para seleccionar mejores acciones en el futuro. El dilema es encontrar un equilibrio correcto entre la explotación y la exploración, ya que siguiendo una estrategia de exploración pura o explotación pura lo más probable es que fracase. Por lo tanto, el agente debe probar una variedad de acciones y progresivamente ir favoreciendo aquellas que muestran mejores resultados.

Dilema plasticidad/estabilidad

Las aplicaciones prácticas de los algoritmos de aprendizaje por refuerzo generalmente dividen el ciclo de vida del sistema en dos fases diferenciadas. La primera es el *proceso de aprendizaje*, donde el robot adquiere un controlador aplicando uno de los muchos algoritmos existentes. La segunda fase sería la fase de *puesta en uso*, donde el robot estará operando en su entorno aplicando la política de control aprendida durante la fase de aprendizaje. Esta diferenciación entre etapas hace que el comportamiento del robot no sea tan adaptable como sería deseable, ya que para aprender nuevas situaciones se necesitaría volver a la fase de aprendizaje. El problema viene dado por el hecho de que si se permite continuar el aprendizaje

ante circunstancias nuevas, el sistema puede incurrir en inestabilidades y olvidar parte de lo aprendido anteriormente. En un entorno de robótica de servicio personal esto es inasumible, por lo que se necesitan sistemas capaces de inyectar nuevo conocimiento al robot sin que esto signifique una desestabilización de lo ya aprendido. El robot debe ser capaz de aprender en cualquier momento.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Capítulo 3

Incrementando el intervalo de tiempo antes del fallo

Al final del capítulo anterior se mostraron los principales retos a los que se enfrenta el aprendizaje por refuerzo. Si deseamos tener un robot que disponga de la capacidad de aprender en entornos cotidianos y sin la supervisión de un experto en robótica será necesario hacer frente a dichos retos. En este capítulo se proponen soluciones a algunos de ellos.

En primer lugar se reformulará la función objetivo del aprendizaje por refuerzo, lo que proporcionará una mayor interpretabilidad al aprendizaje. La segunda aportación de este capítulo es un nuevo algoritmo de aprendizaje que implementa esta nueva función objetivo y que reduce el número de parámetros necesarios.

La primera pregunta que tratamos de responder en este capítulo es: ¿hay alguna manera clara de entender lo que el robot aprende tras cada interacción con el entorno? De manera similar a lo que ocurre con la mayoría de las redes de neuronas artificiales, durante el aprendizaje por refuerzo la mayor parte de los algoritmos generan una gran cantidad de números que no muestran de manera clara qué está aprendiendo el robot. Estos números tratan de estimar el refuerzo futuro que el robot espera recibir:

$$E\left\{\sum_{t=0}^{\infty}\gamma^t r_t\right\}$$

Dicho valor, aunque proporciona una valoración de los estados perfectamente válida para los algoritmos de aprendizaje, es de difícil interpretación por parte de una persona. Sabemos que una acción es mejor que otra si se espera recibir un mayor refuerzo ejecutando esa acción, pero

no sabemos en términos absolutos qué significa esa valoración. ¿Cuándo recibirá refuerzo el robot? ¿Esta acción me garantiza un error inmediato? En este capítulo describiremos una nueva función objetivo mediante la cual se puede predecir el tiempo que transcurrirá hasta que el robot cometa un error.

A continuación presentamos un nuevo algoritmo de aprendizaje, capaz de iterar las políticas de control tratando de incrementar ese tiempo hasta el fallo hasta lograr una política que ejecute la tarea de manera satisfactoria. Este nuevo algoritmo de aprendizaje por refuerzo, inspirado en algoritmos clásicos como *Q-learning*, pretende proporcionar aprendizajes rápidos que no necesiten un ajuste fino de parámetros y que permitan interpretar intuitivamente lo que el robot está aprendiendo. Llamaremos al algoritmo *I_Tbf* a partir del acrónimo de *Increasing the time before failure* (Incrementando el tiempo antes del fallo). Esta nueva propuesta, que no necesita un modelo del entorno, empleará un único parámetro cuyo valor debe precisar el usuario. Ahora bien, tal y como se podrá ver a través de la aplicación experimental de la propuesta, no será necesario un ajuste fino de este parámetro para que se produzca un aprendizaje correcto. También se planteará una solución propia al dilema de la exploración-explotación, típico de los aprendizajes por refuerzo.

De manera resumida, las mejoras que proporciona el algoritmo que se presenta en este capítulo son: un aprendizaje rápido, la práctica ausencia de parámetros y la facilidad para interpretar lo que el robot está aprendiendo.

3.1. Prediciendo el tiempo antes del fallo

Con el fin de obtener un algoritmo más interpretable se hará una redefinición de la función objetivo del aprendizaje por refuerzo. Recordemos que en los algoritmos clásicos de aprendizaje por refuerzo la función objetivo que se pretende maximizar es el refuerzo que se espera recibir en el futuro. Se cambiará dicha función y, en lugar del refuerzo esperado, se maximizará el tiempo transcurrido antes de la comisión de un error:

$$E\left\{-e^{(-Tbf(s=s_0,a=a_0)/50T)}\right\}, \tag{3.1}$$

donde $Tbf(s_0,a_0)$ (de *Time before failure*, Tiempo antes de fallo) representa el tiempo esperado antes de que el robot cometa un fallo (expresado en segundos), ejecutando la acción a_0 en el estado s_0 y siguiendo a continuación la política egoísta. T es el período de control del robot (expresado en segundos). Al multiplicar el período de control por 50 se consigue una

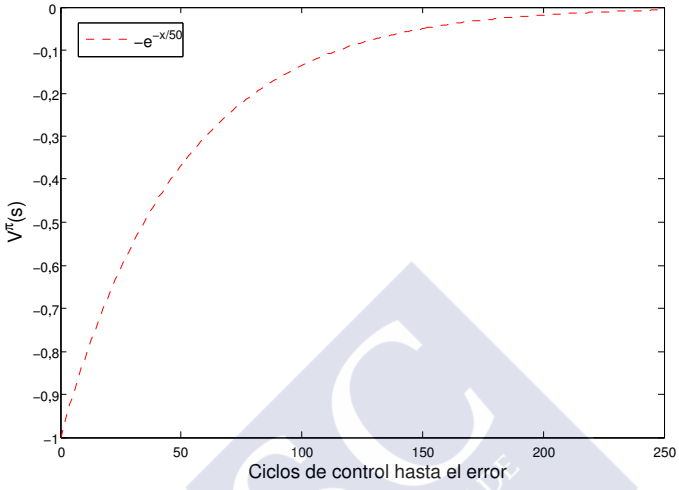


Figura 3.1: Evolución de la función de valor respecto al tiempo estimado de fallo. Los valores del eje x representan *Tbf/T*, donde *Tbf* es el tiempo antes del fallo (en segundos) y T es el período de control del robot (en segundos).

función con una pendiente alta pero que converge a 0 conforme el tiempo antes del fallo se incrementa de forma significativa (figura 3.1).

Esta nueva función objetivo (ecuación 3.1) proporciona una información mucho más interpretable, dado que la valoración de un estado o un par estado-acción se podrá traducir directamente al tiempo que se espera que transcurra hasta que ocurra un error. Por ejemplo, atendiendo a la figura 3.1, la cual representa la función $-e^{-x/50T}$ utilizada para aproximar la valoración del tiempo antes del fallo, se puede afirmar que si se conoce que el valor de un estado es $0,4$ se espera que el robot cometa un error tras aproximadamente 50 ciclos de control, que, con un período de control de 200 ms se traduciría en 10 segundos hasta el fallo.

Este cambio de la función objetivo implica que se deben reescribir los algoritmos iterativos que calculan la valoración de los estados o de los pares estado-acción. Tal y como se vio en el capítulo 2, existen dos funciones de valoración V y Q, las cuales habrá que modificar para adecuarlas a la nueva función objetivo. A continuación se muestran las nuevas ecuaciones adaptadas para el cálculo del tiempo antes del fallo.

Asumamos que tenemos un robot moviéndose por el entorno. Este robot se estará moviendo siguiendo una política π, que determina para cada posible estado qué acción debe ejecutar el robot:

$$\begin{aligned} \pi : S &\rightarrow A \\ s \in S &\rightarrow \pi(s) \in A, \end{aligned} \tag{3.2}$$

donde S es el conjunto de estados mediante los cuales el robot representa el entorno, y A es el conjunto finito de todas las acciones posibles. Asumamos también que podemos conocer si el robot está actuando de la manera correcta o no, esto es, existe una función de refuerzo que penaliza al robot cuando muestra un comportamiento incorrecto (refuerzo = -1), o se inhibe (refuerzo = 0) en otro caso.

En este contexto, definimos el valor de un estado $s \in S$, con respecto a la política π en función del tiempo esperado antes de que el robot haga algo mal (reciba refuerzo negativo), empezando en el estado s y siguiendo la política π como:

$$V^{\pi}(s) = E\left\{-e^{-Tbf^{\pi}(s)/50T}\right\}. \tag{3.3}$$

Según la ecuación 3.3 el término $-e^{-Tbf/50T}$ es una función continua que toma valores en el intervalo $[-1,0]$, y que varía suavemente a medida que el tiempo estimado de fallo aumenta (figura 3.1).

Por otra parte, la función Q, la función de valoración de los pares estado-acción, representa el tiempo esperado a fallo si en el estado s se ejecuta la acción a y a continuación se sigue la política egoísta:

$$Q(s,a) = E\left\{-e^{-Tbf(s,a)/50T}\right\}. \tag{3.4}$$

Es importante recordar que la política egoísta π^* selecciona la acción con el Q-valor máximo para cada estado:

$$\pi^*(s) = \arg\max_a \{Q(s,a)\}. \tag{3.5}$$

Dado que $V^{\pi}(s)$ y $Tbf^{\pi}(s)$ son desconocidos, habrá que utilizar sus estimaciones actuales $V_t^{\pi}(s)$ y $Tbf_t^{\pi}(s)$:

$$Tbf_t^{\pi}(s) = -50\ T\ Ln(-V_t^{\pi}(s)), \tag{3.6}$$

o, en el caso de $Q_t(s,a)$ y $Tbf_t(s,a)$:

$$Tbf_t(s,a) = -50\ T\ Ln(-Q_t(s,a)), \tag{3.7}$$

La definición de $Q(s,a)$, $Tbf(s,a)$ y la política egoísta determina la relación entre los Q-valores de dos estados consecutivos:

$$Tbf_t(s_t,a_t) = \begin{cases} T & \text{si } r_t < 0 \\ T + \max\limits_a \{Tbf_t(s_{t+1},a)\} & \text{en otro caso,} \end{cases} \tag{3.8}$$

donde r_t es el refuerzo que el robot recibe cuando ejecuta la acción a_t en el estado s_t.

Si combinamos las ecuaciones 3.7 y 3.8, se puede afirmar que:

$$Q_{t+1}(s,a) = \begin{cases} -e^{-1/50} & \text{si } r_t < 0 \\ Q_t(s_t,a_t) + \delta & \text{en otro caso,} \end{cases} \tag{3.9}$$

donde,

$$\delta = e^{\frac{-1}{50}} \max_a \{Q(s_{t+1},a)\} - Q(s_t,a_t). \tag{3.10}$$

Siguiendo un razonamiento similar se puede obtener la relación entre las valoraciones de dos estados consecutivos:

$$\begin{aligned} V_t^\pi(s) &= -e^{-Tbf_t(s)/50T} \\ &= -e^{-(T+Tbf_t^\pi(s_{t+1}))/50T} \\ &= V_t^\pi(s_{t+1})\, e^{-1/50} \end{aligned} \tag{3.11}$$

3.2. Algoritmo *I_Tbf*

Utilizando las ecuaciones mostradas en la sección anterior se pueden reescribir los algoritmos clásicos de aprendizaje por refuerzo. De todas maneras, hemos decidido proponer un nuevo algoritmo de aprendizaje con el fin de minimizar el número de parámetros utilizado por el algoritmo.

Un proceso iterativo aplicando la ecuación 3.9 permitirá la obtención de la función de valor de una política de control π. Básicamente el robot empieza con un conjunto inicial de valores negativos aleatorios, $Q(s,a) \in [-0,7,\ -0,675]$, $\forall s \in S, a \in A$, y entonces se moverá interactuando con el entorno. En cada instante de tiempo el robot observa el estado actual del entorno s_t, y ejecuta la acción sugerida por la política, $\pi(s_t)$. Durante la exploración, el robot irá haciendo predicciones de la recompensa que recibirá, de manera que mediante las predicciones y el refuerzo que realmente recibe se actualizará el valor de $Q(s,a)$. La ecuación 3.9 describe la versión más simple de aprendizaje de la función de valor, ya que únicamente

actualiza los valores basándose en la transición entre dos estados consecutivos. Cuando los refuerzos negativos son poco frecuentes, puede que se necesiten muchas pruebas de aprendizaje para que esos valores se propaguen a los estados anteriores. Debido a esto, lo que llamamos *TD-learning* [123] intenta acelerar el proceso simplemente añadiendo más *memoria* al sistema. En nuestro caso se hace teniendo en cuenta toda la ejecución de la política π. Se ejecutará cada política hasta que el robot cometa un fallo o la ejecución alcance una duración límite, tras la cual se considera que el algoritmo ha convergido a una solución y el aprendizaje finaliza. Con esta información conocemos con exactitud cómo de bien resultó cada acción en el episodio actual y por lo tanto podemos actualizar la función de valor de acuerdo a lo ocurrido en el experimento.

El algoritmo I_Tbf consta de dos etapas (algoritmo 3.1): una primera fase durante la cual el robot se mueve por el entorno hasta que comete un fallo; y una segunda etapa consistente en la actualización de los Q-valores y que tiene lugar una vez ha finalizado el episodio. Dado que en un aprendizaje se trabaja con valores estocásticos y las funciones de valoración se van aproximando poco a poco, a la hora de actualizar los Q-valores de los pares estado acción se ha añadido un parámetro $\beta \in [0,1]$ que actúa como coeficiente de aprendizaje, y es el único parámetro que debe ser ajustado por el usuario.

3.3. Dilema exploración-explotación

A la hora de ejecutar un algoritmo de aprendizaje por refuerzo se debe alcanzar un equilibrio entre la exploración de nuevas acciones y la explotación de acciones conocidas. Para obtener un refuerzo alto, el robot debe ejecutar acciones que se probaron en el pasado y que proporcionaban un refuerzo alto, pero para encontrar dichas acciones el robot debe explorar acciones que no ejecutó anteriormente. Si hay excesiva exploración el algoritmo puede no converger o demorarse demasiado. Por otro lado, si no se explora lo suficiente y únicamente se explotan las acciones conocidas se dificulta la aparición de nuevas soluciones, lo que puede hacer que el algoritmo no converja o lo haga a una solución sub-óptima.

Existen diferentes estrategias para alcanzar un equilibrio entre exploración y explotación. Una de las más populares se conoce como $\varepsilon - greedy$. Con esta estrategia se ejecutará la acción egoísta (*greedy*) pero habrá una pequeña probabilidad ε de ejecutar una acción exploratoria escogida al azar de entre todas las posibles acciones. Por norma general se escoge $\varepsilon \leq 0,1$. El problema de esta estrategia es que en el caso de que se escoja una acción explo-

Algoritmo 3.1 Algoritmo de aprendizaje I_Tbf

Primera etapa: Recolectando información
$m = 0$
repetir
  Observar el estado actual, s_t: $s[m] = s_t$
  Seleccionar una acción a_t para s_t: $a[m] = a_t$
  Ejecutar acción a_t, observar el nuevo estado s_{t+1} y el valor de refuerzo r_t, $r[m] = r_t$
  Obtener *Tbf estimado*: $u[m] = -50\ T\ ln(-\max_a\{Q_t(s_{t+1},a)\})$
  si $r_t >= 0$ **entonces**
    $m \leftarrow m+1$
  fin si
hasta que $r_t < 0$ **o** $m >= m_{max}$

Segunda etapa: Actualizar los Q-valores
para $k = m-1, m-2, \cdots, 0$ **hacer**
  si $k = m-1$ **entonces**

$$Tbf = \begin{cases} T & \text{si } r[m-1] < 0 \\ u[m-1] & \text{en otro caso} \end{cases}$$

  si no
    $Tbf \leftarrow Tbf + T$
  fin si
  $\Delta Q_t(s[k],a[k]) = \beta\left(-e^{-Tbf/50T} - Q_t(s[k],a[k])\right)$
fin para

ratoria todas las acciones son equiprobables, sin que se tenga en cuenta que algunas acciones, pese a no ser la acción egoísta, pueden estar mucho mejor valoradas que otras.

La solución obvia al problema mencionado anteriormente es variar la probabilidad de escoger una acción en función de su valoración actual. De esta manera la acción egoísta seguirá teniendo la mayor probabilidad de ser escogida, pero el resto de acciones serán ordenadas en función de su calidad. En base a esta idea se propuso el método de selección de acciones *softmax*, el cual usa la distribución de Boltzmann para escoger las acciones:

$$P_t(s,a) = \frac{e^{Q_t(s,a)/\tau}}{\sum_{b=1}^{|A|} e^{Q_t(s,b)/\tau}}, \tag{3.12}$$

donde τ es un parámetro positivo llamado temperatura. Altas temperaturas causan que las acciones sean prácticamente equiprobables, lo que beneficia la exploración. Bajas temperaturas causan mayores diferencias en las probabilidades, favoreciendo la selección de acciones con mayores valoraciones. Cuando $\tau \to 0$ se seleccionará siempre la acción con mayor valoración. En muchas implementaciones de softmax inicialmente el valor de τ es alto y va disminuyendo a medida que pasa el tiempo. El problema de esta estrategia es que el parámetro τ implica un límite temporal a la exploración, por lo que si cuando $\tau = 0$ aún no se ha alcanzado una solución correcta la falta de exploración dificultará el aprendizaje.

Un parámetro de temperatura común para todos los estados provocará que si unos estados se visitan en numerosas ocasiones, aquellos estados que se visitan de manera menos frecuente no logren aprender. Por el contrario, si se permite que los estados con pocas visitas exploren en etapas avanzadas del aprendizaje, se provoca que los estados que deberían estar ya aprendidos exploren de manera innecesaria, lo que causará inestabilidades. Una solución a este problema es vincular un parámetro τ a cada estado. De esta manera, valores de temperatura independientes permiten un nivel de exploración diferente para cada estado.

Continuando nuestros esfuerzos por reducir el número de parámetros, en los experimentos que se describen en este capítulo hemos decidido aplicar un método similar al *softmax*, pero donde se elimina el parámetro de la temperatura. Gracias a la variación exponencial de los valores de Q debido a la nueva función objetivo (figura 3.1), es posibile eliminar el parámetro de temperatura y mantener la exploración en aquellos estados sin una solución conocida mientras los estados ya aprendidos son explotados. La probabilidad de elegir una acción dependerá de la ponderación de su valoración respecto a todas las demás acciones del mismo estado:

$$P_t(s,a) = \frac{e^{Q_t(s,a)}}{\sum_{b=1}^{|A|} e^{Q_t(s,b)}}. \tag{3.13}$$

3.4. Espacio de estados

Uno de los aspectos clave de un sistema de aprendizaje por refuerzo es el espacio de estados. Para que los algoritmos aquí empleados puedan converger se debe reducir el infinito número de lecturas sensoriales que recibe el robot en un número finito y manejable de estados, entendidos como situaciones relevantes y distinguibles unas de otras. Es importante que el número de estados sea lo suficientemente pequeño como para que los algoritmos de aprendizaje converjan en un tiempo aceptable. Por otro lado, el espacio de estados debe tener la suficiente riqueza como para permitir distinguir aquellas situaciones donde el robot necesita

ejecutar acciones diferentes. En los resultados que se muestran en este capítulo la creación de un espacio de estados es una tarea *ad hoc*, su diseño se hace teniendo en cuenta los diversos factores que afectan a la ejecución del robot: los sensores utilizados por el robot, la tarea a ejecutar y el entorno donde se llevará a cabo.

Para transformar la señal de entrada a un conjunto finito de situaciones se utilizaron técnicas de cuantificación vectorial, en particular una red de neuronas artificiales tipo SOM (Mapa Auto-Organizable, también conocidas como redes de Kohonen [70, 68]). La creación de las redes SOM es un paso previo al aprendizaje del robot. El conjunto de entrenamiento procedía de los datos capturados por un robot ejecutando un comportamiento similar al buscado. Cada neurona de la red SOM representa cada uno de los posibles estados en los que puede estar el robot.

3.4.1. Reducción dinámica del número de estados

El número de estados utilizados para representar el entorno alrededor del robot influye directamente en el tiempo requerido para encontrar una política óptima. Por esa razón es posible usar las propiedades de las cadenas de Markov [14] para ajustar dinámicamente el número de estados que intervienen en el aprendizaje.

Al transformar las diferentes situaciones que detecta el robot mediante sus sensores a un conjunto finito de estados, $S = \{s_1, .., s_N\}$, se puede crear una matriz de probabilidades de transición P de tamaño $N \times N$, donde cada componente P_{ij} representa la probabilidad de que el robot pase del estado s_i al estado s_j:

$$P_{ij} = \Pr\{s_t = j | s_{t-1} = i\}.$$

Por lo tanto, si tenemos alguna información sobre el estado en el que el robot está en el instante t, podemos intentar predecir el estado en el que el robot estará en $t+1$:

$$\overrightarrow{\chi_{t+1}} = (\chi_{t+1}(s_0), \chi_{t+1}(s_1), ..., \chi_{t+1}(s_N)) = \overrightarrow{\chi_t} P, \tag{3.14}$$

donde $\chi_{t+1}(s_i)$ es una distribución de probabilidad que representa la probabilidad de que en el instante $t+1$ el robot esté en el estado $s_i, \forall i = 1, ..., N$. $\overrightarrow{\chi_t}$ representa la distribución de probabilidad correspondiente al instante de tiempo t.

Si P es la matriz de transición de un sistema de Markov, y $\overrightarrow{\chi_{t+k}}$ es un vector de distribución con la propiedad de que $\overrightarrow{\chi_{t+k+1}} = \overrightarrow{\chi_{t+1}} P^k = \overrightarrow{\chi_{t+k}}$, entonces nos referimos a $\overrightarrow{\chi_{t+k}}$ como un vector estático. Esto significa que la probabilidad de encontrar al robot en el estado i es la

misma todo el tiempo: aunque el robot pasa de un estado a otro estado, la distribución de probabilidad permanece constante. Como el vector estático $\overrightarrow{\chi_{t+k}}$ no depende de t, se eliminará el sufijo y se denotará únicamente como $\overrightarrow{\chi}$. Cuando la matriz de transición P satisface la condición de que existe alguna potencia de P donde todos sus elementos son no nulos, está probado que $\overrightarrow{\chi}$ existe y es único, es lo que llamamos el estado estacionario de Markov [53].

En nuestro caso, durante el proceso de aprendizaje se estima la matriz de transición para poder calcular $\overrightarrow{\chi}$. Una vez se hace esto, y ya que $\overrightarrow{\chi}$ define cómo el robot ve el entorno a largo plazo, únicamente aquellos estados s_j para los que la probabilidad a largo plazo es no nula $(\chi(s_j) \neq 0)$ se consideran en el proceso de aprendizaje.

La descripción de esta técnica se incluye aquí ya que es un método que se probó con éxito en el pasado [53] y es útil para representaciones *ad hoc*. Durante las pruebas de esta tesis no se usará esta técnica ya que, si bien su aplicación permite reducir el tiempo de aprendizaje, únicamente es útil si se aplica a una representación estática de estados. Como veremos a partir del capítulo 4 nuestros algoritmos usarán una representación dinámica del espacio de estados. Por otra parte, la reducción dinámica del número de estados requiere un modelo de las transiciones entre estados, algo que queremos evitar.

Una utilidad del estado estacionario de Markov a mayores de la reducción del número de estados es que permite comprobar la estabilidad, en el sentido clásico de Lyapunov, de los algoritmos de aprendizaje [53]. Esto permite conocer si un algoritmo es capaz de converger a la misma solución independientemente de las condiciones iniciales, entendiendo por condiciones iniciales los Q-valores inicializados aleatoriamente. Hay que aclarar que para el caso que nos ocupa, la solución a la que nos referimos es la trayectoria final seguida por el robot, la cual se puede caracterizar por los estados por los que pasa el robot a lo largo de esa trayectoria, lo que permite asociar un estado estacionario a cada solución. Se realizó un estudio de la estabilidad de diferentes técnicas de aprendizaje mediante el análisis del estado estacionario de Markov. Para ello se construyó una red bayesiana con la que se mide la proximidad de las soluciones, y que sirve para clasificar cada aprendizaje en base al algoritmo utilizado. Por otro lado se midió directamente la distancia entre soluciones utilizando la métrica euclídea. Los resultados de este estudio muestran que los estados estacionarios obtenidos tras aprendizajes con el mismo algoritmo se parecen más entre sí que a los logrados con diferentes algoritmos. Esto quiere decir que cada uno de los algoritmos utilizados en el estudio tiende a converger a la misma solución.

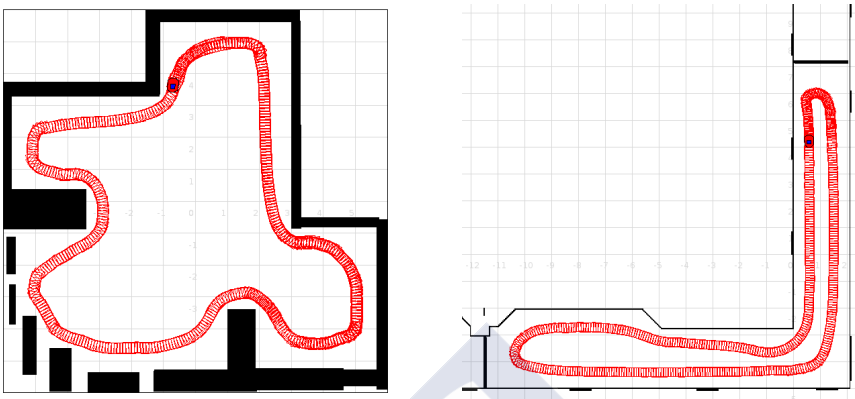
3.5. Aplicación experimental

Vamos a aplicar el algoritmo *I_Tbf* para dos tareas en particular. La primera es seguir pared, donde el robot debe circular manteniendo cierta distancia con la pared situada a su derecha. Se ha escogido la tarea seguir pared ya que es una tarea muy representativa y habitual en robótica móvil. Esta tarea es la que usaremos como comparativa a lo largo de toda esta tesis y con la que ya hemos trabajado en el pasado [104, 106, 34, 95]. La segunda tarea donde aplicaremos el algoritmo *I_Tbf* es la tarea cruzar puerta. En este caso el robot debe cruzar una puerta situada a escasos metros frente a él. Al igual que con el comportamiento de seguir pared, la finalidad de estos aprendizajes no radica únicamente en el aprendizaje de un controlador útil, sino que servirán como banco de pruebas para validar la eficiencia de los algoritmos. Es por ello que estos dos comportamientos son utilizados a lo largo de toda esta tesis.

Para poder comparar el algoritmo *I_Tbf* con algoritmos clásicos se realizarán nuevos aprendizajes en las mismas condiciones utilizando los algoritmos *Q-learning* [135], *Watkins's Q(λ)* [135] y *Naive Q(λ)* [123]. Estos algoritmos se han presentado en la sección 2.3.3. Todas las implementaciones de los algoritmos mostrados en estas pruebas utilizan la misma estrategia de selección de acciones (ecuación 3.13). Los parámetros utilizados son: coeficiente de aprendizaje $\beta = 0,288282$, y para los algoritmos *Watkins's Q(λ)* y *Naive Q(λ)* se usó $\gamma = 0,9$ y $\lambda = 0,869965$.

Todas las pruebas se han realizado en un entorno simulado utilizando el *software* Player/Stage [43]. El ejecutar las pruebas en simulación permite repetir los experimentos bajo las mismas condiciones y promediar los resultados. Al comienzo del aprendizaje el robot es colocado en una posición segura, donde no recibe refuerzo negativo, y se mueve hasta cometer un error. Al conjunto de experiencias acumuladas desde el posicionamiento del robot hasta el fallo se le llama episodio. Tras cada fallo tendrá lugar un proceso de actualización de las valoraciones de los pares estado-acción visitados durante el episodio y el robot será recolocado automáticamente en una nueva posición sin refuerzo, cercana y anterior al punto en el que se cometió el error, desde donde comenzará un nuevo episodio.

Cuando la política de control es capaz de controlar al robot por un intervalo de 10 minutos, se considera que el algoritmo ha convergido a una buena política de control y se finaliza el proceso de aprendizaje. Si por el contrario, el algoritmo no es capaz de encontrar una política de control válida en un tiempo inferior a 4 horas se considera que no se pudo alcanzar la



(a) (b)

Figura 3.2: Entornos de aprendizaje y trayectorias seguidas por los robots una vez que el comportamiento seguir pared fue aprendido.

convergencia y el aprendizaje se da por finalizado. Con el fin de obtener datos fiables sobre el rendimiento de los algoritmos cada prueba se ha repetido 15 veces.

El robot usado es un Pioneer P3-DX equipado con un sensor láser SICK LMS-200 y la duración del ciclo de control es de 300 ms. El sensor láser proporciona medidas de distancia cada grado en un arco de 180°. De esta forma se obtienen 181 medidas que representan la distancia a los obstáculos ubicados en la dirección de avance del robot. Durante estos aprendizajes el robot se mueve a una velocidad lineal constante de 0,15 m/s y el conjunto de acciones disponibles será un grupo de 19 velocidades angulares en el rango $[-0,7,\ 0,7]$ rad/s. Parte de las pruebas que se muestran aquí se corresponden con las publicadas en [93].

3.5.1. Seguir pared

Para aprender esta tarea el robot recibe un refuerzo que penaliza aquellas situaciones donde el robot está demasiado cerca o demasiado alejado de la pared que se sigue ($r < 0$), en otro caso $r = 0$. Los algoritmos de aprendizaje se aplicaron para el aprendizaje de esta tarea en los entornos simulados mostrados en la figura 3.2.

Para reducir la dimensionalidad de los estados se agruparon los datos del láser en 8 haces de 22,5º, donde la medida proporcionada por cada uno se corresponde con el mínimo de las

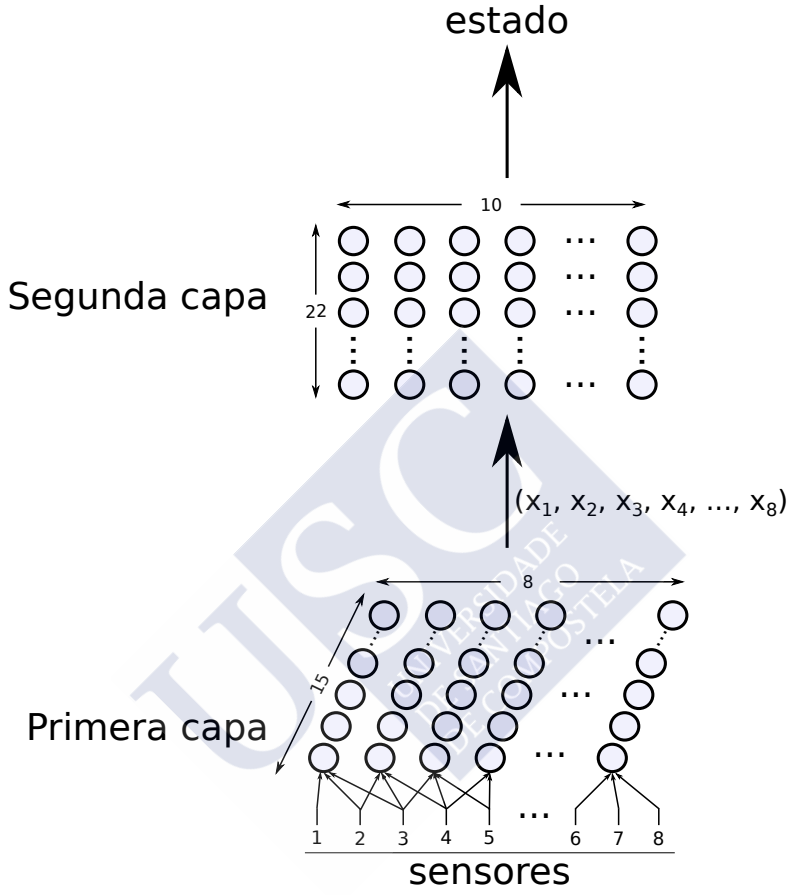


Figura 3.3: Esquema de la estructura jerárquica de redes SOM utilizada para la representación de estados a partir de las lecturas sensoriales. Los sensores utilizados son los haces láser descritos en la ecuación 3.15.

medidas agrupadas en el mismo haz:

$$haz_i = \min\left\{laser_{i\cdot 22}, \cdots, laser_{(i+1)\cdot 22}\right\} \forall i \in [0,\ 7]. \tag{3.15}$$

Para trasladar el enorme número de situaciones diferentes que los sensores pueden detectar a un número manejable de estados se utiliza una estructura jerárquica de redes de Kohonen [51]. Dicha estructura consiste en dos capas de redes SOM (figura 3.3). La primera capa consiste en un conjunto de redes unidimensionales, cada una de las cuales procesa la información

de tres sensores adyacentes, y donde dos redes consecutivas se solapan en dos sensores. El objetivo de esta primera capa es clasificar paredes a diferentes distancias y ángulos. Cada una de las neuronas de una red unidimensional será sensible a paredes con la misma orientación pero a diferentes distancias. Por otro lado, cada grupo de neuronas en la misma posición pero en diferentes redes serán sensibles a paredes a la misma distancia pero con distinta orientación. Con el fin de reducir la dimensionalidad de la salida de la primera capa se introduce una segunda capa consistente en una red SOM bidimensional de 220 neuronas, lo que proporciona el número total de estados en los que se pueden clasificar las entradas sensoriales. Para entrenar dichas redes se utilizó un conjunto de medidas sensoriales recogidas durante recorridos del robot cercanos a la pared, por lo tanto la red es capaz de reconocer aquellas situaciones que ocurren con más frecuencia durante la ejecución de un comportamiento de seguir pared.

En la figura 3.2 se pueden ver dos ejemplos de trayectorias seguidas por el robot después de que las políticas de control fuesen aprendidas. El tiempo medio de aprendizaje en el entorno de la figura 3.2(a) del algoritmo *I_Tbf* fue de 16:38 minutos, con una desviación estándar de 8:52 minutos (tabla 3.1). Anteriormente en el grupo de investigación se aplicaron diferentes estrategias con el objetivo de reducir el tiempo de aprendizaje para esta misma tarea [53, 108]. El mejor tiempo medio de aprendizaje obtenido fue de 24:12 minutos, el cual fue logrado usando un controlador difuso como consejero del algoritmo de aprendizaje. En [53] también se prueba el algoritmo $TTD(\lambda, m)$, el cual proporciona un tiempo medio de aprendizaje de 41:16 minutos. Ninguno de los algoritmos clásicos probados fue capaz de mejorar el tiempo de aprendizaje obtenido por *I_Tbf* (tabla 3.1). El rendimiento de *Naive Q(λ)* ha sido muy similar al de nuestra propuesta, las diferencias en el tiempo medio y la desviación no son significativas. *Watkins's Q(λ)* proporciona un tiempo de aprendizaje y una desviación ligeramente superiores. Como era de esperar, el algoritmo *Q-learning* es el que peores resultados da, siendo su tiempo medio de aprendizaje casi el doble del obtenido con *I_Tbf*. Ningún algoritmo tuvo problemas de convergencia al aprender esta tarea, todos ellos encontraron una política de control válida en un tiempo razonable.

Adicionalmente a este primer conjunto de aprendizajes, se llevaron a cabo experimentos utilizando el algoritmo *I_Tbf* en el entorno mostrado en la figura 3.2(b). En este nuevo entorno el tiempo medio de aprendizaje fue de 21:45 minutos y la desviación estándar fue de 10:30 minutos. Esta diferencia en el tiempo de aprendizaje se explica porque, aunque el entorno es más sencillo, su tamaño hace que las zonas más complicadas de aprender se visiten menos

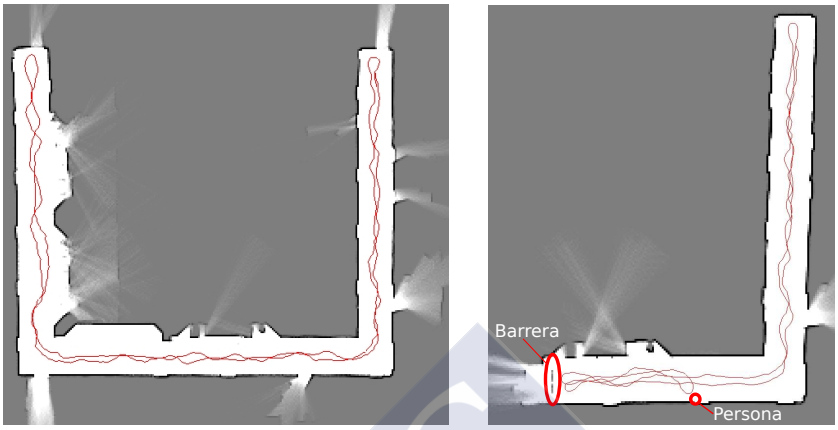
Tabla 3.1: Resultados del aprendizaje de la tarea seguir pared con diferentes algoritmos de aprendizaje y utilizando un espacio de estados prefijado. Tiempos de aprendizaje en *minutos:segundos*.

Algoritmo	Tiempo medio	Desviación estándar	Tasa de fallo
I_Tbf	16:38	08:52	0 %
Q-learning	29:37	13:59	0 %
Watkins's Q(λ)	21:35	12:32	0 %
Naive Q(λ)	17:21	08:21	0 %

Figura 3.4: Imagen del pasillo del Departamento de Electrónica e Computación donde se llevaron a cabo las pruebas con el robot real. Se pueden observar obstáculos como los bancos o los radiadores que no se incluían en la simulación.

frecuentemente, por lo que el robot necesita más tiempo para volver a ellas y aprender a superarlas correctamente.

Para comprobar la fiabilidad y robustez de las políticas aprendidas se decidió probar algunas de las políticas de control aprendidas en simulación en un entorno real. Las pruebas se realizaron en los pasillos del Departamento de Electrónica e Computación (DEC) de la Universidade de Santiago de Compostela. Es importante tener presente que el entorno real contiene elementos que no aparecen en simulación, tales como bancos, puertas abiertas o gente caminando (figura 3.4). Esto, unido a las inexactitudes que tienen los sensores reales, hace que el robot muestre un comportamiento algo más errático que en simulación.



(a) (b)

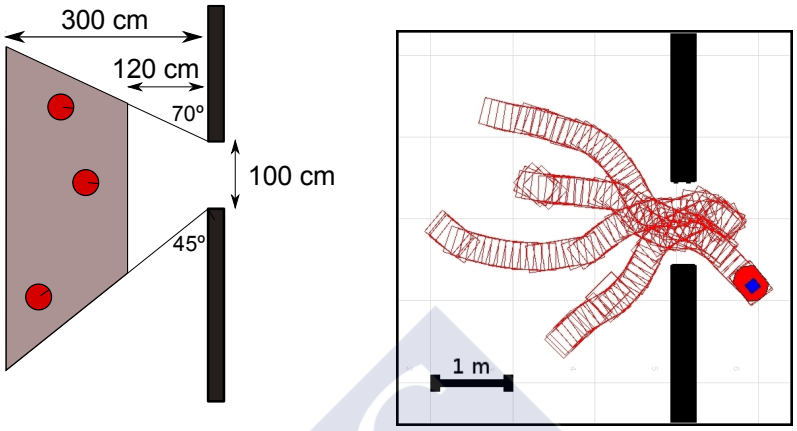
Figura 3.5: Trayectorias ejecutadas por el robot real ejecutando una política de seguir pared derecha. En (b) se puede observar la zona donde se situó una barrera y el punto en el que el robot realiza un giro al detectar a una persona. Píxeles negros significan espacio ocupado, blancos espacio libre y grises desconocido.

En la figura 3.5 se muestran dos ejemplos de trayectorias seguidas por el robot en el entorno real, los mapas se han generado *a posteriori* a partir de los datos adquiridos por el sensor láser durante la ejecución de la tarea. En la prueba mostrada en la figura 3.5(b) se situó una barrera para acotar el entorno. También se puede observar en la parte inferior de la imagen cómo el robot gira en el medio del pasillo debido a una persona que se interpuso en su camino. Dado que la persona es un obstáculo móvil, el algoritmo de generación del mapa no lo considera como espacio ocupado (es decir, obstáculos estáticos del entorno).

3.5.2. Cruzar puerta

Tras probar el algoritmo con la tarea de seguir pared se realizaron pruebas con una nueva tarea: cruzar puerta. El entorno donde el robot debe aprender la tarea es el mostrado en la figura 3.6(a). Al principio de cada episodio el robot se coloca en una posición aleatoria dentro de la zona sombreada mostrada en la imagen y orientado hacia el centro de la puerta.

Para este comportamiento el espacio de estados utilizado son los 220 estados procedentes de la red SOM que se utilizaron para el comportamiento seguir pared derecha. Esta represen-



(a) (b)

Figura 3.6: Tarea de cruzar puerta. a) Entorno de simulación usado; b) Trayectorias generadas por el robot una vez la política ha sido aprendida.

Tabla 3.2: Resultados del aprendizaje de la tarea cruzar puerta con diferentes algoritmos de aprendizaje y utilizando un espacio de estados prefijado. Tiempos de aprendizaje en *horas:minutos:segundos*.

Algoritmo	Tiempo medio	Desviación estándar	Tasa de fallo
I_Tbf	00:46:33	00:36:14	0%
Q-learning	00:43:25	00:22:17	47%
Watkins's Q(λ)	01:02:51	00:47:21	0%
Naive Q(λ)	01:36:53	00:57:55	0%

tación la elegimos porque, tal y como se demuestra en [54], para una tarea de cruzar puerta, contar únicamente con los sensores de un lado es suficiente para poder aprenderla con éxito.

El objetivo del robot será el cruzar la puerta sin salirse de la zona sombreada y sin chocar con ningún obstáculo. En caso de desviarse de la zona marcada o de detectar un obstáculo demasiado cercano el robot recibirá refuerzo negativo. Dado que ésta es una tarea episódica, si el robot logra cruzar con éxito la puerta se recoloca en una nueva posición aleatoria y se continúa la ejecución del mismo episodio. Por lo tanto únicamente se da por finalizado un episodio y se actualizan los valoraciones en caso de error.

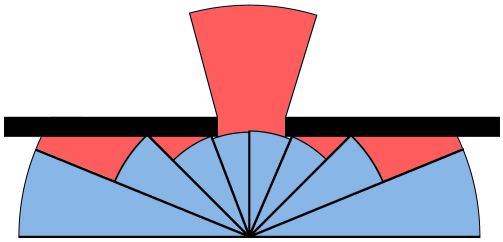


Figura 3.7: El agrupar la información del láser en arcos de 22,5º y tomar como medida la menor detectada dentro del arco – azul – provoca que la puerta sea imposible de detectar en ciertas situaciones. Si en lugar de agrupar el láser en haces se utiliza cada componente individual – rojo – la puerta se detecta correctamente.

En la tabla 3.2 se muestran los resultados de los aprendizajes realizados con los algoritmos *I_Tbf*, *Q-learning*, *Watkins's Q(λ)* y *Naive Q(λ)*. A la hora de medir el rendimiento de los algoritmos hay que tener en cuenta la tasa de fallo, esto es, el porcentaje de aprendizajes que no logran encontrar una política de control válida para la tarea. Para calcular el tiempo medio de aprendizaje se han descartado aquellos aprendizajes que no lograron converger, por esta razón *Q-learning* muestra el mejor tiempo de aprendizaje pese a tener una elevadísima tasa de fallo.

Descartando *Q-learning* por su alta tasa de fallo, el algoritmo *I_Tbf* proporciona el mejor tiempo medio de aprendizaje (46:33 minutos con una desviación de 36:14 minutos). Le sigue *Watkins's Q(λ)*, con un tiempo de poco más de una hora, 01:02:51 horas, y una desviación de 47:21 minutos. *Naive Q(λ)* tiene un tiempo de aprendizaje de 01:36:53 horas y una desviación de 57:55 minutos.

Una hipótesis que justificaría los elevados valores de los tiempos de aprendizaje y unas tasas de fallo tan altas es que la percepción no es la adecuada para esta tarea. No sólo el espacio de estados no es el ideal (ya que ha sido diseñado para la tarea seguir pared derecha), sino que al agrupar la información del láser en arcos de 22,5º hay muchas posiciones desde las que no es posible para el robot detectar la puerta de forma correcta (figura 3.7). Esto provoca que los aprendizajes sufran de solapamiento perceptual, que dependiendo del algoritmo utilizado podrá solventarlo con mayor o menor éxito. Para poder disponer de una percepción correcta sería necesario crear un nuevo espacio de estados donde no se agruparan todos los haces láser (sombreado rojo en la figura 3.7) o se utilizaran arcos más pequeños.

3.6. Discusión

En este capítulo se ha presentado una nueva función objetivo para el aprendizaje por refuerzo. Esta nueva función permite predecir el tiempo que transcurrirá hasta que el robot cometa un fallo. La información del tiempo antes del fallo proporciona aprendizajes más interpretables que la clásica función objetivo que representa el refuerzo esperado a largo plazo. La segunda aportación de este capítulo es un nuevo algoritmo de aprendizaje por refuerzo llamado *I_Tbf*. Este algoritmo implementa la función objetivo de tiempo antes del fallo y sus mayores contribuciones son el eliminar la necesidad de ajustar parámetros y proporcionar buenos tiempos de aprendizaje.

La nueva función objetivo aquí propuesta cambia la valoración de los estados, la cual pasará de ser la suma descontada del refuerzo esperado para un estado, o par estado-acción, a ser el tiempo esperado hasta que el robot cometa un fallo. Esto permite interpretar de manera más sencilla el proceso de aprendizaje, y comprobar de una forma mucho más intuitiva si la valoración de un estado realmente es acertada o no. La predicción del tiempo antes del fallo abre nuevas vías para la mejora del algoritmo de aprendizaje. La diferencia entre el tiempo antes del fallo predicho para un par estado-acción y lo que se observa realmente permite detectar errores en el aprendizaje de una manera mucho más directa, lo que facilitaría implementar mecanismos de autoevaluación del algoritmo.

Se ha implementado esta función objetivo en un nuevo algoritmo llamado *I_Tbf*. Este algoritmo se desarrolla con el fin de obtener procesos de aprendizaje con un menor número de parámetros y que proporcione buenos tiempos de convergencia. Para probar el rendimiento del algoritmo *I_Tbf* se hicieron diferentes pruebas donde el robot debía aprender los comportamientos seguir pared y cruzar puerta, y se compararon los resultados obtenidos con aquellos correspondientes a tres algoritmos clásicos: *Q-learning*, *Watkins's Q(λ)* y *Naive Q(λ)*. Los resultados obtenidos con *I_Tbf* son los mejores, junto con *Naive Q(λ)* para la tarea de seguir pared, con un tiempo medio de aprendizaje de 16:38 minutos. Con este reducido tiempo de aprendizaje el algoritmo se muestra válido para su aplicación en un robot real. En la tarea de cruzar puerta *I_Tbf* muestra los mejores resultados con bastante diferencia respecto al resto de algoritmos. Considerando las ventajas en lo referido a simplicidad e interpretabilidad del algoritmo *I_Tbf*, y unido a los buenos tiempos de aprendizaje obtenidos, este algoritmo será el que utilicemos para el desarrollo de un sistema de aprendizaje desde cero en el robot real.

Con el algoritmo aquí mostrado se solventan algunos de los problemas del aprendizaje por refuerzo reseñados en el capítulo 2, pero aún quedan otros por resolver. El principal escollo

que sufre el sistema presentado en este capítulo es la necesidad de disponer de una percepción creada *a priori* específicamente para la tarea a aprender. En el capítulo siguiente se mostrará cómo se soluciona este problema mediante la creación dinámica del espacio de estados.

Capítulo 4

Representación dinámica del entorno

Como se ha comentado en el capítulo 1, una característica necesaria de los robots, si queremos que operen en entornos reales, es la capacidad de adaptarse a cambios en el entorno. Para esta tarea es necesario por un lado adaptar el comportamiento, y por otro adaptar la forma que el robot tiene de percibir dicho entorno. En este capítulo se presentará una estrategia capaz de detectar eventos relevantes en el flujo de información sensorial, y así adaptar la forma en que el robot percibe el entorno.

Con el sistema de percepción que se presenta en este capítulo será posible aplicar el aprendizaje a diferentes tareas, entornos o sensores, sin realizar ningún ajuste previo del espacio de estados. Cada vez que el robot deba aprender una tarea empezará sin ningún conocimiento sobre el espacio de estados, y a medida que el robot vaya explorando el entorno se irán añadiendo nuevos estados para representar las distintas situaciones que el robot encuentra durante su interacción con el entorno.

Esta tarea se hará mediante la creación dinámica de estados utilizando para ello una red de neuronas basada en la teoría de la resonancia adaptativa, más conocida por sus siglas en inglés ART [44]. En concreto se usará una red *Fuzzy ART* [26].

Al igual que en el capítulo anterior, queremos cuantificar la información procedente de los sensores del robot, de manera que se obtenga un conjunto finito de diferentes estados adecuado para el aprendizaje. A diferencia de la estrategia mostrada en el capítulo 3, la red *Fuzzy ART* no necesita un diseño *ad hoc* previo al entrenamiento, sino que el propio sistema

será capaz de ir creando dinámicamente el espacio de estados a medida que explora el entorno. Esta característica facilita enormemente la aplicación del sistema de aprendizaje a diferentes entornos, tareas y sensores.

Una característica necesaria en nuestro sistema es que el robot debe ser capaz de identificar situaciones importantes aún cuando éstas ocurran de manera muy poco frecuente. La red *Fuzzy ART* permite la creación de estados que representen estas situaciones sin que pasen desapercibidas al haber otras situaciones más comunes. Por ejemplo, en un comportamiento de seguir pared la situación de estar paralelo a la pared será muy común. Por el contrario un situación donde se está frente a una esquina será muy infrecuente en comparación a la anterior. Nuestro sistema debe ser capaz de identificar ambas situaciones y de evitar que la alta frecuencia de aparición de una situación afecte a la otra.

A continuación se explicará el funcionamiento de las redes *Fuzzy ART* y su adaptación para nuestro sistema. Tras ver el funcionamiento de la red *Fuzzy ART* se integrará con el algoritmo de aprendizaje *I_Tbf* presentado en el capítulo anterior (aunque sería posible aplicarlo a cualquier algoritmo de aprendizaje por refuerzo) para proporcionar un espacio de estados adecuado para la tarea que está siendo aprendida.

4.1. Teoría de la resonancia adaptativa (ART)

Una de las características de la memoria humana consiste en su habilidad para aprender nuevos conceptos sin necesidad de olvidar otros aprendidos en el pasado. Sería deseable que esta misma capacidad se pudiera conseguir en las redes neuronales. Sin embargo, muchas de estas redes tienden a olvidar informaciones pasadas al tratar de aprender otras nuevas.

Muchas de las redes de neuronas artificiales, o incluso estimaciones estadísticas destinadas a la clasificación de patrones, se entrenan de forma supervisada. Esto es, se emplea un conjunto de patrones de ejemplo, de los cuales se conoce la salida deseada del sistema, que serán utilizados durante la fase de entrenamiento. Una vez concluido el aprendizaje, y viendo que la red funciona correctamente, ya no se permite ningún cambio adicional en ésta. Este procedimiento es factible si el problema que se pretende resolver por la red está bien limitado y puede definirse un adecuado conjunto de entrenamiento. Sin embargo, en muchas situaciones reales los problemas no tienen unos límites claros.

Esto es lo que S. Grossberg denomina como el dilema de la estabilidad y plasticidad en el aprendizaje [45]. Este dilema plantea los siguientes interrogantes:

– Cómo una red podría aprender nuevos patrones (plasticidad del aprendizaje).

– Cómo una red podría retener los patrones previamente aprendidos (estabilidad del aprendizaje).

En respuesta a este dilema, Grossberg, Carpenter y otros colaboradores desarrollaron la denominada teoría de la resonancia adaptativa (*Adaptative Resonance Theory*: ART) [23]. Lo que se pretende es categorizar dinámicamente los datos que se introducen en la red. Las informaciones similares son clasificadas formando parte de la misma categoría, y por tanto deben activar la misma neurona de salida, la neurona vencedora. Las clases o categorías deben ser creadas por la propia red, puesto que se trata de un aprendizaje no supervisado.

Para solucionar el problema de la plasticidad y estabilidad, el modelo ART propone añadir a las redes un mecanismo de realimentación entre las neuronas competitivas de la capa de salida de la red y la capa de entrada. Este mecanismo facilita el aprendizaje de nueva información sin destruir la ya almacenada.

La teoría de la resonancia adaptativa se basa en la idea de hacer *resonar* la información de entrada con los representantes o prototipos de las categorías que reconoce la red. Si entra en resonancia con alguno, es suficientemente similar, la red considera que pertenece a dicha categoría y únicamente realiza una pequeña adaptación del prototipo almacenado como representante de la categoría para que incorpore algunas características del dato presentado. Cuando no *resuena* con ninguno de los patrones almacenados por la red, la red se encarga de crear una nueva categoría con el dato de entrada como prototipo de la misma.

Como resultado de este enfoque, los autores mencionados presentaron varias redes neuronales especialmente adecuadas para tareas de clasificación de patrones. Éstas son ART-1 (o ART) [23], ART-2 [22], ART-3 [24], *Fuzzy ART* [26] y ARTMAP [25]. La red ART-1 trabaja con vectores de entrada binarios; ART-2 es capaz de procesar informaciones continuas o analógicas; ART-3 es una evolución de ART-2 con un mayor parecido en su funcionamiento a las neuronas naturales, puede ejecutar búsquedas paralelas de patrones distribuidos en una jerarquía de red multinivel; *Fuzzy ART* es una mejora de ART-1 combinándola con la teoría de conjuntos difusos, esta red es capaz de clasificar tanto vectores de entrada binarios como analógicos; por último ARTMAP combina dos unidades ART-1 o ART-2 ligeramente modificadas en una estructura de aprendizaje supervisado, la primera unidad toma la información de entrada y la segunda unidad toma la salida deseada, que será utilizada para realizar el mínimo ajuste posible de los parámetros de la primera red.

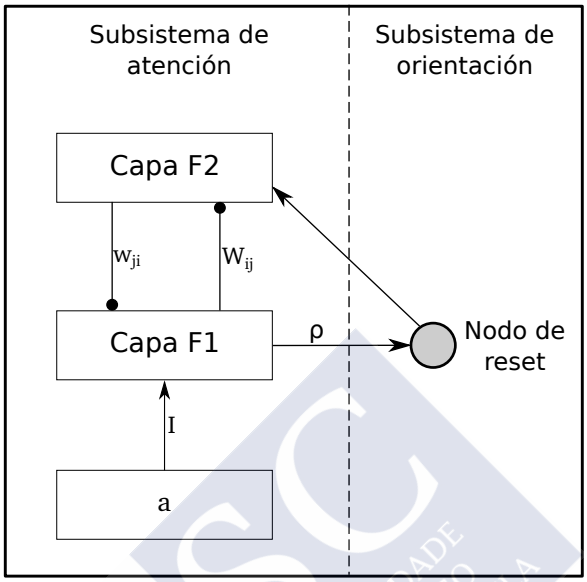


Figura 4.1: Vista esquemática de una red *Fuzzy ART*. La entrada I se introduce en la red y a continuación se realiza un proceso competitivo (W_{ij}). Se comprueba la resonancia de la neurona ganadora (w_{ji}) y en caso de no superarla se pone dicha neurona a *reset*.

La naturaleza de nuestro problema hace que no sea posible el utilizar un sistema de clasificación supervisada. Cuando el robot se esté moviendo por el entorno no será posible saber si una percepción concreta debe pertenecer o no al estado al que ha sido asignada. Por esta razón para nuestro sistema se ha elegido utilizar una red *Fuzzy ART*.

4.1.1. *Fuzzy ART*

La red *Fuzzy ART* es capaz de realizar un aprendizaje rápido y estable de estados (generalmente llamados categorías) en respuesta a secuencias arbitrarias de entradas analógicas o binarias. Aquí aprovecharemos esta característica para realizar el aprendizaje dinámico del espacio de estados.

En la figura 4.1 se muestra una vista esquemática de una red *Fuzzy ART*. Dicha red se compone de dos subsistemas: el subsistema de atención se encarga de la memorización de las categorías. Este subsistema contiene las capas F1 y F2, las conexiones entre las capas representan los procesos de activación y resonancia de neuronas. El subsistema de orientación

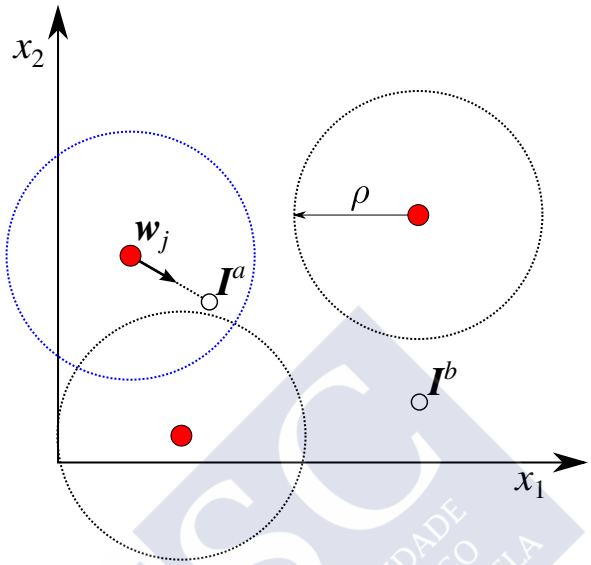


Figura 4.2: Representación del funcionamiento de una red ART.La entrada $\mathbf{I}^a$ se asigna a la categoría w_j dado que está dentro del radio de vigilancia ρ. La entrada $\mathbf{I}^b$ no entra en resonancia con ninguna categoría, por lo que se creará una nueva categoría para representar dicha entrada.

se encarga de inhibir las neuronas de la capa F2 que no han entrado en resonancia con la entrada I para permitir continuar la búsqueda de categorías. A continuación se muestra en detalle el funcionamiento de la red *Fuzzy ART* (figura 4.2):

Vector de entrada: cada entrada **I** de una red *Fuzzy ART*, capa F1, será un vector M-dimensional $(I_1, ..., I_M)$, donde cada componente I_i pertenece al intervalo [0, 1].

Vectores prototipo: cada neurona j de la capa F2 se corresponderá con un vector prototipo de pesos adaptativos $w_j \equiv (w_{j1}, ..., w_{jM})$, el número de categorías potenciales es arbitrario.

Parámetros: la dinámica de una red *Fuzzy ART* está determinada por un parámetro de elección $\alpha > 0$; un coeficiente de aprendizaje $\beta \in [0, 1]$; y un parámetro de vigilancia $\rho \in [0, 1]$.

Proceso competitivo de elección de categoría: tras la presentación de un patrón de entrada **I** se produce un proceso competitivo entre las neuronas (estados) de la capa F2, tras el cual los estados se ordenarán en función de su valor de activación. Para cada entrada **I** y un

estado j, la función de activación T_j se define como:

$$T_j(\mathbf{I}) = \frac{|\mathbf{I} \wedge w_j|}{\alpha + |w_j|}. \tag{4.1}$$

Las formas más comunes de definir el operador difuso AND ($\wedge$) son a través de la función mínimo o el producto. En nuestro caso elegiremos el mínimo:

$$(x \wedge y)_i \equiv min(x_i, y_i), \tag{4.2}$$

y donde la norma $|\cdot|$ se define como

$$|x| \equiv \sum_{i=1}^{M} |x_i|. \tag{4.3}$$

El parámetro α hace que, en el caso de que dos neuronas de la capa F2 se exciten por igual, tienda a activarse la neurona más nueva. Esto se debe a que para un mismo valor de α, se favorece la elección de aquellas neuronas con mayor norma; y generalmente los prototipos favorecidos son los últimos introducidos en la red, ya que, debido a la función de actualización usada que se verá más adelante, la norma de un prototipo únicamente podrá disminuir con el tiempo.

Por simplicidad en la notación, $T_j(\mathbf{I})$ en la ecuación 4.1 se escribe como T_j cuando la entrada $\mathbf{I}$ es fija. Para elegir una categoría éstas se indexan por J, donde

$$J = \arg\max_j \{T_j : j = 1...N\}. \tag{4.4}$$

Si hay varias categorías que comparten el valor máximo T_j se escoge aquella con menor índice j.

Resonancia o *reset*: tras el proceso competitivo anterior, se comprueba para la categoría ganadora j si la red entra en resonancia. Se produce resonancia si la función de semejanza supera al parámetro de vigilancia ρ; esto es, si

$$\frac{|\mathbf{I} \wedge w_j|}{|\mathbf{I}|} \geq \rho. \tag{4.5}$$

En caso de que dicha relación no se cumpla ocurrirá un *reset*. Entonces, entra en funcionamiento el nodo de *reset* y el valor de la función de elección T_j se asigna a -1 para que dicha categoría no vuelva a ser seleccionada durante la búsqueda para el patrón de entrada actual.

Un nuevo índice J se escoge siguiendo la ecuación 4.4 y el proceso continúa hasta que un elemento satisfaga la ecuación 4.5 o no se encuentre ninguna categoría que entre en resonancia con la entrada.

Aprendizaje: cuando la red entra en resonancia ante una entrada, el vector prototipo w_j que representa el estado se actualiza a partir de la siguiente ecuación:

$$w_j^{(new)} = \beta(\mathbf{I} \wedge w_j^{(old)}) + (1-\beta)w_j^{(old)}. \tag{4.6}$$

El aprendizaje rápido se corresponde con $\beta = 1$.

Creación de nuevas categorías: en el caso de que ninguna de las categorías actuales provoque que la red entre en resonancia se deberá crear una nueva categoría que represente al vector de entrada. Esta nueva categoría será inicialmente igual al vector de entrada:

$$w_j^{(new)} = \mathbf{I}. \tag{4.7}$$

Normalización de las entradas: para evitar la proliferación de categorías las entradas pueden ser normalizadas, esto es, para algún $\gamma > 0$, $|\mathbf{I}| \equiv \gamma$, para todas las entradas $\mathbf{I}$.

Una regla de normalización, llamada codificación complementaria o *complement coding*, normaliza las entradas a la vez que mantiene la información de amplitud. El complementario de un vector de entrada $\mathbf{a}$, representado como $\mathbf{a}^c$, es

$$a_i^c \equiv 1 - a_i. \tag{4.8}$$

La codificación complementaria de una entrada $\mathbf{I}$ es un vector 2M-dimensional

$$\mathbf{I} = (\mathbf{a}, \mathbf{a}^c) \equiv (a_1, ..., a_M, a_1^c, ..., a_M^c). \tag{4.9}$$

En este caso

$$|\mathbf{I}| = |(\mathbf{a}, \mathbf{a}^c)| = M. \tag{4.10}$$

De todos los parámetros, el más importante es la vigilancia, ρ. Dicho parámetro determina la granularidad de la clasificación. Valores bajos de ρ producirán clasificaciones con menos estados que valores altos de ρ. A medida que ρ se aproxima a 1, habrá prácticamente un estado por cada vector de entrada.

4.1.2. Adaptación de *Fuzzy ART* para el aprendizaje en el robot real

Tal y como se señala en el artículo donde se presenta la red *Fuzzy ART* [26], la regla de aprendizaje de la red *Fuzzy ART* – ecuación 4.6– puede ser modificada para adaptarla a

las necesidades específicas de cada problema. En nuestro caso la función de aprendizaje que utilizamos es de la forma:

$$w_j^{(new)} = \beta \mathbf{I} + (1-\beta) w_j^{(old)}. \tag{4.11}$$

Esta función de aprendizaje es utilizada por Moore en [80]. En dicho trabajo se indica que el aprendizaje puede ser inestable debido a que los prototipos no siempre decrecen. Ante esto Carpenter y colaboradores afirman que el uso de la codificación complementaria con esta función de actualización evita la proliferación de categorías.

El uso de esta función de actualización nos proporciona claras ventajas a la hora de implementar el algoritmo *Fuzzy ART* para su uso en el aprendizaje en tiempo real. Al trabajar con codificación complementaria y al utilizar la ecuación 4.11 para actualizar los prototipos se cumplirá que: $|\mathbf{I}| = |w_j| = M$. Esto implica que, si se elimina el parámetro α de la función de activación (ecuación 4.1) ésta sea igual a la función de semejanza (ecuación 4.5). Esto nos proporciona una gran mejora en el costo computacional: si la categoría ganadora del proceso competitivo, aquella con mayor T_j, no entra en resonancia, ninguna otra categoría podrá hacerlo, por lo que se evitan los ciclos de búsqueda de resonancia habituales en las redes *Fuzzy ART*.

4.2. Aprendizaje simultáneo de percepción y acción

Esta sección presenta la integración del algoritmo de aprendizaje por refuerzo con la red *Fuzzy ART*, esto permitirá a los robots aprender simultáneamente cómo percibir y cómo actuar a partir de su interacción con el entorno (figura 4.3). El sistema que proponemos consta de dos módulos: uno de ellos responsable de la percepción y el otro de la acción. El robot construye una representación del entorno mediante una red *Fuzzy ART* que va creciendo dinámicamente para incluir nuevas situaciones que no han sido vistas antes (*aprendizaje de percepción* en la figura 4.3). Estas nuevas situaciones serán los estados del sistema. Por otro lado, la representación dinámica del entorno se combina con una estrategia de aprendizaje (*aprendizaje de acción* en la figura 4.3) capaz de adaptar el comportamiento del robot en aquellas situaciones relevantes (estados) que son identificadas por el módulo de percepción.

Anteriormente se intentó lograr este mismo objetivo [49], sin embargo el sistema, aunque prometedor, necesitaba demasiados parámetros y presentaba aprendizajes lentos. Con el algoritmo *I_Tbf* ahora podemos proponer una nueva alternativa donde el número de parámetros se reduce al mínimo y se consiguen aprendizaje rápidos.

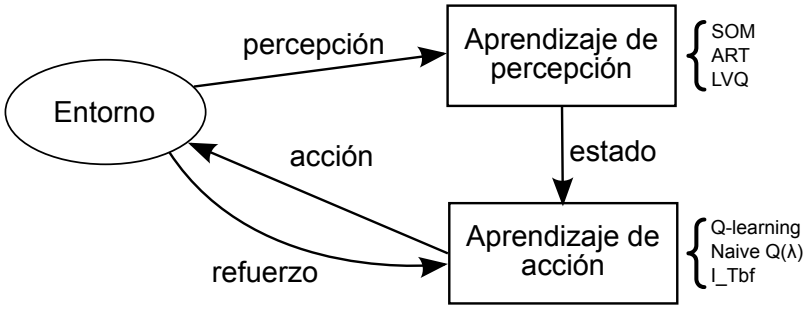


Figura 4.3: Esquema general de nuestra propuesta para el aprendizaje simultáneo de percepción y acción.

Mediante la combinación de la red *Fuzzy ART*, con la regla de aprendizaje modificada, y el algoritmo de aprendizaje I_Tbf se puede lograr el aprendizaje simultáneo de percepción y acción. La partición del espacio sensorial en regiones depende únicamente de las percepciones recibidas y no de la evolución del proceso de aprendizaje. Deseamos vincular el tamaño de las regiones al proceso de aprendizaje, de manera que se adapten automáticamente a las necesidades del aprendizaje. De esta manera se evita la necesidad de ajustar el parámetro de vigilancia.

A continuación se presentan dos alternativas al ajuste del parámetro de vigilancia, estas alternativas permiten el aprendizaje de diferentes tareas sin la necesidad de ajustar previamente un valor de vigilancia adecuado para ellas. La primera propuesta ajusta el parámetro de vigilancia en función del aprendizaje. Este parámetro será independiente para cada estado, por lo que podrá haber una división más pequeña allí donde se necesite. La segunda propuesta consiste en añadir criterios adicionales para insertar un nuevo estado. En este caso todos los estados compartirán el mismo valor de vigilancia, intencionadamente bajo, y se insertarán nuevos estados donde se requiera una representación más densa.

4.2.1. Red *Fuzzy ART* de vigilancia variable

La primera propuesta para resolver el problema del ajuste del parámetro de vigilancia, ρ, es una modificación de la red *Fuzzy ART*, la llamamos Red *Fuzzy ART* de vigilancia variable (*Fuzzy ART VV*) [102]. En este caso no habrá un parámetro de vigilancia común para todos los estados, sino que cada estado contará con su propio parámetro que podrá ser modificado en función del progreso del aprendizaje. Tener diferentes valores de vigilancia permitirá a la

red representar cada área del entorno con diferente granularidad. Esto permitirá tener menos estados, ya que únicamente donde sea preciso se utilizará un valor de vigilancia muy alto.

Esta propuesta implica que se debe prevenir que los estados pequeños, aquellos con un ρ alto, no lleguen a activarse debido a la presencia de estados grandes en sus cercanías (ecuación 4.1). En nuestro caso, para evitar que esto suceda los nuevos estados añadidos a la red empezarán con una vigilancia igual a la de su vecino más próximo. Empezaremos utilizando valores bajos de ρ, y únicamente se incrementará su valor en aquellas zonas que requieran un mayor detalle.

El primer estado introducido en la red empezará con un valor de vigilancia de 0. Obviamente es un valor erróneo, y todas las percepciones serán asignadas a este primer estado mientras no se incremente el valor de ρ. La estrategia seguida para encontrar el mejor el valor de vigilancia para cada neurona será empezar con un ρ bajo y, en caso de detectar solapamiento perceptual, incrementar el valor de ρ hasta que desaparezca el solapamiento.

La detección del solapamiento perceptual es la parte crítica de esta propuesta. Para detectar el solapamiento perceptual en un estado se estudia el resultado de la ejecución de cada acción. Durante el proceso de aprendizaje el robot recibirá refuerzo negativo, y aquellas acciones ejecutadas antes del error serán marcadas como erróneas para el estado en el que fueron ejecutadas. Se prefiere ser agresivo en la marcación de acciones erróneas, por eso una vez una acción falla se marca como errónea y no vuelve a desmarcarse hasta que se modifica la vigilancia del estado. Si para un estado dado un número significativo de acciones son erróneas, se asume que ese estado sufre solapamiento perceptual, y como consecuencia de ello se incrementará su valor de vigilancia. Al incrementar la vigilancia, el radio del estado disminuye, por lo que será más probable que se cambie de estado cerca del error, permitiendo así ejecutar una acción que evite el error.

4.2.2. Inserción dinámica de neuronas vinculada al aprendizaje

Esta segunda propuesta, consistente en la inserción de neuronas, surge de la sospecha de que puede haber solapamiento perceptual en los estados que han cometido un error teniendo una buena valoración o tras un cierto tiempo de ejecución sin consecuencias negativas. Mediante la predicción del tiempo a fallo proporcionada por el algoritmo *I_Tbf* podemos detectar estas situaciones y condicionar la partición del espacio sensorial a la evolución del aprendizaje. Con el fin de deshacer el solapamiento para poder ejecutar una acción diferente, y así evitar

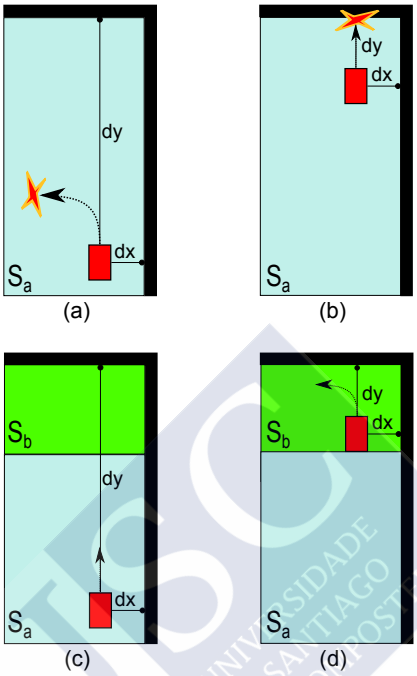


Figura 4.4: Solapamiento perceptual causado por un parámetro de vigilancia demasiado bajo para el comportamiento seguir pared: en (a) y (b) el robot está en el estado S_a y, la acción que es correcta en (a) causa un error en (b). Si se ejecutara la acción adecuada para (b) en la situación (a) también ocurriría un error. Si se inserta un estado cerca del error ocurrido en (b) se permitirá un cambio de acción, lo que permite ejecutar la acción adecuada en cada situación, (c) y (d).

el error, se insertará una nueva neurona entre el estado original y el punto donde se cometió el fallo.

La primera condición para la creación de una nueva categoría se basa en aprovechar la información del tiempo predicho hasta el siguiente fallo que proporciona el algoritmo *I_Tbf*, de esta manera se podrá detectar cuándo una predicción falla, principalmente porque es demasiado optimista, y en consecuencia insertar una neurona cercana al punto de fallo para provocar un cambio de acción.

Otra condición para insertar un nuevo estado en la percepción será que el estado que cometa un fallo se haya ejecutado de manera continuada durante un cierto número de ciclos antes del error. Esta situación se puede ver en la figura 4.4. En el estado S_a se ejecuta la acción

de seguir recto (a), la cual es correcta durante gran parte del tiempo que el robot permanece en dicho estado pero termina causando un error al colisionar con la pared (b). Si cerca del punto de error se inserta un nuevo estado (c), S_b, el robot podrá evitar el solapamiento perceptual y aprender la tarea correctamente (d).

El objetivo de todo esto es lograr la creación automática de una percepción acorde con la tarea que se está aprendiendo, y que no requiera ajustar ningún parámetro. El principal problema relativo con el parámetro de vigilancia era encontrar un valor que minimizara el tiempo de aprendizaje. Para esta propuesta se escogerá un valor de vigilancia bajo, no necesariamente cercano al que suponemos correcto, y que sabemos que proporcionará una representación con solapamiento perceptual. Este solapamiento será solucionado mediante la inyección de estados. En resumen, las condiciones de creación de un nuevo estado son las siguientes:

1. **No resonancia:** se inserta una neurona si la entrada no resuena con ninguna de las existentes. Se usa como prototipo la entrada que no resonó en la red.

2. **Fallo de un estado con *Tbf* alto:** si el estado en M ciclos antes del error tiene un *Tbf* alto. Se usa como prototipo la entrada en el ciclo M antes del error.

3. **Mucho tiempo en el mismo estado antes del error:** si durante los últimos M ciclos antes del fallo la red Fuzzy ART no ha cambiado de estado se inserta una clase usando como prototipo la entrada en el ciclo M antes del momento actual.

Una vez más, incidimos en cómo a través de esta alternativa se vincula la generación de neuronas a la propia evolución del proceso de aprendizaje.

4.3. Aplicación experimental

Se ha realizado un exhaustivo estudio experimental para probar la combinación de la red *Fuzzy ART* con el aprendizaje por refuerzo (figura 4.5). Las tareas a aprender en los experimentos han sido las mismas que en el capítulo anterior (ver sección 3.5): seguir pared (el robot debe aprender a moverse manteniendo una cierta distancia con la pared a su derecha) y cruzar puerta (el robot debe aprender a atravesar una abertura en la pared frente a él). Al igual que en el capítulo anterior, el objetivo de estos experimentos no es únicamente el aprender los comportamientos, sino también servir de banco de pruebas para nuestros algoritmos.

Con estos experimentos queremos responder a varias preguntas: ¿el aprendizaje simultáneo de estados y acciones incrementa significativamente el tiempo de aprendizaje? ¿Es la red

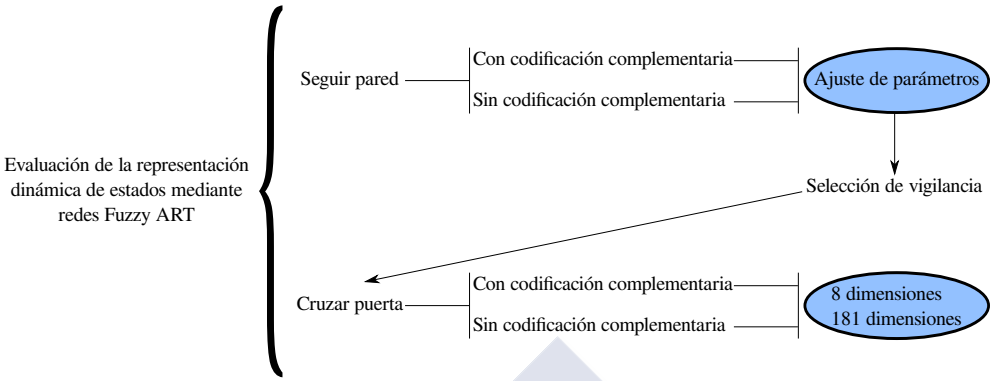


Figura 4.5: Conjunto de experimentos para evaluar evaluar la creación dinámica de estados mediante *Fuzzy ART*. Se buscarán los valores válidos de vigilancia para el comportamiento seguir pared. La mejor configuración de la red *Fuzzy ART* será entonces utilizada para aprender la tarea cruzar puerta añadiendo dimensiones al vector de entrada.

Fuzzy ART una estrategia válida para obtener una representación del estado incremental y no supervisada?

En estos experimentos primero hemos analizado el impacto que tiene el parámetro de vigilancia (figura 4.5), e intentamos obtener un valor aproximado que sirva de partida para futuros aprendizajes. Esta es la razón del segundo conjunto de pruebas: hacemos que el robot aprenda una tarea diferente (cruzar puerta en este caso) usando el valor de vigilancia previamente seleccionado. No podemos asegurar que el valor de vigilancia sea correcto para cualquier tarea, pero podría ser un buen punto de partida. En estos últimos experimentos también hemos probado los efectos de aumentar la dimensión de los vectores de entrada de la red *Fuzzy ART*. En todos los experimentos se han utilizado tanto la codificación complementaria de las entradas (CC) como las entradas sin aplicar ningún tipo de normalización, con el objetivo de poner a prueba la influencia de la codificación complementaria y obtener algunas pistas sobre las ventajas y desventajas del uso o no uso de la codificación complementaria.

Las condiciones de los experimentos y los valores de los parámetros son los mismos que los utilizados en el capítulo 3. Para todos los experimentos el robot aprenderá a ejecutar cada tarea en un entorno de simulación utilizando el *software* Player/Stage [43]. Se considera que el aprendizaje ha convergido si el robot es capaz de moverse ejecutando el comportamiento

deseado durante 10 minutos, o que fracasa si no se aprende una política válida en 4 horas. Los resultados de estos experimentos se han publicado en [93, 96].

Considerando los resultados obtenidos en el capítulo anterior hemos elegido probar tanto la combinación de *Naive Q(λ)* y *Fuzzy ART*, como el algoritmo de aprendizaje *I_Tbf* y *Fuzzy ART*.

Tal y como se mencionó en la sección 4.1, el vector de entrada **I** de la red *Fuzzy ART* debe estar compuesto de valores en el intervalo [0,1]. Esto implica que es necesario realizar una transformación de la información proporcionada por el láser. El láser usado (SICK LMS-200) proporciona 181 medidas – una medida cada grado de 0ºa 180º–, y cada una de estas componentes puede tener un valor entre 0 y 8 metros. Al igual que sucedía en el capítulo anterior, hemos agrupado la información del láser en haces de 22,5º, esto permite una reducción de la dimensión del vector de entrada al obtener un vector de sólo 8 componentes. El valor de cada uno de esos componentes es la mínima distancia medida por el láser en el intervalo. Por lo tanto será necesario escalar los valores del intervalo [0, 8] al intervalo [0, 1]:

$$I_i = \frac{1}{(1+input_i)} - \frac{1}{9} \tag{4.12}$$

La función de normalización escogida se debe a que, para las tareas para las que inicialmente se diseña el sistema, se considera que las medidas menores serán más relevantes que aquellas mayores. Es decir, se decidió crear una función de normalización que maximizara los cambios en distancias pequeñas y minimizara los cambios observados a grandes distancias.

4.3.1. Seguir pared

En este primer bloque de experimentos se ha estudiado cómo la variación del parámetro de vigilancia, ρ, afecta al aprendizaje. Para ello se han lanzado una serie de baterías de 15 aprendizajes donde tanto I_Tbf como *Naive* $Q(\lambda)$ han sido evaluados utilizando diferentes valores de vigilancia en la red *Fuzzy ART*.

En la figura 4.6(a) se muestran los resultados de los aprendizajes cuando la red *Fuzzy ART* utiliza como entradas la información sensorial sin complementar. A partir de estos resultados se puede afirmar que los espacios de estados creados con un valor de vigilancia menor que 0,875 no son válidos para aprender la tarea, ya que ni I_Tbf ni *Naive* $Q(\lambda)$ logran converger en un tiempo razonable. Bajo estas condiciones *Naive* $Q(\lambda)$ obtiene mejores tiempos de aprendizaje que I_Tbf. En ambos casos el mejor aprendizaje se consigue utilizando $\rho = 0,9125$. Para I_Tbf el tiempo medio es 43:16 minutos, con una desviación estándar de

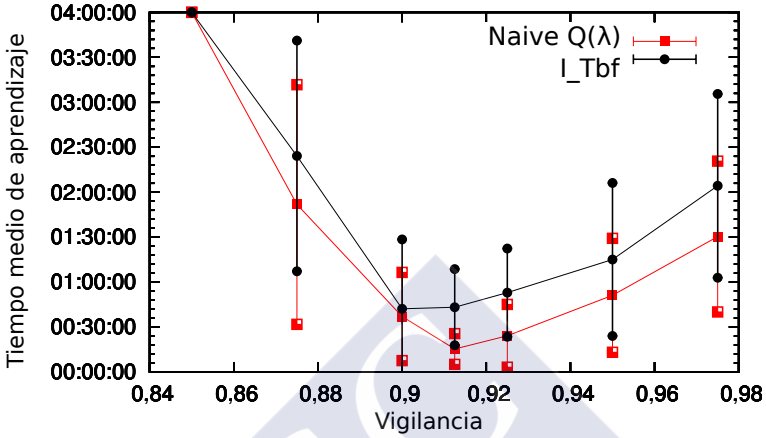
25:24 minutos. Con *Naive* $Q(\lambda)$ el tiempo aprendizaje es mucho más rápido, con un valor medio de 15:23 minutos y una desviación de 10:16 minutos. La evolución del número de estados respecto al parámetro de vigilancia se puede observar en la figura 4.6(b). Incrementos en el valor de la vigilancia provocan que el número de estados creados durante el aprendizaje crezca exponencialmente.

Si las entradas de la red *Fuzzy ART* se codifican utilizando codificación complementaria las diferencias entre los dos algoritmos disminuyen (figura 4.7(a)). En este caso el rendimiento de nuestro algoritmo es muy similar al conseguido con *Naive* $Q(\lambda)$. El rango de vigilancia válido crece y todos los tiempos de aprendizaje mejoran. El valor óptimo de ρ se mantiene en 0,9125. Dicho valor proporciona un tiempo medio de aprendizaje con I_Tbf de 22:18 minutos y una desviación de 14:10 minutos. Con *Naive* $Q(\lambda)$ el tiempo medio es de 14:50 minutos y una desviación de 09:27 minutos. En el caso de la codificación complementaria, si ρ continúa aumentado su valor el crecimiento del número de estados es mucho más acusado, llegando a valores medios cercanos a 1000 estados para valores de ρ próximos a 0,98 (figura 4.7(b)).

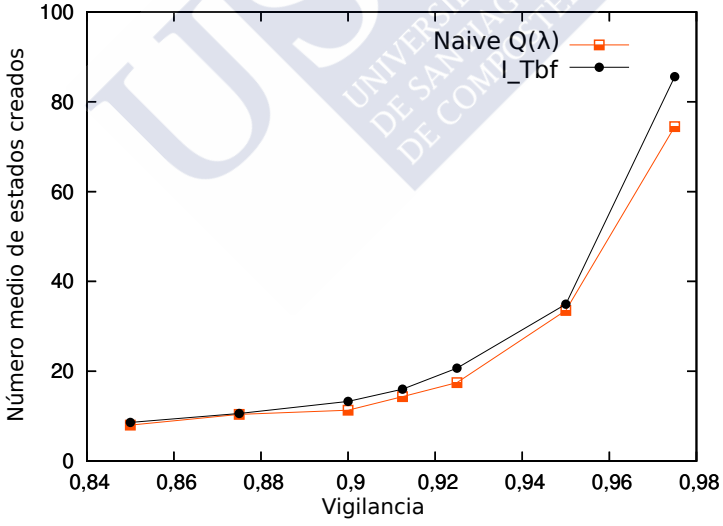
Estos resultados muestran oscilaciones difíciles de explicar. Se observa cómo el algoritmo *Naive* $Q(\lambda)$ mejora ligeramente su rendimiento utilizando una red *Fuzzy ART* frente al uso de una red SOM predefinida. Por el contrario, el algoritmo I_Tbf muestra peores tiempos de aprendizaje si se utiliza una red *Fuzzy ART* que si se utiliza la red SOM. Consideramos que las diferencias en estos tiempos de aprendizaje no son significativas debido a la alta desviación que muestran los tiempos de aprendizaje.

Tras probar el funcionamiento del sistema en simulación, se transfirieron las políticas de control aprendidas utilizando el algoritmo *I_Tbf* al robot real. Los entornos de simulación son diferentes a los entornos donde se llevaron a cabo las pruebas reales. Dado que durante estas pruebas no tiene lugar ningún proceso de aprendizaje, la red *Fuzzy ART* no creará nuevas categorías. En el caso de que una percepción no entre en resonancia con ningún estado, en lugar de crear un nuevo estado, el sistema asignará la percepción al estado con mayor valor de semejanza (ecuación 4.5).

En la figura 4.8 se muestran las trayectorias del robot durante la ejecución de la tarea seguir pared en dos entornos diferentes. Las posiciones iniciales se indican con una estrella y los círculos representan posiciones donde el robot cometió un error. En la figura 4.8(a) la razón del fallo fue que el robot se dirigía a una puerta de cristal, que es transparente para el

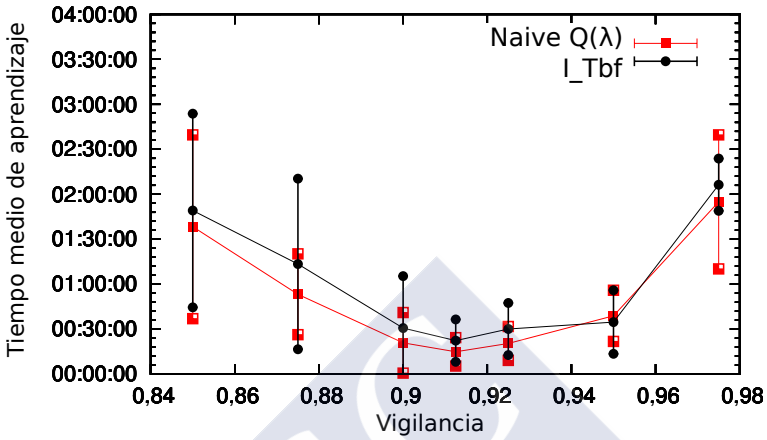


(a)

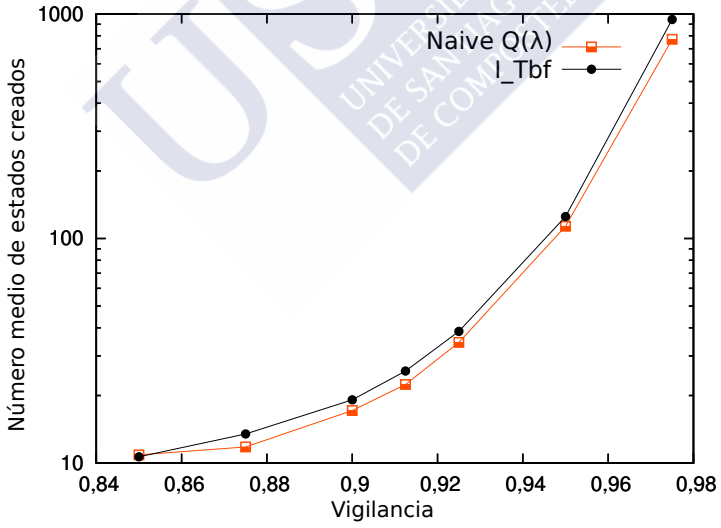


(b)

Figura 4.6: Resultados de los aprendizajes para la tarea seguir pared combinando *Fuzzy ART* y *I_Tbf* o *Naive* $Q(\lambda)$ para distintos valores de vigilancia. No se utiliza codificación complementaria: a) tiempo medio de aprendizaje y desviación estándar, y b) número medio de estados creados.

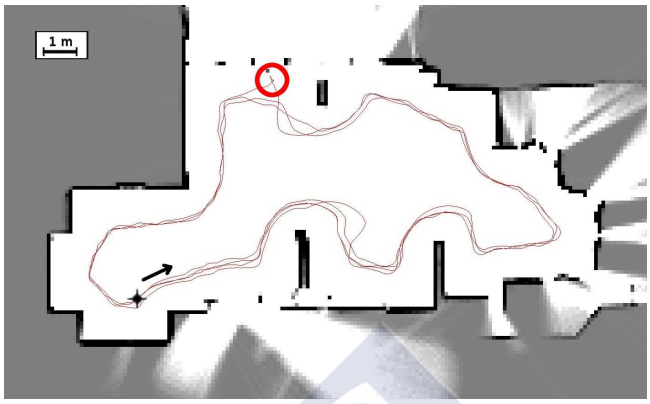


(a)

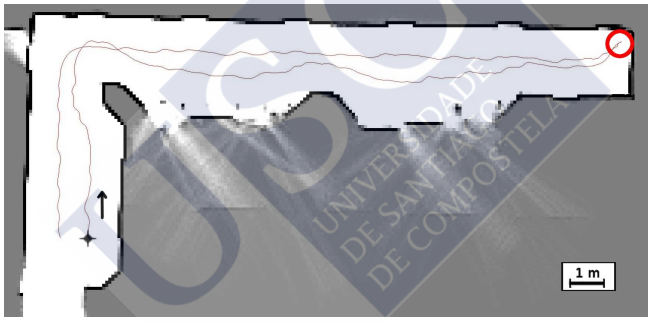


(b)

Figura 4.7: Resultados de los aprendizajes para la tarea seguir pared combinando *Fuzzy ART* y I_Tbf o *Naive* $Q(\lambda)$ para distintos valores de vigilancia. Las entradas utilizan codificación complementaria: a) tiempo medio de aprendizaje y desviación estándar, y b) número medio de estados creados.



(a)



(b)

Figura 4.8: Trayectorias del robot mientras ejecuta la tarea seguir pared en el entorno real. Mapas generados a partir de los datos de odometría y del sensor láser durante la ejecución del comportamiento. Las estrellas marcan las posiciones iniciales. Los círculos marcan las posiciones donde el robot cometió un error. Los píxeles blancos representan espacio libre, los negros espacio ocupado y los grises espacio desconocido.

sensor láser. En la figura 4.8(b) el error fue debido a que la entrada no resonó y el estado asignado no ejecutaba la acción correcta en esa situación.

4.3.2. Cruzar puerta

Siguiendo el esquema de las pruebas mostrado en la figura 4.5, tras evaluar el rendimiento de la red *Fuzzy ART* y seleccionar el mejor valor para el parámetro de vigilancia ($\rho = 0,9125$),

se ha examinado el comportamiento del sistema en un nuevo entorno y con una tarea diferente. Para esta tarea también se ha probado la respuesta de la red *Fuzzy ART* ante entradas de mayor tamaño (figura 4.5). Al igual que en la sección anterior, se han realizado 15 aprendizajes con cada configuración del sistema, incluyendo pruebas para medir el rendimiento del uso o no de la codificación complementaria. En la sección anterior se vio cómo el uso de la codificación complementaria mejoraba el rendimiento, para las pruebas que se mostrarán a continuación se decidió mantener las dos codificaciones dado que en alguno de los experimentos se incrementó el tamaño del vector de entrada y desconocemos cómo reaccionará la red *Fuzzy ART* ante vectores de alta dimensionalidad.

Al cambiar de tarea, y gracias a utilizar la red *Fuzzy ART* para la creación dinámica del espacio de estados, únicamente es necesario cambiar la definición de la función de refuerzo y ajustar los parámetros del algoritmo de aprendizaje (en el caso de *I_Tbf* no es necesario el ajuste de ningún parámetro). Si se quisiera usar una red predefinida, tal y como se hizo en el capítulo 3 con redes SOM, sería necesario entrenar una nueva red para el entorno y la tarea. Es en este caso donde se ve la mayor ventaja de usar una red dinámica para la creación del espacio de estados.

Elegimos probar el sistema utilizando los valores de vigilancia que mejores resultados nos dieron para el comportamiento seguir pared. La tabla 4.1 muestra los resultados de los experimentos usando *Naive Q(λ)* con las diferentes configuraciones de la red *Fuzzy ART* y del vector de entrada. La tabla 4.2 muestra los resultados para las mismas pruebas si se usa *I_Tbf*. En la sección 3.5.2 ya se comentó el hecho de que si se agrupa la información del láser en haces demasiado grandes, en este caso 22,5º, es imposible distinguir la puerta desde ciertas posiciones y distancias (ver figura 3.7). Para solucionar este problema optamos por realizar pruebas utilizando todas las componentes del láser (181) como entrada sensorial, por lo que se dispondrá de suficiente precisión para detectar la puerta (sombreado rojo en la figura 3.7).

Como se puede ver en las tablas 4.1 y 4.2, el mapeo de sensores a estados escala de manera correcta cuando se aumentan las dimensiones del vector de entrada de 8 a 181. El número de estados creados se mantiene en unos valores aceptables y los tiempos de aprendizaje se reducen al usar los datos individuales proporcionados por el sensor láser.

Combinando la red *Fuzzy ART* con los algoritmos de aprendizaje se mejora el tiempo de aprendizaje respecto a si se usa la red SOM utilizada en el capítulo 3. *Naive Q(λ)* mejora con *Fuzzy ART* su mejor tiempo de aprendizaje un 58 % del tiempo empleado con SOM, de 01:36:53 horas a 39:56 minutos. El tiempo de aprendizaje de *I_Tbf* se reduce, pero dado que

Tabla 4.1: Resultados del aprendizaje del comportamiento cruzar puerta con *Naive Q(λ)* y *Fuzzy ART*. Se realizaron pruebas con dos formas diferentes de procesar el láser y aplicando o no codificación complementaria.

Vigilancia	Láser	CC	Tiempo medio de aprendizaje	Desviación estándar	Número medio de estados
0,9125	8	No	NA	NA	63,22
		Sí	01:37:12	01:00:55	94,87
	181	No	01:57:48	01:05:31	134,33
		Sí	00:39:56	00:23:20	82,87
0,925	8	No	02:51:36	00:48:41	78,77
		Sí	01:41:19	00:38:23	146,30
	181	No	02:26:24	01:12:13	172,27
		Sí	00:56:18	00:39:50	123,33

Tabla 4.2: Resultados del aprendizaje del comportamiento cruzar puerta con *I_Tbf* y *Fuzzy ART*.

Vigilancia	Láser	CC	Tiempo medio de aprendizaje	Desviación estándar	Número medio de estados
0,9125	8	No	02:43:00	01:13:19	71,47
		Sí	01:34:37	01:03:26	99,13
	181	No	01:23:56	01:02:58	149,47
		Sí	00:34:11	00:14:38	88,40
0,925	8	No	02:54:28	01:11:49	73,73
		Sí	01:12:35	00:42:17	129,87
	181	No	01:29:59	01:00:16	187,40
		Sí	00:40:26	00:22:32	135,13

ya era un tiempo más corto la mejora no es tan significativa - se reduce un 39 % del tiempo de referencia con SOM, pasando de 46:33 minutos a 34:11 minutos.

Tal y como ocurrió con la tarea seguir pared, el uso de codificación complementaria mejora los tiempos de aprendizaje y permite la convergencia con un rango más amplio de valores de vigilancia.

Al igual que con el comportamiento seguir pared, una vez aprendidas las políticas de control en simulación utilizando el algoritmo *I_Tbf* se realizaron pruebas en un entorno real para evaluar el rendimiento de dichas políticas. La figura 4.9 muestra algunas de las trayectorias del robot mientras ejecuta la tarea cruzar puerta. Estas trayectorias fueron capturadas mediante un sensor láser externo. La imagen contiene algunos errores en la estimación de la orientación del robot debido a errores de lectura y procesamiento de los datos de dicho láser externo.

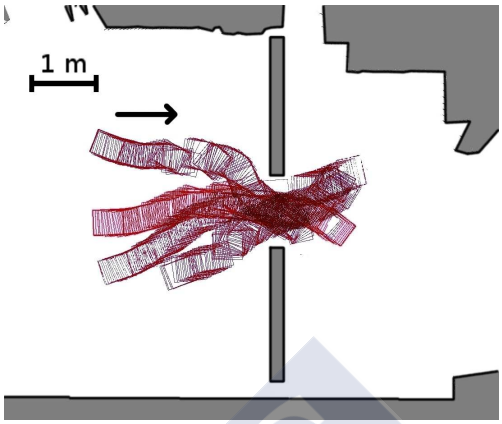


Figura 4.9: Trayectorias generadas por el robot durante la ejecución del comportamiento cruzar puerta. Esta imagen se generó utilizando un láser externo para seguir la localización del robot.

4.3.3. Modificación del espacio de estados vinculado al aprendizaje

En esta sección se muestran los resultados experimentales del aprendizaje de los comportamientos seguir pared y cruzar puerta utilizando las estrategias de modificación del espacio de estados en función del aprendizaje. En primer lugar veremos los resultados con la red *Fuzzy ART* de vigilancia variable. A continuación se muestran los resultados obtenidos utilizando la inserción dinámica de neuronas. En ambos casos se utilizó el mismo robot Pioneer P3-DX utilizado en anteriores experimentos. El valor de M, utilizado para considerar a un estado cercano al error se fijó en 20 ciclos. Dicho valor depende de la frecuencia de control y la velocidad del robot, en nuestro caso, con un ciclo de control cada 300 ms el valor $M = 20$ equivale a 6 segundos.

Red *Fuzzy ART* de vigilancia variable

Aquí se presentan los resultados obtenidos en los experimentos llevados a cabo con la red *Fuzzy ART* de vigilancia variable. Para estos experimentos el sensor láser se configuró de manera que se agruparon las lecturas en 8 haces de 22,5°. Como decíamos anteriormente, se han realizado experimentos con el comportamiento seguir pared y el comportamiento cruzar puerta. Se ha considerado que se debía incrementar la vigilancia de un estado cuando el 75 % de sus acciones eran erróneas. La vigilancia se aumentó en incrementos de 0,0125, empezando el primer estado con $\rho = 0$.

Para el comportamiento seguir pared se obtuvo un tiempo medio de aprendizaje de 20:45 minutos, con una desviación de 16:09 minutos y creándose una media de 21 estados. Estos resultados son muy similares a los obtenidos con una red *Fuzzy ART* sin modificaciones (tabla 4.3).

En el comportamiento cruzar puerta el tiempo medio de aprendizaje fue de 47:02 minutos, con una desviación de 40:32 minutos. Se crearon una media de 45 estados por aprendizaje. En este comportamiento se observa una gran desviación en los tiempos de aprendizaje, dándose aprendizajes que llevaron unos pocos minutos mientras que otros aprendizajes se demoraron más de una hora. Al igual que ocurrió anteriormente, estos tiempos tan altos se deben al uso del láser en haces de 22,5°, los cuales impiden detectar la puerta en ciertas situaciones.

Un problema de esta aproximación es que al principio de cada aprendizaje se pierde gran cantidad de tiempo hasta que se incrementa la vigilancia a valores cercanos al necesario para el aprendizaje. El uso de un valor inicial de ρ más alto aceleraría el proceso de aprendizaje.

Inserción dinámica de neuronas

Recordemos que en la estrategia de inserción dinámica de neuronas no es necesario ajustar el parámetro de vigilancia ρ, únicamente debe ser un valor inferior al óptimo para la tarea que se va a aprender. Para estas pruebas se escogió un valor de vigilancia de $0,85$, bastante por debajo del valor óptimo para aprender estas tareas. Nuestro sistema puede funcionar con cualquier rango de valores entre 0 y el valor de vigilancia óptimo. A medida que la vigilancia disminuye un mayor porcentaje de estados serán creados por inserción dinámica. Uno de los criterios de inserción es el fallo de un estado con un Tbf alto. En estos experimentos consideramos alto todo valor de Tbf mayor de 40 ciclos de control.

Para estas pruebas se configuró el robot para usar las 181 componentes del láser y las 16 del anillo de ultrasonidos, por lo que no se le introduce ningún sesgo a la capacidad perceptual del sistema.

En el comportamiento seguir pared el tiempo medio de aprendizaje fue de 09:41 minutos con una desviación de 06:19 minutos. Mucho mejor que cualquier tiempo conseguido anteriormente con una representación de estados estática o dinámica. El número medio de estados creados fue de $13,1$ estados, con una desviación de $5,5$ estados.

Las pruebas con el comportamiento cruzar puerta proporcionaron un tiempo medio de aprendizaje de 16:16 minutos y una desviación de 09:41 minutos. Se crearon una media de $28,6$ estados, con una desviación de $8,5$ estados.

Tabla 4.3: Mejores tiempos de aprendizaje obtenidos con cada combinación de módulos de aprendizaje de acción y de aprendizaje de percepción. Tiempo medido en horas:minutos:segundos.

Tarea	Algoritmo	Representación de estados	Tiempo medio de aprendizaje	Desviación estándar
Seguir pared	*Naive Q(λ)*	Estática	00:17:21	00:08:21
		Fuzzy ART	00:14:50	00:09:27
	I_Tbf	Estática	00:16:39	00:08:14
		Fuzzy ART	00:22:18	00:14:10
		Fuzzy ART VV	00:20:45	00:16:09
		Inserción dinámica	00:09:41	00:06:19
Cruzar puerta	*Naive Q(λ)*	Estática	01:36:53	00:57:55
		Fuzzy ART	00:39:56	00:23:20
	I_Tbf	Estática	00:46:33	00:36:14
		Fuzzy ART	00:34:11	00:14:38
		Fuzzy ART VV	00:47:02	00:40:32
		Inserción dinámica	00:16:16	00:09:41

Con estos resultados se puede afirmar rotundamente que el utilizar el algoritmo de inserción dinámica de neuronas vinculada al aprendizaje, no solamente evita el ajustar un valor de vigilancia, sino que proporciona unos tiempos de aprendizaje mucho mejores que otras alternativas estudiadas hasta el momento.

4.4. Discusión

En la tabla 4.3 se pueden ver los mejores tiempos de aprendizaje obtenidos para cada tarea con diferentes configuraciones. Los resultados obtenidos al utilizar una representación de estados dinámica son comparables o mejores a los de usar un espacio de estados estático. Esto, añadido al no emplear tiempo en la creación de un espacio de estados *ad hoc*, permite afirmar que el uso de la red *Fuzzy ART* es una mejora significativa para el sistema de aprendizaje. En lo referente al rendimiento de los algoritmos, al compararlos bajo las mismas condiciones se ve que en algunas situaciones *Naive Q(λ)* obtiene unos resultados ligeramente mejores que *I_Tbf*. Decidimos seguir utilizando el algoritmo *I_Tbf* en lo que resta de tesis ya que consideramos que las ventajas relativas a la reducción de parámetros y la interpretabilidad del algoritmo compensan esa ligera bajada de rendimiento.

El estudio de los diferentes valores de vigilancia nos proporciona un valor ρ óptimo para los dos comportamientos aquí probados. No podemos afirmar que dicho valor sea válido para otros nuevos comportamientos, pero se puede usar como punto de partida para nuevos

aprendizajes. La creación de estados en función de la evolución del aprendizaje es una aproximación interesante para resolver el problema del ajuste de la vigilancia, principalmente la estrategia basada en la inserción dinámica de estados, la cual proporciona unos resultados mucho mejores que cualquier otro método de los utilizados durante las pruebas. Pese a estos buenos resultados, las estrategias de modificación de la percepción en función del aprendizaje no serán utilizadas en lo que resta de tesis. Con el sistema presentado en el capítulo siguiente, se dispondrá de diferentes redes *Fuzzy ART* independientes entre sí que proporcionarán la riqueza necesaria para disponer de un espacio de estados adecuado para cada tarea.

Los test realizados en el entorno real demuestran que los comportamientos aprendidos con nuestra propuesta son útiles, pero también que es necesario probar el mayor número de situaciones posibles a las que se puede enfrentar el robot para conseguir un aprendizaje adecuado. El sistema de aprendizaje mostrado en este capítulo sigue presentando un problema de generalización, ya que un pequeño cambio del entorno podría afectar al controlador. A medida que se acelera el aprendizaje y se mejoran los tiempos medios obtenidos, se incrementa porcentualmente la desviación de los aprendizajes. El problema es que al mejorar el tiempo de aprendizaje el robot explorará menos su entorno, por lo que su espacio de estados no incluirá una gran variación de situaciones. Dado que en muchos casos el entorno donde se ejecutará el comportamiento es complejo, y a menudo cambia con el tiempo, se hace necesario un sistema de aprendizaje continuo, ya que será imposible aprender todas las posibles situaciones. En el capítulo siguiente describimos una nueva propuesta que hace frente a estos problemas.

Capítulo 5

Aprendizaje mediante comités

En el capítulo anterior se mostró un sistema que permite el aprendizaje de manera simultánea de percepción y acción, capaz de adquirir un comportamiento sin ningún tipo de conocimiento previo sobre el entorno o la tarea. Pese a que los resultados obtenidos con dicha estrategia fueron satisfactorios, el sistema no responde a todas las cuestiones planteadas en el capítulo 1. Es necesario acelerar el proceso de aprendizaje para permitir su uso en un robot real o incluso para conseguir que dicho aprendizaje sea continuo.

En este capítulo se presentan dos estrategias de aprendizaje donde, combinando un conjunto de aprendedores subóptimos independientes en un comité, el robot será capaz de aprender a partir de un conjunto mínimo de ejemplos mientras se mantiene una buena generalización de lo aprendido. Gracias a la combinación de aprendedores se podrá aprovechar al máximo cada conjunto de ejemplos sin que ello implique un sobreajuste a dichos ejemplos, lo que se conoce como dilema *bias-varianza*.

Una ventaja adicional proporcionada por el uso de comités es que permite añadir, suprimir o modificar aprendedores sin que ello implique una desestabilización del sistema, lo que ayuda en gran medida a conseguir aprendizajes continuos. Trabajar con una única política de control supone un problema cuando hay cambios en el entorno que requieren una readaptación del robot o simplemente la misma tarea se lleva a cabo en entornos diferentes. La idea es que el robot, bien partiendo de un comportamiento aprendido en simulación, bien sin ningún conocimiento previo, sea capaz de adaptar y modificar su comportamiento de acuerdo al entorno en el que se está moviendo. Bajo estas condiciones se presupone que el aprendizaje nunca finaliza, ya que ante cualquiera cambio o nueva situación a la que se enfrente el ro-

bot éste debe ser capaz de aprender a superarla. Esto implica que debe ser posible incorporar nuevo conocimiento, o destruir conocimiento previo, sin causar importantes inestabilidades en el comportamiento del robot. Creemos que el trabajo realizado y que se describe en estos capítulos 5 y 6 representa un avance importante de cara al aprendizaje continuo.

5.1. Dilema *bias-varianza*

Existen publicaciones que señalan el interés del aprendizaje en entornos reales a partir del refuerzo [126, 71]. Sin embargo, en la mayoría de estos trabajos el aprendizaje se consigue tras una meticulosa parametrización del espacio de acciones o la función de refuerzo, en lugar de rediseñar los algoritmos de aprendizaje para adaptarlos al aprendizaje en un robot real. Otros artículos se enfrentan al problema de aprender sin ningún conocimiento previo. En [11] el robot explora el entorno y utiliza sus propias experiencias para crear un modelo del entorno, dicho modelo se utiliza para mejorar la política de control y por consiguiente se consigue un aprendizaje por refuerzo cercano al tiempo real.

Considerando el coste de las interacciones robot-entorno, resulta obvia la necesidad de reducir el número de ejemplos para poder llevar a cabo aprendizajes en el robot real sin necesidad de recurrir a procesos simulados. Es por esto que se quiere acelerar el aprendizaje y sacar el máximo provecho a un conjunto de ejemplos limitado. Ahora bien, acelerar el proceso de aprendizaje implica aprender mucho a partir de pocos ejemplos, lo que nos lleva a un problema clásico del aprendizaje máquina, el dilema *bias-varianza* [42]. Dicho dilema consiste en alcanzar un compromiso entre un buen ajuste al conjunto de datos de ejemplo y el mantener la capacidad de generalización ante datos diferentes. Para solucionar este dilema se recurrirá a una estrategia habitual en redes de neuronas artificiales que consiste en la creación de comités. Esta solución se aplica generalmente a problemas de clasificación (tal y como se hace en AdaBoost [40] o *bagging* [19]). Un comité de clasificadores es un conjunto de clasificadores cuyas decisiones individuales son combinadas de alguna manera para clasificar nuevos ejemplos. El resultado es un comité que en general presenta una mejor clasificación que cada uno de sus clasificadores individuales [33]. Gracias a la combinación de un comité de diferentes expertos que puedan llegar a exhibir un bajo *bias* y alta *varianza*, se puede lograr un sistema con bajo *bias* y baja *varianza* [83].

Como se describió en los capítulos 2 y 3 el robot puede aprender una función de valor $Q(S \times A)$ basada en la información recopilada mientras el robot se mueve $(s_t, a_t, r_t, s_{t+1},$

$a_{t+1}, r_{t+1}, ...$). Dado que Q se aproxima a partir de un conjunto de ejemplos, D, obtenidos mediante la interacción del robot con el entorno, podemos asumir que existe cierta dependencia entre la función que se aprende Q, y el conjunto de datos empleados D. Por este motivo, y de forma similar a Sharkey [116], emplearemos la notación $Q(S \times A; D)$, en lugar de $Q(S \times A)$ [111], para representar la dependencia de la función Q en el conjunto de datos D. El error cuadrático medio de Q como función que refleja el tiempo antes de fallo (*Tbf*) real puede escribirse como:

$$E_D[(Q(S \times A; D) - Tbf[S \times A])^2], \tag{5.1}$$

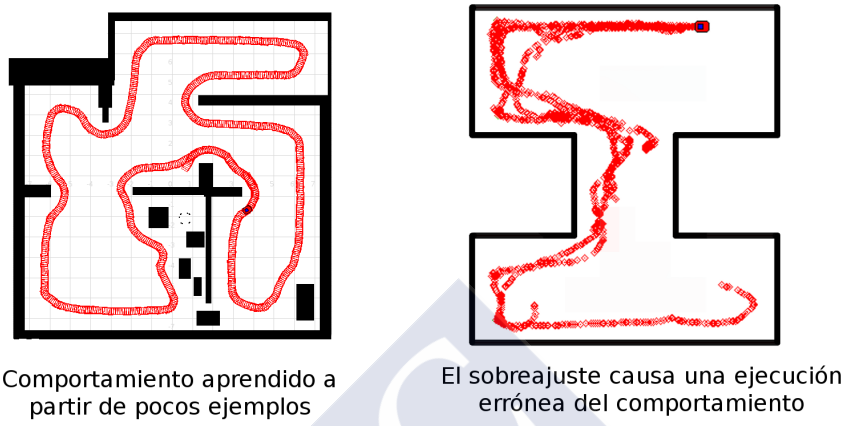
donde E_D es el operador esperanza relativo al conjunto de datos D. $Tbf[S \times A]$ es la función objetivo, esto es, el tiempo real que transcurrirá antes de que se produzca un fallo para cada par $s \in S$ y $a \in A$, y cuando a continuación el robot se controla mediante la política egoísta. Se puede dividir el error cuadrático medio en dos componentes (*bias* y *varianza*):

$$\begin{aligned} &E_D[(Q(S \times A; D) - Tbf[S \times A])^2] = \cdots \\ &(E_D[Q(S \times A; D)] - Tbf[S \times A])^2 && \text{bias} \\ &+ E_D[(Q(S \times A; D) - E_D[Q(S \times A; D)])^2] && \text{varianza} \end{aligned} \tag{5.2}$$

El *bias* representa en qué medida lo aprendido, Q, se ajusta al conjunto de datos D. Un bajo *bias* significa que el aprendedor se ajusta de manera precisa a los datos. La *varianza* mide cómo de sensible es la predicción ante datos diferentes a los utilizados durante el aprendizaje, es decir, mide si se habrían obtenido los mismos resultados si el conjunto de muestras fuese diferente. Por lo tanto una baja *varianza* implica una buena generalización.

Generalmente hay un compromiso entre *bias* y *varianza*; intentos para reducir el *bias*, es decir, ajustarse mucho al conjunto de aprendizaje, son proclives a provocar una alta *varianza* (figura 5.1), mientras que reducciones de *varianza* suelen provocar un aumento del *bias*, es lo que se conoce como el dilema *bias-varianza*.

Una estrategia para reducir tanto el *bias* como la *varianza* es usar un conjunto de expertos, en nuestro caso construiremos un comité de aprendedores. La mejora que se obtiene al utilizar un comité de aprendedores es generalmente una reducción de la *varianza*. Por lo tanto, una aproximación efectiva sería el utilizar un conjunto de aprendedores que proporcionen un bajo *bias*, lo cual podría provocar que dichos aprendedores tengan una alta *varianza*. Esto no es un problema ya que la reducción de *varianza* vendrá dada mediante la combinación de aprendedores. En las técnicas clásicas de clasificación mediante comités cada uno de los clasificadores no tiene por qué tener una gran tasa de acierto. Con clasificadores que tengan un



(a) (b)

Figura 5.1: Efectos del sobreajuste en el aprendizaje. Cuando el robot aprende a partir de pocos ejemplos (a), puede provocar que el comportamiento no se ejecute correctamente en un entorno diferente (b).

rendimiento ligeramente superior a un clasificador aleatorio, conocidos como clasificadores débiles, es posible construir un comité que clasifique con una precisión de aproximadamente el 100 % [114].

Es muy importante que los errores que cometan los aprendedores sean independientes. La importancia de tener errores no correlacionados se explica a partir del hecho de que, si de un conjunto de aprendedores únicamente algunos conocen la acción correcta y los demás comenten el error de escoger una acción incorrecta, esas acciones erróneas estarán repartidas por todo el espacio de acciones y la acción correcta destacará sobre las demás.

5.2. Creación de comités

Como se comentó al anteriormente, se pretende obtener un aprendedor óptimo mediante la combinación en un comité de aprendedores subóptimos independientes. El hecho de que cada miembro del comité proporcione un aprendizaje subóptimo no es un problema, es más, de esta manera se obtiene una *varianza* menor. La mayor dificultad al combinar las opiniones de aprendedores es que dichas opiniones tienden a estar correlacionadas o ser dependientes.

Una solución inspirada en comités de expertos fue presentada por Tham [124] en 1995. Recibe el nombre de arquitectura HME-CMAC debido a que el método propuesto se basa en una estructura jerárquica basada en Combinaciones Jerárquicas de Expertos (HME [57]) de redes CMAC [3]. Las redes CMAC (*Cerebellar Model Articulation Controller*) se propusieron como un modelo de controlador que imita el procesamiento de la información en el cerebelo. La red CMAC divide el espacio de entrada en varios conjuntos unidimensionales de neuronas, por lo que cada neurona sólo responde a una dimensión del vector de entrada. Para cada dimensión se realiza un proceso competitivo y como resultado del mismo se obtiene la salida de una neurona, la cual será combinada con las salidas del resto de neuronas ganadoras de cada dimensión para formar un vector, o hipercubo, de las mismas dimensiones que el vector de entrada. Una misma dimensión puede estar representada por varios conjuntos de neuronas, cada uno de los cuales realiza un procesamiento diferente de los datos de entrada, por lo que se dispondrá de varios hipercubos. Combinando estos hipercubos se obtiene un vector de pesos que pondera cada componente del vector de entrada para generar la salida de la red CMAC.

El otro elemento de la arquitectura HME-CMAC es el modelo HME. HME se basa en el principio de divide y vencerás, donde un problema de aproximación lineal es dividido en varios subproblemas más fáciles de resolver, cada uno de los cuales será resuelto por un experto. Cada uno de estos subproblemas puede a su vez dividirse en un nuevo nivel de subproblemas, construyendo así la estructura jerárquica que da nombre al modelo. Dentro de la arquitectura HME-CMAC las redes CMAC juegan el papel de cada uno de los expertos dentro de la jerarquía HME. La propuesta de Tham proporciona buenos resultado, aunque tiene la desventaja de ser sensible al ajuste de los parámetros y de requerir una gran cantidad de memoria para representar cada una de las redes CMAC.

En nuestro caso (figura 5.2), para lograr la independencia de los errores cada uno de los aprendedores del comité contará con su propio sistema de aprendizaje, basado en el algoritmo *I_Tbf* (capítulo 3), y su propio espacio de estados, basado en una red *Fuzzy ART* (capítulo 4). En lugar de construir un sistema que necesita determinar la acción adecuada para cada estado del robot, cada uno de los aprendedores deberá determinar el intervalo de acciones más adecuado para cada uno de estos estados. Será combinando los intervalos propuestos por los aprendedores que el sistema determinará la acción que será ejecutada, tomada de entre los intervalos propuestos por los aprendedores. De esta manera cada uno de los aprendedores será un aprendedor subóptimo, ya que no le pedimos que aprenda la acción correcta, le pedimos que aprenda una aproximación al intervalo de acciones que contiene la acción correcta.

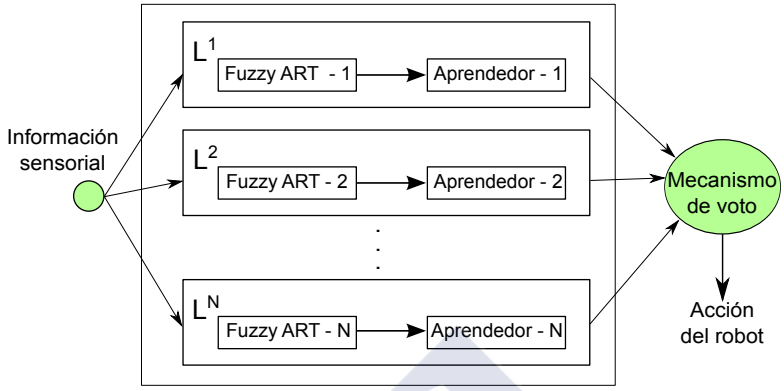


Figura 5.2: Esquema general del comité de aprendedores.

Durante el desarrollo de nuestra investigación hemos hecho diferentes propuestas de uso de comités como estrategias de aprendizaje [101, 110]. Ahora bien, en este capítulo solo se recoge la descripción de los algoritmos finales. No obstante, a través de estos trabajos previos hemos podido observar cómo los tiempos de aprendizaje y las varianzas se reducían al emplear los comités. Nos gustaría incidir en que el principal objetivo de los comités que se describen a continuación no es tanto reducir los tiempos de aprendizaje como incrementar su estabilidad y facilitar su posible aplicación en aprendizajes continuos. En esta memoria proponemos dos posibilidades para la creación de comités: la creación de comités de evaluación de acción, donde cada uno de los aprendedores podrá estimar la utilidad de ejecutar cualquiera de las acciones posibles; y la creación de comités de evaluación de políticas, donde cada aprendedor estima la utilidad de un único intervalo dentro del espacio de acciones. A continuación se explica de manera detallada cada una de estas alternativas con sus ventajas e inconvenientes.

5.3. Comités de *evaluadores de acción*

Para la realización del comité de evaluadores de acción se construye un conjunto de N aprendedores independientes $(L^1, L^2, \cdots, L^N)$ (figura 5.3) del que únicamente un subconjunto de ellos $(L^i, L^j, \cdots, L^M)$ tomará parte en la toma de decisiones en cada episodio. Este subconjunto de M aprendedores ($M < N$) que toma parte en la toma de decisiones se determina aleatoriamente y será el mismo durante todo el episodio. Al empezar un nuevo episodio se escogerá otro subconjunto de aprendedores. El objetivo de construir un subconjunto de apren-

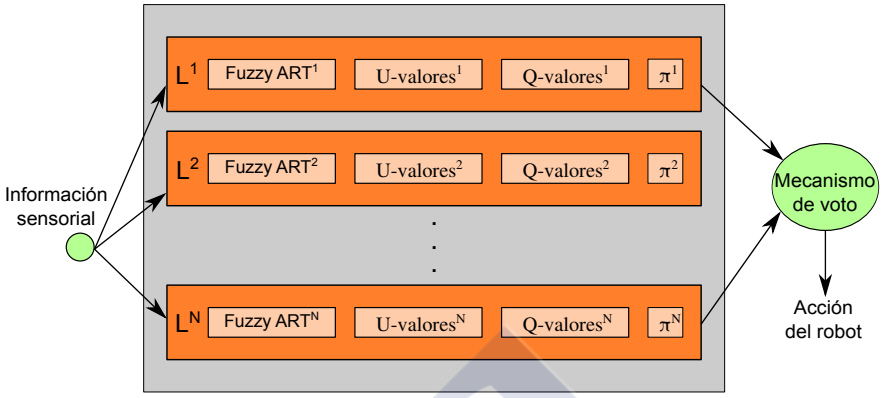


Figura 5.3: Esquema del aprendizaje mediante un comité de evaluadores de acción. Obsérvese que no todos los miembros del comité participan en el proceso de votación.

dedores de forma aleatoria es el lograr la independencia de los errores entre los miembros del comité. La principal característica de esta propuesta es que cada aprendedor l dividirá el espacio de acciones A en un conjunto independiente de P intervalos A^l, $\forall l = 1, \cdots, N$.

Dado que cada miembro del comité trabaja con intervalos de acciones, se puede observar un cambio importante en la definición de política de control cuando se trabaja con comités. Habrá una política π^l para cada aprendedor l, la cual proporciona un intervalo de acciones para un estado dado en lugar de una acción concreta. Por lo tanto una política de control π^l es una función que determina para cada posible estado del robot, el intervalo de acciones que parece adecuado para la tarea. Como cada aprendedor tiene un espacio de estados (S^l, red *Fuzzy ART*) y un conjunto de intervalos (A^l) diferentes, definimos su política de control (π^l) como:

$$\begin{aligned} \pi^l : S^l &\rightarrow A^l \\ s \in S^l &\rightarrow \pi^l(s) \in A^l \end{aligned} \tag{5.3}$$

La acción final ejecutada por el robot se selecciona tras un procedimiento de votación que considera los intervalos propuestos por los aprendedores pertenecientes al subconjunto M. Dentro del subconjunto de acciones más votado se elegirá aleatoriamente la acción que será ejecutada (figura 5.4). Aunque hay una gran variedad de estrategias para combinar diferentes fuentes de información [92], se seleccionó la elección por mayoría dado que es una de las estrategias más rápidas y conocidas.

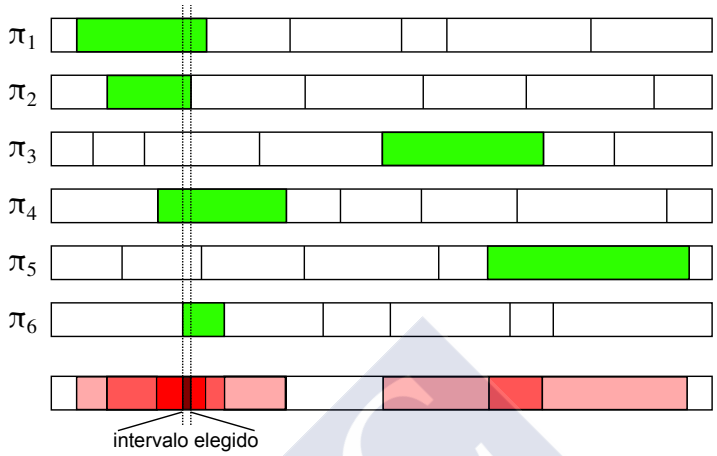


Figura 5.4: Ejemplo del procedimiento de votación. Cada aprendedor sugiere un intervalo de acciones. La acción ejecutada se elegirá aleatoriamente entre aquellas que reciban el máximo número de votos.

5.3.1. Función de utilidad de cada aprendedor

Para incrementar la robustez de nuestra propuesta y resolver el dilema exploración-explotación (sección 3.3), se considerará una nueva función de valor que determina cómo de buena es una política en particular para un estado dado. Representamos esta nueva función de valor como U (figura 5.3). La idea tras la inclusión de la función de utilidad es evitar cambiar las políticas de control si éstas son exitosas, aun cuando no sean la política óptima (egoísta). Por lo tanto la función U regulará cuándo se cambia la política votada por cada miembro del comité.

Esto nos lleva a diferenciar el cambio de política para cada estado de un aprendedor en función del valor U:

– Un aprendedor no cambiará su política en un estado dado mientras el intervalo propuesto no implique un fallo del robot.

– En caso de que la política sea "sospechosa" de haber causado algún fallo, el aprendedor perderá parte de su influencia en el proceso de votación, pero no cambiará su política y podrá recuperar su influencia si en otro episodio aparece y no es responsable del fallo.

– Si se considera a la política responsable del fallo se cambia en base a la política egoísta (ecuación 5.7).

El uso de las políticas egoístas para actualizar las políticas de control permite una recuperación más rápida de errores debido al uso de experiencias pasadas aprendidas por el robot. Por otro lado, se debe tener en cuenta el hecho de que cada vez que el robot comete un error, el uso de la función U limita el número de cambios y evita que el sistema caiga en grandes inestabilidades, ya que un aprendedor solamente cambiará su política en caso de que la acción sea considerada errónea.

Al principio del aprendizaje cada aprendedor calcula su política óptima. Dado que en ese momento los Q-valores son aleatorios, estas políticas serán también aleatorias. Ahora bien, en lugar de actualizar las políticas al final de cada episodio, cada política será modificada únicamente en aquellos estados donde se demuestra que hubo un error. Esto permite que la política de cada aprendedor del comité sea pseudo-egoísta, y por lo tanto subóptima, lo que facilita que haya cierta exploración y resulta beneficioso para el conjunto del comité.

El valor de U de cada una de las políticas que intervienen en el proceso de votación se calcula para cada estado al finalizar cada episodio, y representa el porcentaje de veces que la ejecución de una acción del intervalo propuesto por la política de control para dicho estado provocó el fallo que finalizó el episodio. También se tendrá en consideración un umbral δ, tal que si un valor U es mayor que dicho umbral se considerará que la política es errónea y deberá ser cambiada.

El cambio en las políticas de control se llevará a cabo cuando el valor de U sobrepase el valor umbral δ. En este caso las políticas de control $\pi^1, \cdots, \pi^N$ propuestas por cada aprendedor son actualizadas utilizando las políticas egoístas $\pi^{1*}, \cdots, \pi^{N*}$ (ecuación 5.7). Básicamente cada política será modificada en aquellas acciones que se consideran erróneas ($U^i(s) > \delta$), en este caso el nuevo intervalo de acciones coincidirá con la política egoísta (algoritmo 5.1).

Como consecuencia de la inclusión de U, podemos ponderar la importancia del intervalo propuesto por cada miembro del subconjunto de voto. La siguiente ecuación resume el proceso de votación:

$$a_t = \arg\max_a \left\{ \sum_1^N \delta[a \in \pi^l(s^l(t))] * \left(1 - U^l(s_t^l)\right) \right\}, \forall a \in A, \tag{5.4}$$

donde δ es la Delta de Kronecker:

$$\delta[a \in \pi^l(s^l(t))] = \begin{cases} 1 & \text{si } a \in \pi^l(s_t^l) \\ 0 & \text{si } a \notin \pi^l(s_t^l) \end{cases} \tag{5.5}$$

Algoritmo 5.1 Algoritmo de actualización de la política en el comité de evaluadores de acción

para $i = 1, \cdots, N$ **hacer**
  para todo $s_j^i \in S^i$ **hacer**
    si $U^i(s_j^i) > \delta$ **entonces**
      $\pi^i(s_j^i) = \pi^{i*}(s_j^i)$
      $U^i(s_j^i) = 0$
    fin si
  fin para
fin para

Por lo tanto, $U^l(s) = 1$ significa que la política de control l es inválida para el estado s y en consecuencia su voto no será tenido en cuenta. Por el contrario, $U^l(s) = 0$ significa que la política l parece adecuada para el estado s y por lo tanto su voto será tomado en consideración. Cualquier otro valor intermedio ponderará el voto reduciendo así su influencia.

5.3.2. Construcción de los intervalos de acciones

Recordemos que en esta propuesta cada uno de los aprendedores (figura 5.3) dispone de su propia partición del espacio de acciones en intervalos (figura 5.4). Cada partición A^l verifica las siguientes propiedades:

$$\begin{aligned} A^l &= \cdots \\ ... &= \{A^l(1) = [a_1^l, b_1^l), A^l(2) = [a_2^l, b_2^l), \cdots \\ ..., &A^l(P) = [a_P^l, b_P^l]\}, \forall l = 1, \cdots, N. \end{aligned} \tag{5.6}$$

– A^l cubre todo el espacio de acciones $A, \forall l = 1, \cdots, N$:

$$\bigcup_j A^l(j) = A.$$

– A^l es una colección disjunta de intervalos, $\forall l = 1, \cdots, N$:

$$\bigcap_j A^l(j) = \emptyset.$$

El número de intervalos P es el mismo para todas las particiones:

$$P = |A^i| = |A^j|, \forall i, j = 1, \cdots, N.$$

Cada aprendedor l del comité aprenderá una función de utilidad $Q^l(S^l \times A^l)$, de tal forma que $Q^l(s \in S^l, A^l(j))$ representa el tiempo que transcurrirá antes del fallo si el robot ejecuta alguna de las acciones incluidas en $A^l(j)$ y a continuación sigue una política de control óptima. En este sentido, la política de control óptima, o egoísta, para cada aprendedor l se puede obtener de la forma habitual, esto es, a partir de los Q-valores:

$$\pi^{l*}(s \in S^l) = \arg\max_j \{Q(s, A^l(j))\}, \forall l = 1,..,N, \ \forall j = 1,..,P. \tag{5.7}$$

Siendo N el número de aprendedores, y P el número de intervalos en los que se particiona el espacio de acciones.

5.3.3. Actualización de las funciones de valoración

Como se comentó anteriormente, la acción que el robot ejecuta en cada instante viene determinada por un proceso de votación en el que intervienen dos funciones Q y U (figura 5.3). Ambas funciones de valoración son independientes para cada aprendedor del comité: $Q^1, \cdots, Q^N$ y $U^1, \cdots, U^N$.

Tras cada episodio el Q-valor de todos los aprendedores que participaron en el proceso de votación será actualizado. Esto implica que el algoritmo de aprendizaje I_Tbf descrito en la sección 3.1 debe ser modificado para incluir estos nuevos elementos. El algoritmo 5.2 muestra las adaptaciones realizadas al algoritmo I_Tbf para el aprendizaje con comités de evaluadores de acción. Recordemos que se puede obtener el valor de Tbf a partir de los Q-valores siguiendo la ecuación 3.7.

En este algoritmo a_{t-m} es la acción ejecutada en el instante $t-m$, y $A^l_{a_{t-m}}$ es el intervalo de A^l que contiene a_{t-m}. El parámetro λ es un parámetro que pondera la importancia del episodio actual frente a la experiencia pasada y acumulada en los Q-valores. A valores más altos de λ mayor será la influencia de las últimas interacciones en los Q-valores. Se recomienda valores bajos de λ cuando los refuerzos o las lecturas sensoriales contienen gran cantidad de ruido.

La actualización de la función de utilidad U es directa: aquellas políticas que votaron por acciones ejecutadas lejos del error verán disminuido su valor U (hasta el valor mínimo de 0). Por el contrario, a aquellas políticas que votaron por acciones ejecutadas justo antes del error se les aumentará el valor U (hasta el valor máximo de 1). Por último, las políticas que votaron por acciones que no se ejecutaron no verán alterado su valor U. Esta actualización de U provoca que el robot tienda a repetir aquellas acciones ejecutadas en el pasado y que no llevaron directamente a un error y a evitar aquellas acciones cercanas al error.

Algoritmo 5.2 Algoritmo de aprendizaje I_Tbf con comités de evaluadores de acción

repetir

Primera etapa: Selección de aprendedores

Seleccionar aleatoriamente un *subconjunto de voto* de M aprendedores

Segunda etapa: Recolectando información

$m = 0$

repetir

Observar el estado actual en cada aprendedor del *subconjunto de voto*, s_t^l

Seleccionar mediante un proceso de votación una acción a_t para ser ejecutada

Ejecutar acción a_t, observar el nuevo estado s_{t+1}^l para cada miembro del *subconjunto de voto* y el valor de refuerzo r_t

$m \leftarrow m+1$

hasta que $r_t < 0$ **o** $m >= m_{max}$

Tercera etapa: Actualizar los Q-valores

1. Actualizar el tiempo a fallo de cada aprendedor del *subconjunto de voto*:

para $k = 0, 1, \cdots, m$ **hacer**

si $k = 0$ **entonces**

$$Tbf^l = \begin{cases} T & \text{si } r_{t-k} < 0 \\ Tbf^l(s_{t-k}) & \text{en otro caso} \end{cases}$$

si no

$Tbf^l \leftarrow \lambda * (Tbf^l + T) + (1-\lambda) * Tbf^l(s_{t-k})$

fin si

fin para

2. Actualizar el Q-valor de cada aprendedor del *subconjunto de voto*:

$$\Delta Q_t^l(s_{t-m}, A_{a_{t-m}}^l) = \beta_L * (-e^{-Tbf^l/50T} - Q_t^l(s_{t-m}, A_{a_{t-m}}^l))$$
$$\forall l \in subconjunto\ de\ voto$$

hasta que Convergencia alcanzada

Hay que recordar que, para mantener la independencia de los errores entre miembros del comité, únicamente aquellos aprendedores que participaron en el proceso de votación verán sus valoraciones cambiadas.

5.3.4. Aplicación experimental

Para probar el sistema de aprendizaje se ha seguido una estrategia similar a la empleada en experimentos anteriores. El robot móvil, un Pioneer P3-DX, deberá aprender a ejecutar la tarea seguir pared con una configuración sensorial consistente en un láser SICK LMS-200 y un anillo de 16 ultrasonidos. La velocidad lineal se mantiene constante a 15,24 cm/s y el ciclo de control se ejecuta cada 300 ms. La función de refuerzo será la misma que se usó en anteriores aprendizajes del comportamiento seguir pared. La aplicación de este algoritmo se presentó inicialmente en [55, 94]. En primer lugar hemos realizado un análisis del sistema en simulación, que se pasa a explicar a continuación.

En el primer conjunto de experimentos se escalaron los valores provenientes de los sensores del intervalo $[0,\ 8]$ al intervalo $[0,\ 1)$ mediante la función:

$$I_i = \frac{1}{1+input_i} - \frac{1}{9}. \tag{5.8}$$

Es importante destacar que en este caso utilizamos toda la información procedente del láser sin agruparla en haces de 22,5°como hacíamos hasta ahora. Los parámetros utilizados para estas pruebas fueron: coeficiente de aprendizaje del algoritmo *I_Tbf* $\beta_1 = 0,25$, $\lambda = 0,5$. El parámetro de vigilancia de cada red *Fuzzy ART* se selecciona aleatoriamente en el rango $[0,86,\ 0,92]$, y finalmente el coeficiente de aprendizaje de las redes *Fuzzy ART* es $\beta_2 = 0$. Cada experimento contó con un comité de 435 aprendedores, cada uno de los cuales dividía el espacio de acciones en 7 intervalos aleatorios. En lo referente a la función U, el umbral δ para considerar errónea una política se estableció en $0,1$. Para considerar a una acción cercana al fallo se eligió aleatoriamente, tras cada fallo, un número de ciclos de control n en el intervalo $[0,\ 35]$. Por lo tanto, toda acción ejecutada n ciclos de control antes del fallo se considera causante del mismo e incrementará el valor de U.

Las pruebas ejecutadas en simulación proporcionaron un tiempo medio de 14:33 minutos y una desviación de 8:48 minutos para un total de 30 experimentos. Estos tiempos son mucho mejores que los conseguidos con la versión simple de *I_Tbf* y *Fuzzy ART*: 22:18 de tiempo medio de aprendizaje y 14:10 de desviación. Ahora bien, esta reducción del tiempo de aprendizaje es aún más significativa si consideramos que en el caso del comité se trabaja con las 181 componentes del escáner láser, en lugar de agrupar esta información en haces de 22,5°, que es lo que se ha hecho al aplicar el algoritmo *I_Tbf* combinado con la red *Fuzzy ART* (sección 4.3.1).

Durante las pruebas de este sistema hemos querido comprobar el efecto sobre el aprendizaje del uso de una función de preprocesado sensorial más eficiente. Por lo tanto, tras el primer conjunto de experimentos se llevó a cabo otra batería de pruebas donde el preprocesado de la información sensorial se cambió a la función:

$$I_i = \frac{(8 - input_i)^2 * e^{\frac{-input_i}{1.5}}}{64}. \tag{5.9}$$

Dado que el objetivo de esta tesis no es el realizar un análisis exhaustivo de distintos algoritmos de aprendizaje, sino el lograr un sistema de aprendizaje que funcione de manera continua y en un robot real, no se han repetido las pruebas realizadas en capítulos anteriores con esta nueva función. Para los experimentos realizados con este nuevo preprocesado todos los parámetros del sistema se mantuvieron con los mismos valores, a excepción del intervalo de elección aleatoria de la vigilancia de las redes *Fuzzy ART*, el cual fue ampliado a $[0,8,\ 0,92]$. Tras 30 experimentos se obtuvo un tiempo medio de aprendizaje de 06:36 minutos y una desviación de 04:27 minutos. Este resultado permite destacar la importancia de una buena percepción sensorial para el aprendizaje por refuerzo, ya que tanto el tiempo de aprendizaje como la desviación estándar se han reducido de forma muy apreciable.

En general, en estos experimentos se observa la importancia de la función U, la cual permite que los distintos aprendedores voten o no en función de sus aciertos o errores pasados. Mediante el uso de U se consigue una convergencia más rápida al comportamiento deseado.

Dado que el algoritmo de aprendizaje muestra tan buenos resultados en simulación se decidió dar el salto al robot real y probar el algoritmo en un entorno real. En este caso el robot aprenderá también la tarea de seguir la pared. En la figura 5.5 se puede observar el progreso del aprendizaje de un robot real moviéndose en un entorno real, y como en pocas vueltas es capaz de aprender una política de control válida. Lo más relevante de nuestra propuesta es que el robot empieza sin ningún conocimiento previo ni de la tarea ni del entorno. Esto se puede apreciar claramente en la primera vuelta que el robot realiza al entorno, mostrada en la figura 5.5(a).

Cada vez que el robot recibe refuerzo negativo, detiene su avance y retrocede hasta alcanzar una posición segura, caracterizada por la ausencia de refuerzo negativo, desde la que continúa el aprendizaje. Ha sido necesario el desarrollo de un controlador que permita mover al robot a una posición segura. No obstante, dicho controlador de recuperación es un aspecto delicado del algoritmo, el simple hecho de recolocar al robot en una nueva posición de partida válida no es algo trivial y no siempre funciona de manera eficiente, lo que provoca un retraso

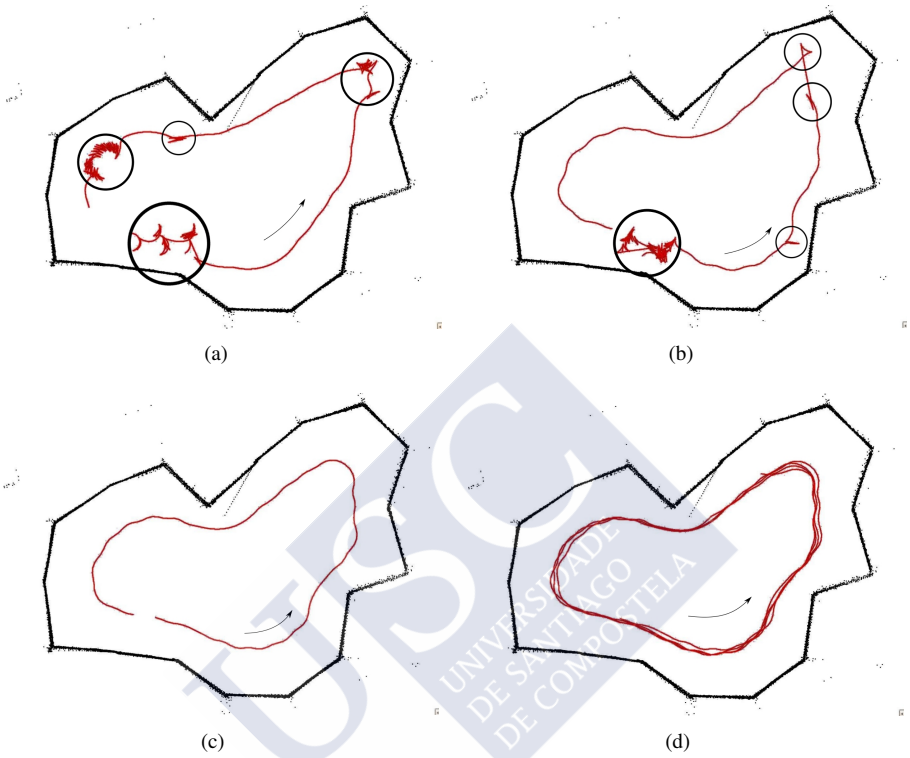


Figura 5.5: Trayectorias del robot real mientras aprende la tarea seguir pared derecha en un entorno real: a) Primera vuelta al entorno. El robot retrocede cada vez que recibe un refuerzo. b) Tercera vuelta al entorno, el robot no recibe tantos refuerzos negativos. c) En la cuarta vuelta es capaz de ejecutar la tarea sin error. d) Últimas vueltas con el comportamiento ya aprendido antes de la finalización del aprendizaje.

en el aprendizaje. El definir el controlador de recuperación, así como la función de refuerzo, son tareas que dependen del comportamiento que se busca, y que por lo tanto impiden una rápida adaptación del sistema en caso de querer aprender diferentes comportamientos. En la siguiente sección se presenta una propuesta diferente donde tanto el controlador de recuperación como la señal de refuerzo son proporcionadas directamente por un observador humano.

5.4. Comité de *evaluadores de políticas*

Dado que ya hemos comprobado la aplicabilidad de la construcción de un comité para el aprendizaje en el robot real, queremos también orientar su diseño hacia el aprendizaje continuo. En el nuevo comité que presentamos en esta sección buscamos dos aspectos: la capacidad de introducir nuevo conocimiento sin desestabilizar lo ya aprendido, y la posibilidad de actualizar las valoraciones sin que ocurra un fallo.

La propuesta que se presenta en este apartado utiliza, al igual que en el caso anterior, un comité de aprendedores. En el comité de evaluadores de acción presentado anteriormente, cada aprendedor l aprende una función de utilidad Q^l para cada par estado-intervalo de acciones $Q^l(S^l \times A^l)$. Por este motivo la política de control propuesta por cada aprendedor del comité evoluciona a lo largo del tiempo. En este apartado describimos una nueva propuesta en la que la política de control propuesta por cada aprendedor no evoluciona a lo largo del proceso de aprendizaje. En este caso cada aprendedor del comité simplemente evalúa una determinada política de control, por lo que no tendrá la capacidad de evaluar la utilidad de cada acción. Gracias a esto con este algoritmo se necesitará almacenar menos información (una única política de control frente a todos los intervalos del espacio de acciones de la propuesta anterior). Otras ventajas relativas al aprendizaje proporcionadas por esta propuesta son una mayor facilidad para su implementación en el robot real y la simplificación del algoritmo.

En este caso cada uno de los N aprendedores evalúa una política de control, $\pi^1, \cdots, \pi^N$, creada inicialmente al azar. Cada una de estas políticas propone un intervalo de acciones para cada estado:

$$\begin{aligned} \pi^l : S^l &\rightarrow A^l \\ s \in S^l &\rightarrow \pi^l(s) \in A = [a,b), a \in A, b \in A, a < b. \end{aligned} \tag{5.10}$$

Cada aprendedor l también construye su propia representación del mundo mediante una red *Fuzzy ART* (figura 5.6). Cada vez que aparece un nuevo estado se le asociará un intervalo de acciones creado de forma aleatoria y que no cambiará durante todo el proceso de aprendizaje.

Habrá dos conjuntos de políticas (figura 5.6). El primero de ellos interviene en el procedimiento de voto para elegir la acción a ejecutar, es el *Conjunto de políticas de decisión*. El segundo conjunto de políticas, llamado *Conjunto de políticas de observación*, no interviene en el proceso de votación, pero observa el resultado de las acciones del robot. A lo largo de esta sección se irá viendo el funcionamiento de cada uno de ellos.

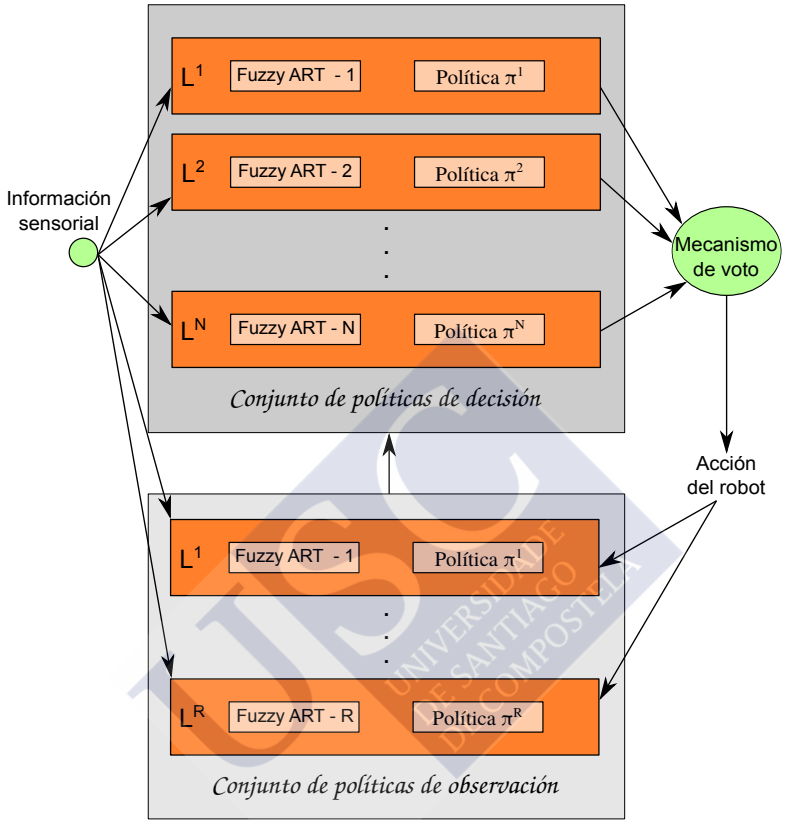


Figura 5.6: Esquema general de la propuesta para aprendizaje en el robot real. El conjunto de políticas de decisión determina qué acción ejecuta el robot en cada instante. El conjunto de políticas de observación va construyendo nuevas políticas a partir de las acciones que no han tenido refuerzo negativo. Cada aprendedor genera su propio espacio de estados.

5.4.1. Actualización de los Q-valores

Dado que cada una de las políticas del comité se construye de manera aleatoria será necesario calcular la idoneidad de cada una de ellas. Por lo tanto se creará una función de utilidad Q para cada una de las políticas con valoraciones aleatorias $Q^l(s) \in [-1,-0,95], \forall s \in S^l$. A continuación se iniciará la exploración del entorno ejecutando las acciones determinadas mediante un proceso de votación. En esta votación participan todas las políticas pertenecientes al *Conjunto de políticas de decisión* (figura 5.6). La acción que el robot ejecuta en cada instante

es aquella que predice un mejor resultado a partir de los datos obtenidos hasta el momento, esto es, el robot ejecutará la acción con mayor Q-valor medio:

$$a_t = arg\_max_{\forall a \in A} \frac{\sum_{l=1}^{l=N} \delta(a \in \pi^l(s_t^l)) \cdot (1 + Q^l(s_t^l))}{\sum_{l=1}^{l=N} \delta(a \in \pi^l(s_t^l))}, \tag{5.11}$$

donde δ es la delta de Kronecker y los valores de $Q^l(s)$ están acotados en $[-1,0]$. En la ecuación anterior se puede observar que cada miembro del comité, aprendedor l, vota por las acciones sugeridas por su correspondiente política π^l y su voto será ponderado por el valor Q^l.

El algoritmo 5.3 muestra la estrategia de actualización de los Q-valores para el uso de un comité de evaluadores de políticas.

La propuesta que presentamos aquí no necesita ser rediseñada en caso de que se quiera generalizar al aprendizaje continuo, ya que las valoraciones serán actualizadas aunque no haya error sin necesitar esperar a que finalice el episodio. A medida que el robot se mueve e interacciona con su entorno, los Q-valores correspondientes a las diferentes políticas de control son actualizados en base a la valoración media del estado siguiente. Observando el algoritmo aquí presentado se pueden distinguir tres situaciones diferentes. Al principio el robot empieza a moverse pero no actualiza los Q-valores hasta que no ha transcurrido un mínimo intervalo de tiempo (M ciclos de control). Tras ese periodo el robot continúa moviéndose mientras actualiza los Q-valores de cada estado teniendo en cuenta que durante los M ciclos anteriores no se cometió ningún error - esta actualización se corresponde con el aprendizaje continuo (paso 8 del algoritmo 5.3). Finalmente, cuando el robot recibe un refuerzo negativo se actualizan los Q-valores de aquellos estados almacenados en el conjunto de experiencias teniendo en cuenta el tiempo real transcurrido hasta el error (paso 9 del algoritmo 5.3).

Es importante destacar que únicamente aquellas políticas que votaron por la acción finalmente ejecutada ven modificadas sus valoraciones. La predicción del tiempo a fallo realizada en cada instante es aproximada mediante el máximo valor medio de los tiempos esperados (Q-valores) de todas las políticas. También hay que resaltar la diferenciación entre los dos conjuntos de parámetros β_1, λ_1 y β_2, λ_2. Esto se debe al hecho de que los refuerzos negativos son muy infrecuentes y por lo tanto, su influencia en los Q-valores debe ser más alta que en aquellos casos en los que el refuerzo es nulo, razón por la que habitualmente β_1 y λ_1 deben ser menores que β_2 y λ_2.

Algoritmo 5.3 Algoritmo de aprendizaje I_Tbf con comités de evaluadores de políticas

$m = 0$
s_t: $s[m] = s_t^1, \cdots, s_t^l, \cdots, s_t^N$, de manera que $s[m,1] = s_t^1, \cdots, s[m,l] = s_t^l, \cdots, s[m,N] = s_t^N$
repetir
  Ejecutar la acción a_t de acuerdo a la elección del comité: $a[m] = a_t$
  $u[m] = -50 * T * Ln\left(-\frac{\sum_{l=1}^{l=N} \delta(a_t \in \pi^l(s_t^l)) \cdot Q^l(s_t^l)}{\sum_{l=1}^{l=N} \delta(a_t \in \pi^l(s_t^l))}\right)$
  Observar el nuevo estado $s[m+1] = s_{t+1}^1, \cdots, s_{t+1}^l, \cdots, s_{t+1}^N$ y el refuerzo r_t : $r[m] = r_t$
  Incrementar el índice m: $m \leftarrow m+1$
  si $r_t >= 0$ **y** $m >= M$ **entonces**
    para $k = m-1, m-2, \cdots, 1$ **hacer**
      si $k = m-1$ **entonces**
        $Tbf \leftarrow u[k]$
      si no
        $Tbf \leftarrow \lambda_1(Tbf + T) + (1-\lambda_1)u[k]$
      fin si
    fin para
    $\Delta Q_t^l(s[0,l]) = \beta_1 \delta(a[0], \pi^l(s[0,l])) \cdot (-e^{-Tbf/50T} - Q_t^l(s[0,l])), \forall l = 1, \cdots, N$
    Borrar $s[0], a[0], u[0], r[0]$ y desplazar el resto de elementos: $m \leftarrow m-1$
  fin si
hasta que $r_t < 0$
para $j = 0, 1, \cdots, m-1$ **hacer**
  para $k = m-1, m-2, \cdots, j$ **hacer**
    si $k = m-1$ **entonces**
      $Tbf \leftarrow T$
    si no
      $Tbf \leftarrow \lambda_2(Tbf + T) + (1-\lambda_2)u[k]$
    fin si
  fin para
  $\Delta Q_t^l(s[j,l]) = \beta_2 \delta(a[j], \pi^l(s[j,l])) \cdot (-e^{-Tbf/50T} - Q_t^l(s[j,l])), \forall l = 1, \cdots, N$
  Borrar $(s[j], a[j], u[j], r[j])$
fin para

5.4.2. Incorporando nuevo conocimiento

Una característica necesaria en un sistema de aprendizaje continuo es la eliminación y la incorporación de nuevo conocimiento en cualquier fase del ciclo de vida del robot. Para permitir la eliminación e inserción de políticas se ha decidido incorporar al sistema un segundo comité de evaluadores de políticas (figura 5.6) llamado *Conjunto de políticas de observación*.

Algoritmo 5.4 Modificación de los intervalos de acciones propuestos por los miembros del comité de observación

repetir
  Observar la acción seleccionada por las políticas del comité de decisión, a_t
  Determinar el estado actual de cada política del comité de observación, $s_t^l, \forall l = 1, \cdots, R$, y el refuerzo r_t
  si $r_t >= 0$ **entonces**
    si s_t^l es nuevo **entonces**
      Inicializar la política de control: $\pi_t^l = [a,b]$, donde $a = a_t$ y $b = a_t$
    si no
      si $a_t \notin \pi^l(s_t^l) = [a,b)$ y $a_t < a$ **entonces**
        $a = a_t$
      si no, **si** $a_t \notin \pi^l(s_t^l) = [a,b)$ y $a_t > b$ **entonces**
        $b = a_t$
      fin si
    fin si
  fin si
fin repetir

Parece razonable incluir políticas que no generen inestabilidades en el comportamiento del robot, es por ello que el comité de políticas de observación irá construyendo políticas que almacenen las acciones que dieron buenos resultados en el pasado.

Cada vez que el robot alcanza una cierta cantidad de errores cometidos la política más vieja del comité de observación es transferida al comité de decisión. El algoritmo 5.4 muestra el proceso que siguen las políticas pertenecientes al comité de observación. En cada instante, y si no se recibe refuerzo negativo, el intervalo de acciones propuesto para el estado actual se expande para incluir la acción ejecutada. De esta manera, cada una de las políticas del comité de observación irá construyendo su propio espacio de estados y focalizará su atención en aquellas acciones que parecen ser correctas para cada estado, por lo que cada política propondrá un intervalo de acciones que incluye todas las acciones que resultaron satisfactorias en instantes anteriores del aprendizaje.

5.4.3. Aprendizaje interactivo. Proporcionando realimentación al robot.

Recordemos que cuando el sistema de aprendizaje mediante un comité de evaluadores de acción se aplicó para el aprendizaje en el robot real, cada vez que el robot cometía un error existía un controlador de recuperación que tomaba el control y recolocaba al robot en una posición segura desde la que poder continuar el aprendizaje. Dicho controlador debe ser específico para la tarea que el robot esté aprendiendo, por lo que esta solución complica seriamente el poder aplicar el sistema de aprendizaje a diversos comportamientos en un robot de servicio operando en entornos cotidianos. Lo mismo ocurre con la función de refuerzo, que debe ser diseñada de manera específica y cuidadosa para cada comportamiento. Para resolver estos problemas haremos que tanto la función de refuerzo como la recolocación del robot tras un error sean tarea de un observador humano.

La idea de utilizar el refuerzo proporcionado por un observador humano se viene usando desde hace tiempo con la intención de permitir la adaptación de los robots sin la necesidad de un experto. Todas las publicaciones coinciden en que para disponer de robots de servicio capaces de trabajar en entornos cotidianos y de realizar una diversidad de tareas, será necesario que usuarios sin conocimientos de programación ni de robótica sean capaces de interaccionar con el robot para guiar su entrenamiento hacia sus necesidades particulares.

Dentro del aprendizaje interactivo existen dos ramas principales en función del rol que juegue el humano. Por un lado el observador humano puede limitarse a indicar el refuerzo o, como ocurrirá en nuestro caso, controlar el robot tras un error; pero en ningún caso el humano condiciona la selección de acciones para encontrar aquellas que resuelven la tarea que se está aprendiendo. La otra opción es que cuando el humano guíe al robot este guiado pueda servir para condicionar el proceso de aprendizaje y guiar su exploración del espacio de acciones. Esta última rama se considera aprendizaje por demostración (sección 1.4.1).

Dentro del grupo de propuestas donde el humano no influye en la exploración de estados podemos encontrar varios trabajos. En [125] se examina la hipótesis de que el refuerzo proporcionado por un humano sea compatible con la señal de refuerzo tradicional del aprendizaje por refuerzo. Los autores han realizado varios experimentos en un entorno simulado donde un humano debe interaccionar mediante una interfaz gráfica con un agente que implementa un algoritmo de aprendizaje *Q-learning*. El objetivo es que el robot aprenda a cocinar un pastel por sí solo. Las principales conclusiones son que los algoritmos de aprendizaje por refuerzo deben tener en cuenta las siguientes consideraciones:

– Además de suministrar el refuerzo al robot, a los usuarios les gustaría guiar al robot hacia la acción que creen correcta, así como dar refuerzos anticipados. En la literatura sobre aprendizaje por refuerzo se ha estudiado en profundidad el efecto de los refuerzos retardados [59], la aplicación de refuerzos anticipados no forma parte del modelo de aprendizaje por refuerzo clásico.

– Los usuarios prefieren dar refuerzo positivo antes que negativo, probablemente reflejando sus opiniones sobre la motivación en el aprendizaje en humanos, o a que sienten que el refuerzo negativo que proporcionan es ignorado por el robot.

– A medida que avanza el aprendizaje los usuarios van creando un modelo mental del agente y, en paralelo va cambiando su estrategia para enseñar al agente a realizar la tarea. El aprendizaje por refuerzo clásico no tiene en cuenta el hecho de que un profesor benevolente ajustará su estrategia de entrenamiento para ajustarse mejor al alumno.

El estudio presentado en [9] analiza la forma en la que los usuarios proporcionan un refuerzo multimodal tanto positivo como negativo, mediante voz, gestos o tacto, mientras enseñan a un robot mascota diversas tareas. Para no estresar ni aburrir al usuario las tareas a enseñar son sencillos juegos como ”conecta cuatro” o ”parejas”. En sus resultados destacan que la mayoría de las interacciones se realizan mediante voz (78,37 %), seguido de contacto físico (20,92 %) y gestos (0,71 %). Es importante tener presente que dado que los experimentos se realizaron con un perro robótico, los usuarios se pudieron ver influenciados por la forma del robot a la hora de interactuar con él. Cada usuario tiene una manera diferente de proporcionar la realimentación, por lo que en lugar de funciones preprogramadas para interpretar el refuerzo, el robot se debe adaptar a la forma que cada usuario tiene de proporcionar la realimentación. Aprender a interpretar el refuerzo solo es útil si el robot es entrenado por un mismo usuario y si dicho usuario mantiene su estilo de realimentación en todas las tareas, lo que sí parece ocurrir vistos los resultados de los experimentos mostrados.

El trabajo de Knox y Stone [64] [65] se centra en la transferencia de conocimiento de los humanos a las máquinas, con el fin de acelerar el aprendizaje y reducir el coste del mismo. Para ello han creado el marco de trabajo TAMER (*Training an Agent Manually via Evaluative Reinforcement*), el cual se construye siguiendo las bases del aprendizaje por refuerzo inverso [86]. El TAMER se basa en modelar el refuerzo proporcionado por el humano para que el sistema escoja aquellas acciones que se espera proporcionen un mayor refuerzo.

Otros autores presentan una combinación de aprendizaje por demostración (sección 1.4.1) con aprendizaje por refuerzo donde es el usuario el que proporciona el refuerzo al robot. El aplicar una realimentación proveniente de un humano al aprendizaje proporciona ventajas, como un menor tiempo de aprendizaje al hacer que el robot necesite explorar menos estados [122], o el poder evitar el diseño de una señal de refuerzo.

En [76] muestran una estrategia de aprendizaje basada en dos fases: 1) demostración, percepción y representación de la tarea, 2) reproducción y refinamiento. Durante la fase de reproducción y refinamiento el usuario puede proporcionar un refuerzo valorando la ejecución del robot: objetivo alcanzado (+100), excelente (+10), bien (+5), mal (-5) y terrible (-10). Otra forma de interactuar del usuario es indicando las acciones que quiere que el robot ejecute. En este mismo trabajo se muestran los resultados alcanzados cuando se enseña a un brazo robótico a voltear un objeto y comparan el rendimiento del aprendizaje a medida que se añaden más módulos al sistema. Se concluye que utilizando la demostración y la realimentación de forma combinada durante el proceso de aprendizaje el tiempo de convergencia se ve reducido.

En [134] se utiliza una estructura híbrida que utiliza tanto comandos del usuario como un sistema de aprendizaje por refuerzo basado en *Q-learning* para controlar el robot. El sistema presentado integra las instrucciones del usuario y las políticas de control en un Proceso de Decisión semi-Markoviano (SMDP). Las políticas se representan como soluciones a dicho SMDP. Para evitar fallos catastróficos y filtrar comandos inconsistentes del usuario, se añade un conjunto de comportamientos que sobrescriben cualquier acción que llevaría al robot fuera de los límites de seguridad. El usuario puede proporcionar comandos tanto de bajo nivel, en la forma de acciones que el robot ejecutará, como de alto nivel, añadiendo subobjetivos al SMDP. La señal de refuerzo que recibe el algoritmo de aprendizaje es la suma de un refuerzo generado a partir de las acciones indicadas por el humano y un refuerzo específico para la tarea. Se muestran resultados experimentales con robots reales. En uno de ellos el robot debe aprender una ruta en un entorno interior, el control de bajo nivel se realiza mediante un planificador de rutas basado en campos de potencial. En este trabajo muestran cómo mediante las instrucciones dadas por el humano, el robot aprende la ruta más rápido, al mismo tiempo el sistema es capaz de ignorar malas indicaciones como ciclos o acciones demasiado arriesgadas.

En nuestro caso se ha decidido que el refuerzo provenga de un observador humano que esté supervisando lo que el robot hace en cada momento, y que en ningún caso el humano condicione la exploración del robot. Este observador será capaz de penalizar al robot simplemente presionando un botón de un *joystick* inalámbrico (figura 5.7). Esta acción será suficiente pa-

Figura 5.7: *Joystick* inalámbrico utilizado por el observador humano para proporcionar la señal de refuerzo al robot.

ra indicarle al robot que lo que está haciendo no es satisfactorio desde el punto de vista del observador. Es importante remarcar el hecho de que esta forma de indicar el refuerzo es altamente no determinista, el mismo usuario puede dar al robot un refuerzo negativo en ciertas situaciones y más adelante permanecer impasible en otros escenarios muy similares. Por otra parte, es posible que el observador humano cambie de parecer durante el aprendizaje sobre el comportamiento deseado. Por lo tanto nuestro algoritmo debe hacer frente a refuerzos inconsistentes y nada sistemáticos. Cuando el usuario pulsa el botón de refuerzo negativo el robot entra en un estado pasivo y se transfiere el control al *joystick*, de manera que el usuario puede mover el robot y colocarlo en una nueva posición de reinicio válida para retomar el aprendizaje. En ese momento se pulsa un segundo botón que devuelve el control al robot y el aprendizaje continúa.

No entra dentro de los objetivos de este trabajo el combinar aprendizaje por demostración con aprendizaje por refuerzo, por lo tanto el algoritmo no aprenderá lo que el robot haga durante el tiempo que el usuario tiene el control. Decidimos proceder de esta manera para resaltar la habilidad del algoritmo de aprendizaje por refuerzo de obtener aprendizajes rápidos a partir de la interacción directa entre el robot y el entorno, a pesar de que la forma en la que el usuario proporciona el refuerzo cambia durante el proceso de aprendizaje.

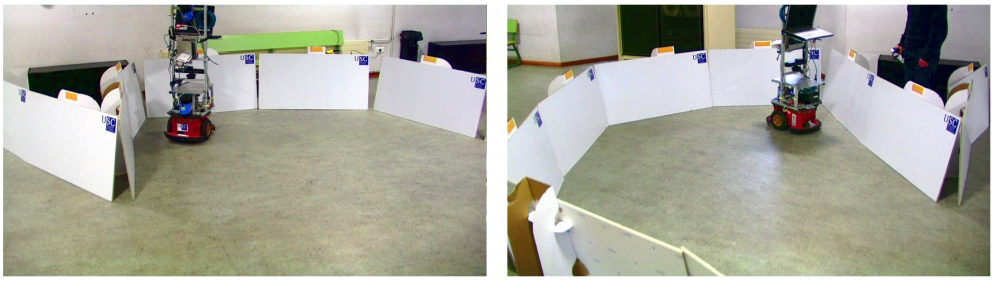
5.4.4. Aplicación experimental

Al igual que hicimos con el aprendizaje basado en un comité de evaluadores de acción, se realizaron una serie de pruebas para evaluar el rendimiento de la propuesta de aprendizaje mediante un comité de aprendedores de políticas. Debido a los buenos resultados mostrados en el robot real obtenidos mediante el comité de evaluadores de acción, las pruebas aquí mostradas se realizaron directamente en el entorno real, sin ningún paso previo por simulación. Se usó el mismo robot Pioneer P3-DX utilizado en pruebas anteriores. En esta ocasión se sustituyó el láser SICK LMS-200 por un modelo SICK LMS-100. El LMS-100 proporciona diversas ventajas sobre el LMS-200 para su aplicación en un robot móvil: es más compacto, tiene un menor consumo energético y abarca un arco de 270° – proporciona un vector de 541 lecturas de distancia, una lectura cada medio grado en el intervalo $[-135°,\ 135°]$. Los parámetros del algoritmo I_Tbf son: $\lambda_1 = 0,99$, $\beta_1 = 0,05$, $\lambda_2 = 0,15$ y $\beta_2 = 0,95$. La velocidad lineal se mantuvo constante en $15,24$ cm/s, y el espacio de acciones a escoger por el robot es un conjunto de acciones equidistantes por una décima de grado que discretizan las velocidades angulares en el rango $[-0,8,\ 0,8]$ rad/s, 900 acciones en total. Los resultados de esta investigación fueron publicados en [99, 97].

En todas las pruebas el robot empieza sin ningún conocimiento previo ni del entorno ni de la tarea, la tabla de Q-valores está inicializada a valores aleatorios en $[-1,\ -0,95]$ y las redes *Fuzzy ART* no contienen ningún estado. Dado que en esta propuesta el refuerzo es proporcionado por un humano y es claramente no determinista, la noción de tarea se difumina. Durante las pruebas aquí mostradas el robot aprendió un comportamiento de navegación teniendo como referencia la pared derecha en diferentes entornos. Hay que volver a señalar que durante todas estas pruebas el robot está aprendiendo de manera continua, incluso en ausencia de refuerzos negativos.

La primera prueba tuvo lugar en el entorno mostrado en la figura 5.8, que consistía en un pequeño espacio acotado por planchas de espuma. La figura 5.9 muestra las trayectorias seguidas por el robot en las primeras vueltas al circuito. Las zonas donde el robot cometió un error se marcan con un círculo. Tal y como se puede ver en las imágenes, los errores se concentran en los primeros instantes del aprendizaje, y a partir de la sexta vuelta el robot no comete ningún error.

La segunda prueba consistió en dos partes. En la primera parte el robot aprendió a moverse en un entorno parcialmente obstruido con unos obstáculos, el vestíbulo del Departamento de Electrónica e Computación (figura 5.10). Como se puede observar en la figura 5.11 el robot



A) B)

Figura 5.8: Entorno en el que el robot deberá aprender a seguir pared situada a su derecha.

aprendió rápidamente cómo moverse en ese entorno. En la segunda parte del experimento se eliminaron los obstáculos que delimitaban el entorno y se le permitió al robot moverse libremente por el edificio (figura 5.12). Es reseñable el hecho de que en esta segunda parte, pese a que el robot debía circular por un entorno muy diferente, con pasillos, puertas y una disposición muy diferente de obstáculos, no recibió muchos refuerzos negativos.

Estos buenos resultados también se muestran en un tercer experimento realizado en un entorno cotidiano, en particular en los pasillos del Departamento de Electrónica e Computación de la Universidade de Santiago de Compostela (figura 5.13). En este caso no se acotó el movimiento del robot durante las fases tempranas del aprendizaje, por lo que el robot se encontró desde el comienzo del aprendizaje con personas, muebles y equipamiento.

La figura 5.14 muestra las trayectorias ejecutadas por el robot durante su aprendizaje en el entorno del departamento. Al igual que ocurrió en el entorno acotado, la mayoría de los errores, marcados con círculos en la figura, se cometieron al principio del aprendizaje. Al llegar al vestíbulo en la primera vuelta (figura 5.14.(a)) se puede observar cómo el comportamiento no fue correcto, esto se debe a que la zona del vestíbulo es muy diferente a los pasillos por donde el robot había circulado anteriormente, por lo que se generan estados nuevos y el robot debía aprender la acción correcta para cada uno de ellos. Como se ve en la figura 5.14.(b) la segunda vuelta la realizó prácticamente sin errores y con una trayectoria más suave.

Es importante recordar el hecho de que el robot aprendía – actualizando los Q-valores – continuamente, incluso en la ausencia de refuerzo negativo.

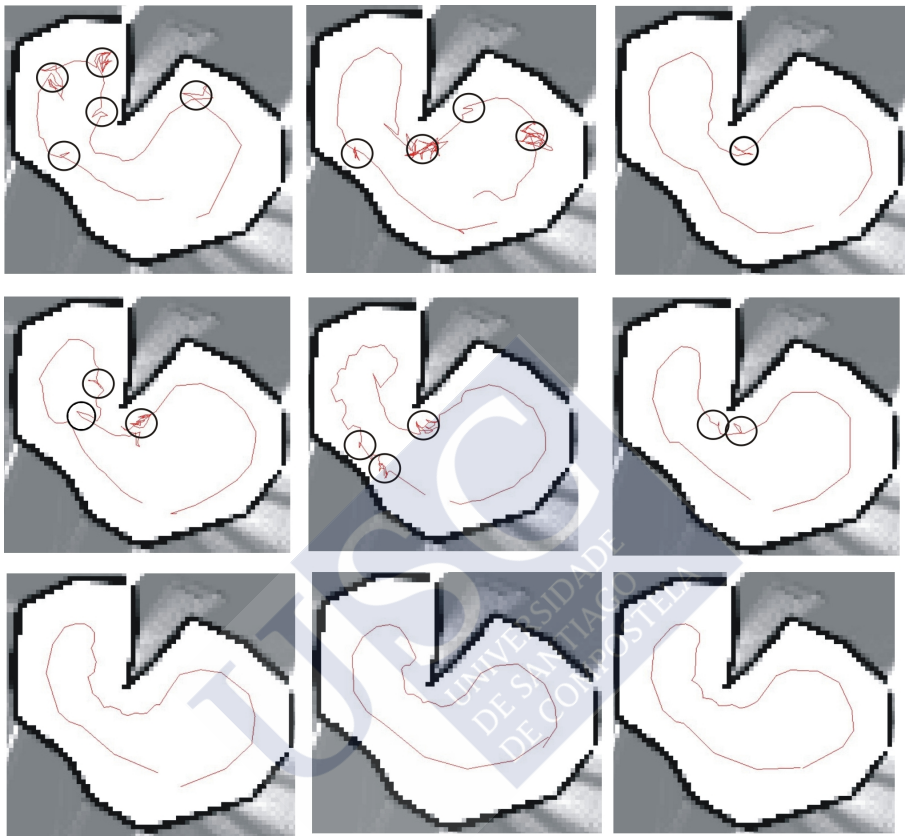


Figura 5.9: Primeras vueltas del primer experimento en el que el robot debe aprender el comportamiento seguir pared derecha en el entorno mostrado en la figura 5.8. La línea continua muestra la trayectoria del robot. Los círculos indican las principales zonas donde el robot recibió refuerzo negativo proveniente del observador humano. Tras 6 vueltas el robot es capaz de ejecutar la tarea sin errores.

5.4.5. Aprendizaje con usuarios no expertos

Durante el desarrollo de la investigación presentada en esta sección se han realizado pruebas de aprendizaje en el robot real con personas ajenas a la robótica. Estas pruebas eran parte de diferentes talleres de divulgación científica donde el perfil de los usuarios era el de una persona de entre 18 y 50 años con estudios universitarios.

La tarea del usuario consistía en conseguir que el robot aprendiese un comportamiento de seguir pared partiendo de un robot con un sistema de aprendizaje sin ningún conocimiento

A) B)

Figura 5.10: Estas figuras muestran el vestíbulo del Departamento de Electrónica e Computación donde el robot empezará a aprender en el segundo de los experimentos. En esta primera parte el robot dará varias vueltas a este entorno antes de que se le permita moverse a través de los pasillos del edificio. Para limitar el movimiento del robot se situaron planchas que bloqueaban los pasillos.

sobre la tarea y el entorno. Para ello se montó un pequeño entorno de aprendizaje donde el robot debía aprender a seguir la barrera que delimita el entorno. A los usuarios se les dieron unas sencillas instrucciones y consejos:

– Antes de empezar el proceso de aprendizaje es necesario que el usuario tenga una idea clara de qué comportamiento quiere conseguir. Al robot se le deben dar refuerzos lo más consistentes posible.

– Cuando el usuario observe que el robot está haciendo algo inadecuado debe proporcionar un refuerzo negativo apretando el gatillo del *joystick*.

– Tras el refuerzo el usuario debe controlar el robot con el *joystick* hasta recolocarlo en una posición y después pulsar un segundo botón para que el sistema de control del robot pueda continuar con el aprendizaje.

En general los usuarios aprecian el progreso del robot durante el aprendizaje, y les agrada ver cómo va mejorando el comportamiento. Aquellos usuarios que logran entender el mecanismo de aprendizaje por refuerzo, las capacidades sensoriales del robot y logran hacerse un modelo mental del sistema son capaces de proporcionar un refuerzo más adecuado y consiguen que el robot aprenda en menos tiempo.

Se encontraron algunos problemas y dificultades durante estas pruebas:

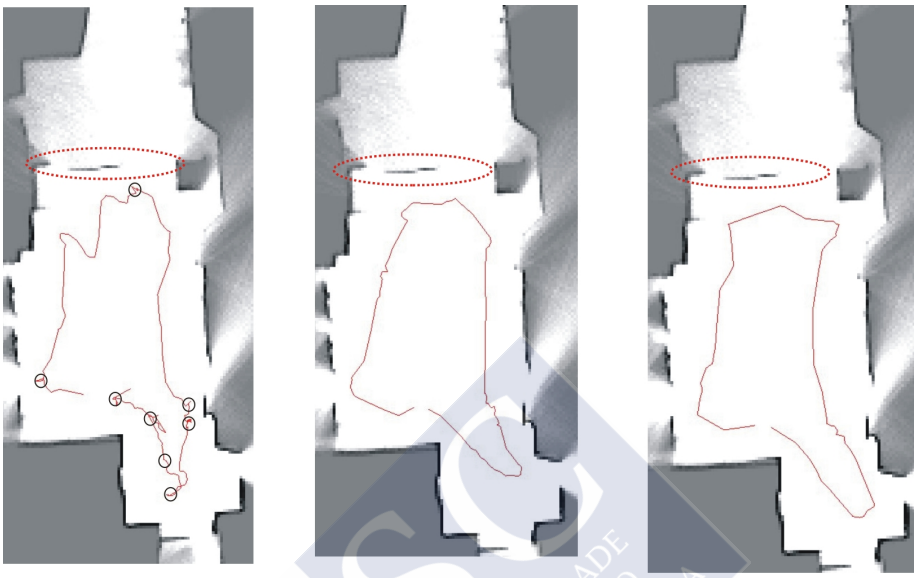


Figura 5.11: Primera parte del segundo experimento realizado para enseñar a un robot a moverse por el Departamento de Electrónica e Computación. Como se puede ver al observar la trayectoria del robot, únicamente en la primera vuelta el robot recibió refuerzos negativos (indicados con círculos).

– El refuerzo proporcionado al robot no es consistente durante la duración de todo el aprendizaje. El comportamiento buscado por el usuario cambia a medida que avanza el aprendizaje. En ocasiones es un cambio menor, como variar la distancia a la que se sigue la pared. En otras ocasiones el usuario busca simplemente el comportamiento seguir pared sin ninguna preferencia por el lado izquierdo o derecho, y al recolocar el robot cambia de sentido, pretendiendo que siga la pared del lado contrario.

– En ocasiones el robot tarda en resolver un problema y el usuario interpreta que el robot está haciendo siempre lo mismo sin obedecer a los refuerzos indicados. La solución que se le da al usuario es que recoloque al robot en una posición más alejada de la zona conflictiva con el fin de que el robot tenga más opciones a la hora de superarla.

– Cuando el robot comete un error, el usuario le indica al robot que lo está haciendo mal mediante la voz, pero no se da cuenta de pulsar el botón de refuerzo hasta pasados unos

Figura 5.12: Segunda parte del experimento mostrado en la figura 5.11. Una vez el robot completó tres vueltas en el recibidor del departamento se eliminaron los obstáculos que bloqueaban el acceso a los pasillos. Se puede observar el pequeño número de refuerzos recibidos – indicados con círculos – pese a que el robot se encontraba en un entorno completamente nuevo.

ciclos. Este retraso en la generación del refuerzo puede conllevar unas consecuencias muy negativas para el aprendizaje.

– En otras ocasiones el usuario no le da tiempo al robot para corregir su comportamiento y le indica un refuerzo negativo demasiado pronto, cuando el robot aún no estaba claramente en una situación de refuerzo.

– Algunos usuarios esperan demasiado del robot y pretenden que siga la pared a una distancia muy precisa, indicando refuerzo tan pronto el robot se aleja ligeramente de la distancia esperada.

Figura 5.13: Estas imágenes muestran el entorno del Departamento de Electrónica e Computación de la Universidade de Santiago de Compostela, donde el robot aprendió a moverse siguiendo la pared situada a su derecha. Este entorno contiene tanto pasillos (a) como espacios abiertos (b).

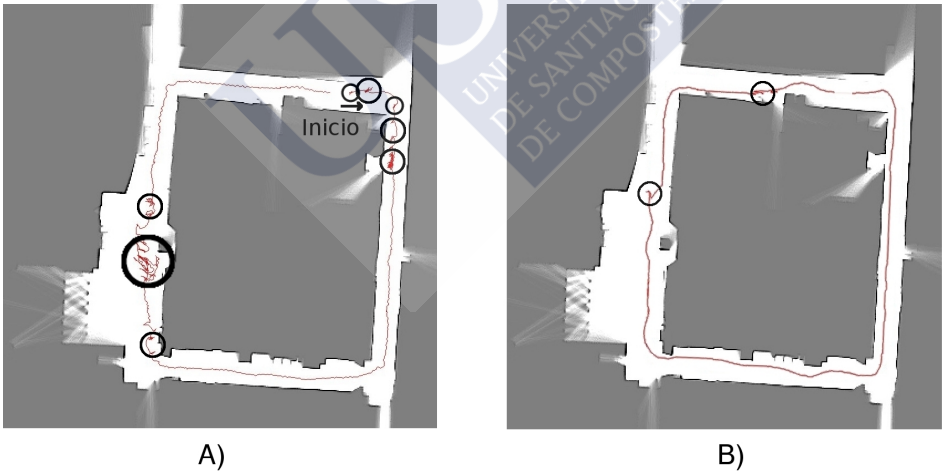
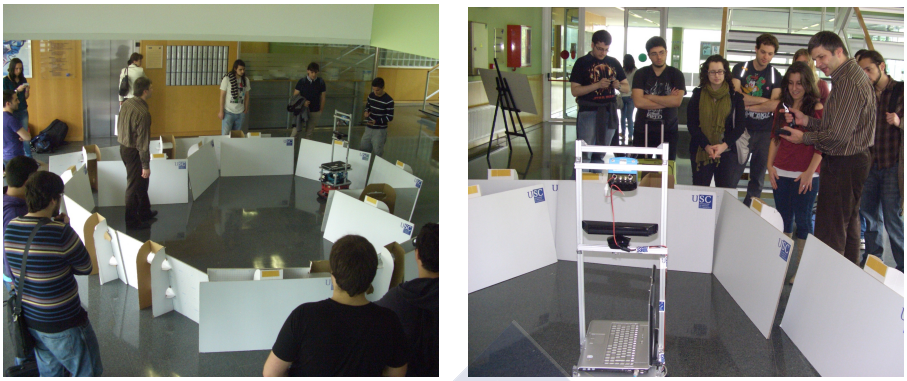


Figura 5.14: Trayectoria seguida por el robot en las dos primeras vueltas al entorno mostrado en la figura 5.13. Tras la primera vuelta el robot no recibe refuerzo negativo (indicados con círculos).

– Tras indicar un refuerzo, algunos usuarios colocan al robot en una posición más adelantada a aquella en la que recibió el refuerzo. Esto impide que el robot se enfrente al problema en el que erró, con lo que no aprenderá a solucionarlo.



(a) (b)

Figura 5.15: Experimento del aprendizaje con usuarios no expertos en robótica. a) Entorno de aprendizaje utilizado. b) Algunos de los usuarios recibiendo indicaciones sobre cómo dirigir el aprendizaje.

5.5. Discusión

El sistema de aprendizaje presentado en este capítulo nos pone un paso más cerca del aprendizaje continuo en robots operando en entornos reales. En este capítulo se han presentado dos propuestas para resolver el dilema *bias-varianza*. Ambas propuestas se basan en el mismo principio, la creación de un comité de expertos, los cuales serán capaces de resolver la tarea actuando de forma conjunta. Las dos propuestas se han demostrado válidas en los resultados experimentales y cada una tiene sus ventajas e inconvenientes.

Un aspecto común es que los miembros de cada comité deben cometer errores no correlacionados, por lo que se necesita que vean y actúen de manera diferente. Para ello cada uno de los miembros de los comités cuenta con su propia representación de estados, por lo que hay una red *Fuzzy ART* independiente para cada uno de los miembros. El uso de comités de aprendedores proporciona la ventaja de que permitiría trabajar con salidas continuas, en lugar del conjunto de acciones discreto que proporciona el algoritmo *I_Tbf* original

La primera propuesta consiste en un comité de evaluadores de acción, donde cada uno de los aprendedores cuenta con una división de todo el espacio de acciones en intervalos aleatorios y aprenderá cuáles son los intervalos válidos para cada estado. Esta propuesta incluye una nueva función de valoración, a la que llamamos función de utilidad U, con la que se consigue añadir estabilidad al sistema. Mediante U se puede discernir qué aprendedores proponen

buenas políticas y cuáles no lo hacen, impidiendo que se les tenga en cuenta en el proceso de votación. Los resultados en simulación muestran que esta estrategia de aprendizaje proporciona convergencias rápidas, y que es por lo tanto válida para su traslado al robot real. Las pruebas de aprendizaje con el robot real dieron resultados satisfactorios, ya que el robot adquiere una política de control correcta en unos pocos minutos. Un problema de esta propuesta es que el algoritmo se hace más complejo, consume más memoria y tiene un mayor número de parámetros. Por otro lado, al aprender las valoraciones de todos los intervalos del espacio de acciones no será necesario que el comité esté formado por un gran número de aprendedores.

La segunda propuesta, el aprendizaje mediante comité de evaluadores de política, es una alternativa a la anterior, donde se introducen características para permitir el aprendizaje continuo y se simplifica el sistema para aligerar los requerimientos de memoria. En este sistema cada miembro del comité se limita a evaluar una única política de control (inicialmente generada de forma aleatoria). La principal mejora de este sistema es el permitir el aprendizaje continuo, por lo que no es necesario que un episodio finalice para actualizar las valoraciones.

Otra característica importante de la segunda propuesta es la existencia de un comité de políticas de observación, las cuales irán recopilando información sobre qué acciones dan buenos resultados con el fin de focalizar la atención y estabilizar el aprendizaje. Cada cierto tiempo se transfieren políticas del comité de observación al comité de decisión, añadiendo así conocimiento sobre las acciones con mejor expectativa de tiempo a fallo. El comité de observación proporciona la estabilidad necesaria para que el algoritmo converja aun cuando el refuerzo proporcionado sea altamente no determinista.

Todas estas mejoras permiten que sea un usuario humano el que proporcione el refuerzo al robot a través de un dispositivo inalámbrico. Un posible problema del uso de refuerzo proporcionado por un humano es la falta de coherencia, por lo que el refuerzo recibido por el algoritmo no será determinista ya que ante las mismas situaciones el refuerzo puede ser diferente. Nuestro sistema se comporta bien siempre que el usuario se mantenga dentro de unos márgenes tolerables. No hemos realizado un estudio exhaustivo sobre la influencia del refuerzo humano en el algoritmo de aprendizaje, si bien en las pruebas llevadas a cabo con usuarios no entrenados se ha visto que la mayoría son capaces de enseñar al robot tras recibir unas sencillas indicaciones sobre el funcionamiento del mismo.

Una desventaja del aprendizaje mediante un comité de evaluadores de políticas respecto al comité de evaluadores de acción es que, si una política de control no es válida el sistema aprende a no tenerla en cuenta, pero dicha política no evolucionará y permanecerá de manera

residual en el sistema. Las dos propuestas tienen sus ventajas e inconvenientes, por lo que se plantea en un futuro próximo unirlas en un único algoritmo que combine las ventajas de cada una de ellas.

Capítulo 6

Selección de los sensores más significativos para los procesos de aprendizaje

Como se ha comentado en capítulos anteriores, el uso de robots en entornos cotidianos es aún muy limitado. Gran parte de la culpa de este hecho es la incertidumbre asociada al comportamiento de cualquier robot autónomo, que dependerá de las propiedades del propio robot, el *software* que lo controla y el entorno en el que se desarrolla la actividad [85]. Para los robots domésticos y de servicio el problema se agrava debido a lo imprevisible que son tanto el entorno como el comportamiento del usuario. Entre otras cosas, esto hace que sea muy difícil conocer de antemano qué sensores serán los más adecuados para desarrollar la tarea.

Con el aumento en el número y resolución de los sensores de los que disponen los robots actuales, hay un aumento asociado en la complejidad y el volumen de la información sensorial. Sin embargo, gran parte de la información sensorial no será relevante para la tarea que el robot está aprendiendo. Debido a esto el robot puede sufrir el problema conocido como la *maldición de la dimensionalidad* es decir, el incremento en la complejidad en los algoritmos aumenta exponencialmente con el número y complejidad de los sensores (dimensiones del espacio sensorial) [13]. Esta información innecesaria para la tarea confunde a las técnicas de *clustering* necesarias para identificar eventos en la información sensorial [88]. Tener más

información no significa que tengamos mejor información, ya que la proporción señal-ruido disminuirá si la información añadida no es relevante para la tarea.

Se hace necesario pues el disponer de un sistema automático de detección de los sensores más significativos por dos motivos. El primero es que deseamos un sistema capaz de adaptar su comportamiento sin la intervención humana. Por lo tanto, el robot debe ser capaz de seleccionar los sensores más significativos por sí mismo dado que en un entorno doméstico no habrá ningún experto disponible para adaptar el conjunto de sensores a las circunstancias particulares. El segundo motivo es que, incluso para un experto, los sensores a escoger pueden no ser intuitivos y en ocasiones la relevancia contradice a la intuición. Por ejemplo, en [54] y [85] se analiza un comportamiento de cruzar puerta con NARMAX. Los resultados obtenidos demuestran que para esta tarea es suficiente con utilizar los sensores del lado derecho del robot, obviando por completo la información proporcionada por los sensores del lado izquierdo.

En este capítulo se presenta un método para reducir de manera automática la complejidad sensorial de cara al aprendizaje de tareas por parte del robot. Dado que el desarrollo de dicho sistema es reciente, no se ha realizado la integración de este sistema con el aprendizaje basado en comités, sin embargo, las pruebas se han orientado hacia una futura implementación conjunta de ambos sistemas.

6.1. Trabajo relacionado

La selección de características relevantes se puede describir de manera resumida como: dado un conjunto de S características, seleccionar un subconjunto S_R de características que proporcione la misma o similar información que el conjunto original [32]. Dentro del campo de selección de características se pueden dividir las estrategias como métodos de filtro o métodos de envoltura (*wrapping*) [77]. En nuestro caso serían de aplicación los métodos de filtro, como el método de selección de características basado en correlación [46] o el algoritmo RELIEF [63], ya que únicamente se basan en características de los datos como la entropía o la correlación. Por el contrario, los métodos de envoltura utilizan la precisión predictiva de un algoritmo de clasificación específico para determinar el poder discriminatorio de un subconjunto de características. Nuestro problema de aprendizaje por refuerzo es no supervisado, por lo tanto los métodos de filtro serán los que mejor se ajusten a lo que necesitamos. Por otra parte, no existen muchos trabajos sobre selección de sensores en robótica, los más relevantes se describen a continuación.

En [133] se presenta un estudio sobre la relación entre el comportamiento y la percepción de un robot. El trabajo está contextualizado en el estudio de la retroalimentación cerebral de la conducta. Proponen el uso de Control Distribuido Adaptable (*Adaptive Distributed Control*) con un conjunto de controladores prefijados, y muestran cómo se puede mejorar el comportamiento de un robot si se aprende qué entradas sensoriales son las más relevantes. El control distribuido adaptable está basado en el supuesto de que los comportamientos flexibles son el resultado de tres capas de control altamente cohesionadas: la capa reactiva, la capa adaptativa y la capa de control contextual.

Iravani [56] obtiene los sensores más relevantes en un robot a partir de la búsqueda de estructuras relacionales en los datos. Su método detecta las características más relevantes en una n-tupla de características binarias utilizando un método de clasificación supervisada basado en el Q-análisis [8]. El Q-análisis es un marco de trabajo utilizado para describir y analizar relaciones entre miembros de un conjunto, dichas relaciones permiten realizar una clasificación de los miembros en función de un número limitado de características. Dado que trabaja con características binarias, con el fin de tratar con sensores continuos deben dividir cada sensor en un conjunto de características binarias que representen posibles valores o rangos de valores del sensor. Su método necesita probar todas las posibles combinaciones de características para detectar aquella que proporciona una mejor clasificación, esto hace que a medida que se incremente el número de sensores esta propuesta deje de ser viable.

Otra opción para estimar la importancia de cada sensor consiste en el método de modelado no lineal conocido como NARMAX [85, 54]. NARMAX es una estrategia de regresión no lineal que crea un modelo del sistema en función de las entradas pasadas, la salida deseada y el error cometido por el modelo. El uso de NARMAX para modelar el comportamiento de un robot proporciona la relación entre entradas y salidas como un polinomio. A partir de este polinomio se puede obtener la importancia de cada sensor. El principal problema de NARMAX para su aplicación al aprendizaje continuo es que, para ser fiable, NARMAX necesita que el robot esté ejecutando el comportamiento de manera correcta para poder modelarlo.

Durante los últimos años las máquinas de soporte vectorial (SVM) han recibido especial atención debido principalmente a su buena generalización y la ausencia de mínimos locales [113]. Las SVM son una técnica de aprendizaje supervisado generalmente utilizada para problemas de clasificación y regresión. Una SVM construye un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta que divide el espacio en clases. En [21] se realiza un análisis de sensibilidad de una SVM basándose en la idea de que los pesos de

las conexiones de las entradas relevantes generalmente tienen grandes valores absolutos, y los pesos de las entradas poco relevantes son mucho más bajos. El análisis de relevancia basándose en el peso de las conexiones detecta y elimina conexiones irrelevantes y sus consiguientes entradas.

Los índices de Sobol [120] son una técnica desarrollada expresamente para realizar análisis de sensibilidad de un modelo del sistema con respecto a sus variables de entrada. Analizando la varianza en la salida del sistema con respecto a diferentes entradas, se crean unos índices que representan cuánto contribuye a la varianza total cada una de las entradas, estos son los llamados índices de primer orden. Índices de órdenes superiores son creados para analizar el efecto en la salida de diferentes combinaciones entre variables de entrada. El método es notable porque funciona correctamente sin necesidad de utilizar aproximaciones simplificadas, incluso con modelos con un gran número de variables. Los índices de Sobol presentan buenos resultados en aquellos casos donde no se puede asumir la linealidad del modelo.

Existe otro conjunto de trabajos que utilizan la información mutua para detectar las características más relevantes. La información mutua es una medida de la información que comparten dos variables. Por lo tanto, si la información mutua entre dos variables cualesquiera es alta, cuanto más sepamos de una variable más se reduce la incertidumbre sobre la otra variable. Por el contrario, si la información mutua entre dos variables es cero dichas variables son independientes.

En [50] se analiza el modelo obtenido con NARMAX de un robot real que sigue una ruta. El modelo NARMAX proporciona la velocidad angular del robot en función de cada una de las entradas sensoriales y se pretende identificar cuales de esas entradas son las más relevantes. Para identificar los sensores más relevantes se han utilizado tres métodos diferentes: índices de Sobol, cálculo de derivadas parciales e información mutua. Los tres métodos proporcionan resultados prácticamente idénticos.

En esta tesis se usará la información mutua como medida de relevancia. En esta memoria describimos dos alternativas, si bien la segunda incluye la primera. La primera alternativa encuentra los sensores relevantes en base a su información mutua con las acciones del robot. En la segunda alternativa se seguirá una estrategia similar a la presentada en [89], la cual presenta un criterio de selección basado en la información mutua capaz de seleccionar los sensores relevantes a la vez que reduce la redundancia entre sensores. Al igual que en la primera propuesta, la relevancia se calcula como la información mutua entre los sensores y las acciones, y la redundancia es la información mutua entre sensores. Más adelante se

presentarán en detalle todos los aspectos relevantes de estas propuestas y nuestra adaptación para su uso en un robot autónomo.

6.2. Información mutua como medida de relevancia

Las medidas de información mutua y entropía pueden usarse para determinar el subconjunto de sensores que son más significativos para la tarea que el robot está realizando. La entropía es una medida que refleja el grado de aleatoriedad de una variable y la información mutua nos indica la reducción de incertidumbre (entropía) sobre una variable que se consigue gracias al conocimiento de otra variable. Al igual que se hace en los árboles de decisión siguiendo la navaja de Occam [112], aquí se reducirán las dimensiones de entrada al sistema en función de la entropía y su información mutua.

Hemos decidido utilizar la información mutua como medida de relevancia dado que es un método robusto, no es excesivamente costoso, necesita pocos parámetros y es capaz de identificar los sensores relevantes para una tarea aun cuando su ejecución no es del todo correcta, algo muy útil durante el aprendizaje de comportamientos. Queremos calcular la información mutua entre sensores y acciones durante el proceso de aprendizaje. Para ello se usarán los datos obtenidos mientras el robot ejecuta la tarea de forma más o menos correcta, por lo que las acciones y las percepciones se van almacenando en aquellas situaciones en ausencia de refuerzo negativo. Por lo tanto se dispone de información acerca de lo que el robot está viendo a través de sus sensores y qué acciones toma de acuerdo con dichas percepciones. Con estos datos se puede calcular la información mutua entre los sensores y las acciones. Cuanta más entropía de la acción se reduzca debido a la información de un sensor en particular, más relevante será dicho sensor para la tarea que está siendo aprendida.

La información mutua proporciona un método para escoger aquellos sensores más relevantes y por lo tanto reducir la complejidad sensorial al descartar los sensores irrelevantes. Otra manera de reducir la complejidad sensorial es discretizando las medidas que un sensor proporciona a un número finito (y reducido) de valores. Se presentará una estrategia basada en información mutua y estrategias evolutivas capaz de detectar la mejor discretización de los valores que proporcionan los sensores. A continuación se explican sendos métodos para ambas alternativas.

6.2.1. Obteniendo los sensores más relevantes

En este trabajo se consideran relevantes aquellos sensores (o dimensiones del espacio de entradas sensoriales) que comparten más información con las acciones. Para encontrar aquellos sensores que tienen una mayor influencia en la selección de las acciones calcularemos la relación entre las entradas sensoriales y la acción ejecutada por el robot utilizando las lecturas obtenidas durante el proceso de aprendizaje y en ausencia de refuerzo negativo. Estimamos la sensibilidad de las acciones del robot, A, respecto a cada sensor, $S_i \forall i \in S$, utilizando la información mutua I_m [1]:

$$I_m(S_i,A) = \sum_{a_k}\sum_{s_j} P_{S_i,A}(s_{ij},a_k) log_2 \left[\frac{P_{S_i,A}(s_{ij},a_k)}{P_A(a_k)\cdot P_{S_i}(s_{ij})} \right], \tag{6.1}$$

donde $P_{S_i,A}(s_{ij},a_k)$ es la distribución de probabilidad conjunta con la que el robot ejecuta la acción a_k cuando el sensor S_i toma el valor s_{ij}. $P_{S_i}(s_{ij})$ y $P_A(a_k)$ son las distribuciones de probabilidad individuales para S_i y A resultando en valores s_{ij} y a_k, respectivamente. Los sumatorios de la ecuación 6.1 se deben a que trabajamos con conjuntos discretos de estados y acciones. En caso contrario sería necesario sustituirlos por integrales.

La información mutua mide la información que comparten S_i y A, es decir, cuánto podemos decir sobre una de las variables (en nuestro caso la acción del robot o el valor del sensor) cuando conocemos el valor de la otra. Si entre S_i y A no hay ninguna relación, eso quiere decir que son independientes, $P_{S_i,A}(s,a) = P_{S_i}(s)\cdot P_A(a)$ y, por lo tanto, la información mutua es cero. En el otro extremo, si las dos variables son idénticas, $P_{S_i}(s) = P_A(a)$, toda la información transmitida por una de ellas es compartida por la otra, en este caso el valor de una de las variables puede ser usado para determinar el valor de la otra. En este caso $I_m(S_i,A) = H(S_i) = H(A)$, donde $H(x)$ es la entropía de la variable x. En el caso más general, podemos asumir que las variables compartirán cierta información, y en consecuencia $I_m(S_i,A) \leq min(H(S_i),H(A))$.

La información mutua de los sensores con las acciones nos proporciona una ordenación de sensores en base a su relevancia. Es necesario algún criterio para decidir el número de sensores que serán seleccionados como sensores relevantes. En nuestro caso se decidió que el subconjunto de sensores relevantes será aquel que proporcione un porcentaje significativo de información mutua con la acción ejecutada. Para obtener dicho conjunto se sigue el procedimiento que se explica a continuación.

El primer paso consiste en sumar todos los valores de información mutua:

$$Total\_I_m = \sum_{i=1}^{i=|S|} I_m(S_i, A).$$

A continuación se ordenan los sensores en función de su información mutua, de manera descendente empezando por aquel que proporciona mayor información. Finalmente se seleccionan uno a uno los sensores de la lista hasta que la suma acumulada de su información mutua alcanza el 50 % de $Total\_I_m$.

6.2.2. Discretización en intervalos de las entradas sensoriales

Dividir los valores de sensores en intervalos en lugar de trabajar con el rango completo de valores posibles – la resolución completa de cada sensor – puede ser beneficioso para muchos comportamientos en robótica. Trabajar con intervalos puede ayudar a focalizar la atención del robot en aquellos aspectos que son más relevantes para la tarea siendo ejecutada, y por lo tanto se evitan detalles que pueden confundir al robot o alterar su comportamiento. Ejemplos de dichos problemas ocurren durante un comportamiento cruzar puerta o seguir convoy. En el caso de la tarea cruzar puerta, todo lo que el robot necesita conocer es la posición de la puerta en cada instante para que pueda moverse hacia el centro de la puerta. Detalles como los objetos encontrados detrás del umbral de la puerta – detectados con los sensores delanteros – harán más complicado el aprendizaje y menos robusto el comportamiento final. Una discretización muy tosca de los sensores que utilice únicamente dos intervalos – "1" mostrando obstáculos cercanos como el marco de la puerta, y "0" representando obstáculos lejanos – será prácticamente suficiente para resolver la tarea. Algo similar ocurre con el seguimiento de un convoy. En este caso el robot debe seguir de cerca al robot que tiene delante. La detección de obstáculos cercanos y situados en la parte frontal del robot es mucho más importante que la detección de obstáculos lejanos o aquellos situados detrás del robot.

La discretización de los valores de cada sensor en intervalos (partición) debe realizarse de manera automática y no supervisada. Se deben evitar criterios heurísticos que son propensos a fallos y dependientes de las consideraciones del diseñador. La discretización debe depender únicamente del propio sensor y del comportamiento ejecutado por el robot. Para ello se usará una combinación de información mutua y estrategias evolutivas [37], en particular se buscará la partición de los valores de cada sensor en el conjunto de intervalos que maximizan la cantidad de información mutua entre dichos sensores y las acciones ejecutadas por el robot.

Cálculo de la información mutua usando intervalos

Si dividimos el rango de posibles valores del sensor $S_i, \forall i$ en un conjunto de P intervalos:

$$S_i = \{S_i(1) = [0, s_1), S_i(2) = [s_1, s_2), ..., S_i(P) = [s_{P-1}, s_P = maximo\_valor\_de\_S_i)\}, \quad (6.2)$$

podemos calcular la información mutua entre la acción A y S_i cuando dicha partición es utilizada:

$$I_m(S_i, A, P) = \sum_{a_i} \sum_{l} P_{S_i,A}(S_i(l), a_i) log_2 \left[\frac{P_{S_i,A}(S_i(l), a_i)}{P_A(a_i) \cdot P_{S_i}(S_i(l))} \right], \quad (6.3)$$

donde $P_{S_i,A}(S_i(l), a_i)$ es la probabilidad de que el valor de S_i esté en el l-ésimo intervalo y la acción A tome el valor a_i. $P_{S_i}(S_i(l))$ es la probabilidad de que el valor de S_i esté en el l-ésimo intervalo.

Encontrando la mejor partición

Dada una partición de S_i en P intervalos, podemos desplazar los límites de los intervalos para maximizar la información mutua entre S_i y A. Para ello hemos escogido estrategias evolutivas (ES), ya que son adecuadas y de uso común para problemas de optimización continua de parámetros [37].

Una estrategia evolutiva es una técnica de optimización basada en ideas de adaptación y evolución: dada una población de individuos dentro de un entorno con recursos limitados, la competición por dichos recursos provoca la selección natural, la supervivencia del más apto. Por lo tanto, las estrategias evolutivas necesitan una función de idoneidad que debe ser maximizada. De hecho, a partir de esta función de idoneidad algunos de los mejores candidatos son seleccionados como semillas para la siguiente generación.

Hemos seleccionado uno de los algoritmos evolutivos más simples, (1+1)-ES [15] (algoritmo 6.1). Este algoritmo encontrará los $P-1$ límites superiores de los intervalos en los que se divide cada sensor S_i $(s_1, s_2, ..., s_{P-1})$, ecuación 6.2. Estas fronteras se encontrarán intentando maximizar la información mutua entre las acciones del robot y las lecturas sensoriales. Para ello, (1+1)-ES trabaja con una población de dos individuos: el actual (padre) y el resultado de su mutación (descendiente). Sólo si la idoneidad del descendiente es mayor o igual que la del padre el descendiente se convertirá en padre de la siguiente generación.

El algoritmo 6.1 muestra el proceso de selección de la mejor partición de intervalos. $I_m(S_i, A, x)$ y $I_m(S_i, A, y)$ son la información mutua entre A y S_i cuando los límites de los intervalos son aquellos en el vector x y el vector y respectivamente. Se utiliza una distribución

Algoritmo 6.1 Algoritmo de selección de la mejor partición de intervalos

$t = 0$
Crear un individuo con los límites de los intervalos, $x^t = (s_1, s_2, ..., s_{P-1}) \in \mathbb{R}^{P-1}$
repetir
  Escoger un nuevo vector de modificación $z^t \in \mathbb{R}^{P-1}$ de una distribución normal $N(0, \sigma)$
  Crear un nuevo candidato (descendiente) añadiendo el vector aleatorio z al individuo padre $y^t = x^t + z^t$
  si $I_m(S_i, A, x) > I_m(S_i, A, y)$ **entonces**
    $x^{t+1} = x^t$
  si no
    $x^{t+1} = y^t$
  fin si
hasta que Condición de finalización satisfecha

Gaussiana, o normal, de media 0 y desviación estándar σ para escoger vectores aleatorios con los que generar nuevos individuos a partir del padre x. Por lo tanto, el valor σ es un parámetro que determina el grado de variabilidad de los valores del padre. Por esta razón a σ se le suele nombrar como parámetro de control de la mutación. En general los valores de σ se alteran utilizando la regla de 1/5:

$$\sigma = \begin{cases} \sigma/c & \text{si } p_s > 1/5 \\ \sigma \cdot c & \text{si } p_s < 1/5 \\ \sigma & \text{si } p_s = 1/5 \end{cases}$$

donde p_s es la frecuencia con la que las mutaciones proporcionan descendientes mejores que el padre medida sobre un conjunto de pruebas. El parámetro c es un valor constante generalmente en el intervalo $[0,8, 1]$. Esta regla de 1/5 se aplica a intervalos periódicos, por ejemplo, cada k iteraciones. Cuanto mayor es el ratio de éxito mayor será el valor de σ en un intento de hacer una búsqueda más amplia de la solución en $\mathbb{R}^{P-1}$. Por el contrario, si el ratio de éxito es menor que 1/5 entonces σ se reduce para centrar la búsqueda en la solución actual.

Encontrando el mejor número de intervalos

De acuerdo a las secciones anteriores sabemos cómo encontrar la mejor partición de S_i en P intervalos, esta partición será aquella que maximiza la información mutua entre S_i y las acciones del robot A. Sin embargo, aún queda conocer el número de intervalos que se

Algoritmo 6.2 Algoritmo de selección del número de intervalos

$P = 1$
$max\_I_m = 0$
repetir
  Aplicar (1+1)-ES para obtener la mejor partición de S_i en P intervalos (algoritmo 6.1)
  si $max\_I_m < I_m(A, S_i, P)$ **entonces**
    $max\_I_m = I_m(A, S_i, P)$
    $mejor\_num\_intervalos = P$
  fin si
  $P = P + 1$
hasta que $P = maximo\_numero\_de\_particiones$
devolver $mejor\_num\_intervalos$ como mejor partición para S_i

deben usar. Debido a esto se aplica un proceso iterativo que, de nuevo, intenta maximizar la información mutua tal y como se ve en el algoritmo 6.2.

6.2.3. Resultados experimentales

Hemos aplicado la información mutua y estrategias evolutivas para encontrar los sensores más relevantes y su particionado para el comportamiento seguir pared derecha. Para el aprendizaje del comportamiento hemos aplicado el comité de aprendedores de acción, en concreto la versión descrita en [74]. Los procesos de aprendizaje se han realizado en simulación utilizando un Pioneer P3-DX equipado con un sensor láser SICK LMS-200 y un anillo de ultrasonidos, con un ciclo de control de 250 ms. El conjunto de datos utilizados para determinar los sensores más relevantes y su partición se almacenó durante las primeras fases del aprendizaje del comportamiento, cuando su movimiento era errático y cometía numerosos errores.

Sensores relevantes obtenidos con información mutua

Hemos calculado la información mutua entre cada sensor (entendido como componente o dimensión del espacio de entradas) y las acciones ejecutadas por el robot mientras aprendía la tarea. El resultado obtenido se puede ver en la figura 6.1. De acuerdo a la figura 6.1(b) las componentes más relevantes del láser son aquellas correspondientes a la parte frontal y el lado derecho del robot. Este resultado concuerda con lo que se espera de un comportamiento seguir pared a la derecha. En relación a los sensores de ultrasonidos, los más relevantes parecen ser

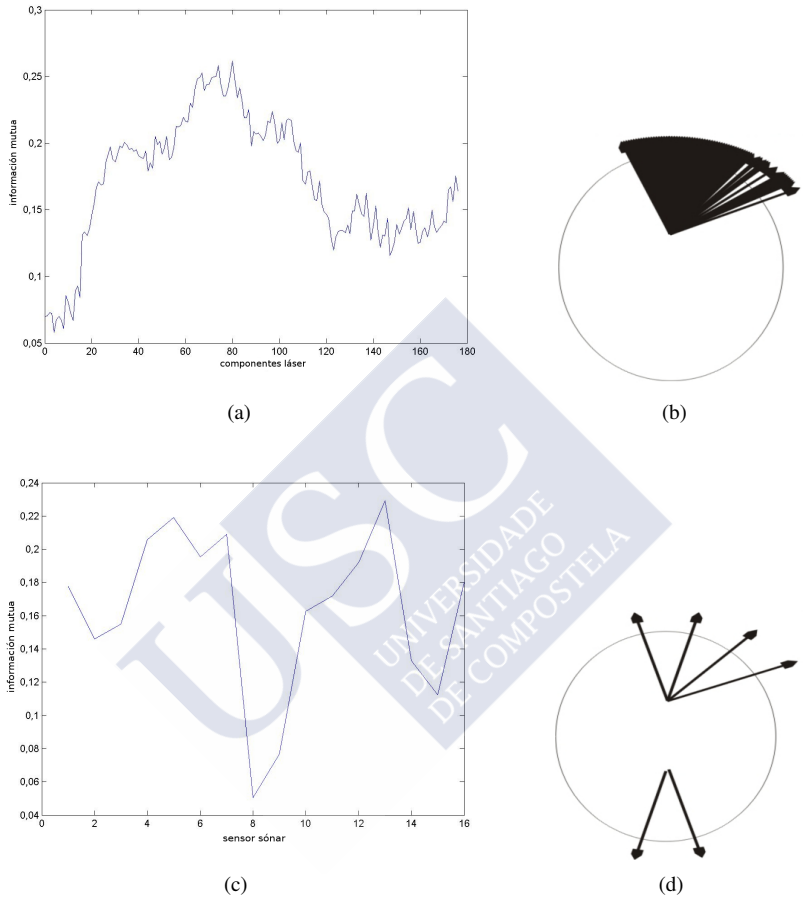


Figura 6.1: Información mutua entre cada sensor y las acciones del robot para la tarea seguir pared derecha, y los sensores relevantes obtenidos. Esta información mutua fue calculada utilizando los datos recopilados durante la fase inicial del proceso de aprendizaje. a) Información mutua de cada componente láser; b) componentes del láser más relevantes; c) información mutua de cada sensor de ultrasonidos; d) posición de los sensores de ultrasonidos más relevantes.

aquellos situados no sólo en la parte frontal, sino también los colocados en el lado derecho y en la parte trasera del robot.

Para determinar si esta información es válida se procedió a realizar una serie de aprendizajes utilizando únicamente el subconjunto de sensores relevantes obtenido mediante el criterio

Tabla 6.1: Tiempo requerido para aprender el comportamiento seguir pared cuando se aplican diferentes niveles de reducción de complejidad sensorial. Los valores que se muestran son los resultados obtenidos tras 15 experimentos de aprendizaje para cada una de las configuraciones que se indican. Los tiempos de aprendizaje están en *minutos:segundos*.

Características del aprendizaje	**Tmpo. aprendizaje**		**Parámetros *Fuzzy ART***	
	Media	σ	ρ	β
todos los sensores, sin intervalos	13:51	15:16	0,910	0,0015
sensores relevantes, sin intervalos	08:58	05:30	0,910	0,0010
todos los sensores, con intervalos	07:40	04:14	0,910	0,0015
sensores relevantes, con intervalos	07:05	03:31	0,926	0,0027

mostrado en la sección 6.2.1. De acuerdo a dicho criterio se ha seleccionado el número de sensores relevantes en función de su información mutua acumulada. Los sensores relevantes obtenidos para esta prueba son 76 sensores (componentes del láser y sensores de ultrasonidos) mostrados en las figuras 6.1(b) y 6.1(d). Dicho conjunto de sensores presenta mucha redundancia en la selección de las componentes del láser, algo esperable ya que entre dos componentes láser adyacentes sólo hay una diferencia de 1°, por lo que sus lecturas e información mutua con las acciones serán muy similares. Cuando únicamente se utilizó este conjunto de sensores relevantes para aprender la tarea se ha observado una importante reducción del tiempo de aprendizaje: de 13:51 minutos a 08:58 minutos (tabla 6.1).

Intervalos significativos obtenidos

A continuación se usó el mismo conjunto de datos para obtener la mejor partición para cada uno de los sensores en un conjunto finito de intervalos. La combinación de (1+1)-ES e información mutua, descrita anteriormente en la sección 6.2.2, nos permitió explorar la partición de los valores de cada sensor en un número de intervalos que puede oscilar entre 1 y un número máximo de 10 intervalos. En muchos casos la información mutua obtenida por la partición de los valores del sensor en intervalos se aproxima a una función logarítmica (figura 6.2). Esto es, la información mutua se incrementa muy rápidamente con los primeros intervalos, pero el ritmo de crecimiento disminuye al usar un mayor número de intervalos.

Para cada sensor tomamos aquella partición con mayor información mutua. Tal y como se observa en la figura 6.3(a) se seleccionaron 8, 9 o 10 intervalos para cada componente del

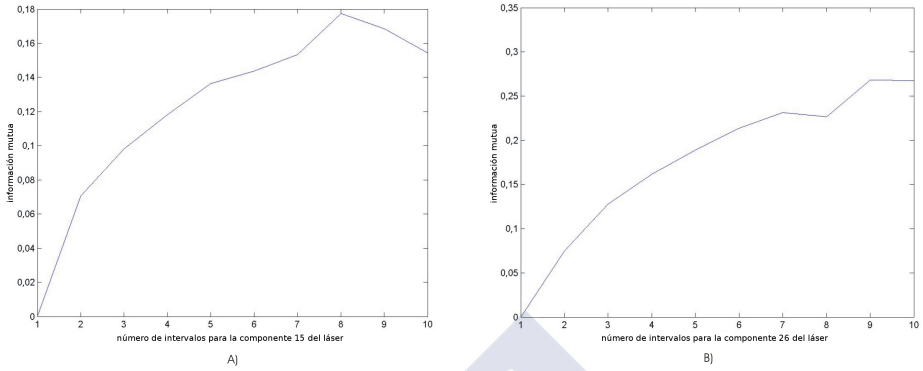
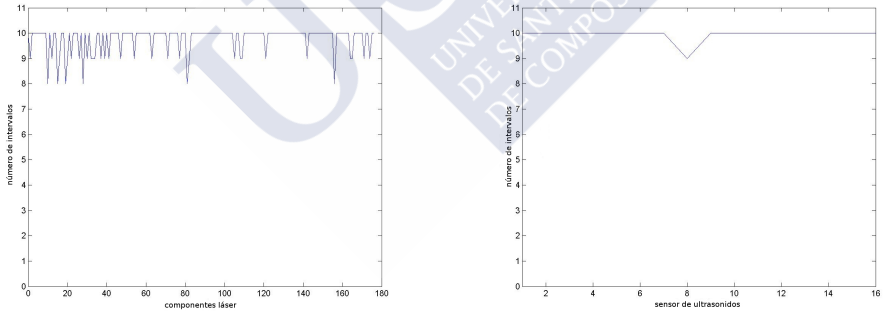


Figura 6.2: Información mutua en función del número de intervalos entre dos componentes láser diferentes y las acciones del robot en la tarea seguir pared derecha. Estas gráficas muestran el incremento de la información mutua a medida que aumenta el número de intervalos tomando como ejemplo las componentes 15 y 26 del láser.



(a) (b)

Figura 6.3: Número de intervalos en los que se discretiza cada sensor: a) componentes del láser; b) ultrasonidos.

láser, mientras que para el caso de los sensores de ultrasonidos todas las particiones resultaron de 10 intervalos a excepción de una de 9 intervalos (figura 6.3(b)).

Una vez más, se hicieron pruebas de aprendizaje para probar el rendimiento utilizando particiones. Hemos consideramos las cuatro posibilidades, según se emplearan todos los sensores o sólo los más relevantes y si se utilizaron el espacio completo de cada sensor o únicamente

las particiones generadas (intervalos). En el caso de las particiones, la red *Fuzzy ART* recibía como entrada el centroide del intervalo al que pertenecía la lectura sensorial. Se consiguieron importantes reducciones del tiempo de aprendizaje empleando las particiones (tabla 6.1). El uso de intervalos reduce de manera significativa el tiempo requerido para aprender la tarea, y esta reducción es incluso mayor cuando únicamente los sensores relevantes son utilizados (tabla 6.1).

6.3. Selección de los sensores más relevantes y menos redundantes

Como se ha visto en la sección anterior, se puede usar la información mutua para identificar los sensores más relevantes para una tarea. Sin embargo, la información mutua tiende a capturar las propiedades de las situaciones más frecuentes, lo que puede causar ciertos problemas ante situaciones muy relevantes pero poco frecuentes. En este caso, los sensores relevantes para esas situaciones poco frecuentes a menudo pasan desapercibidos cuando se calcula la información mutua con los datos capturados durante el aprendizaje de la tarea.

Para solucionar este problema, proponemos utilizar las categorías (estados) generadas por las redes *Fuzzy ART* en lugar de los datos sensoriales. Como ya se explicó en el capítulo 4, las redes *Fuzzy ART* son perfectas para distinguir entre situaciones diferentes, independientemente de su frecuencia de aparición. Esto se debe a que lo que determina las categorías es el parámetro de vigilancia, no el número de patrones identificados por la misma categoría. Situaciones poco frecuentes pero claramente distinguibles se verán representadas por alguna categoría de la red *Fuzzy ART*. Cada una de estas categorías viene caracterizada por un vector de referencia (sección 4.1.1). Por esta razón en esta sección se estudia el cálculo de la información mutua entre los vectores de referencia y la acción asociada a cada uno de ellos, en lugar de sobre cada una de las percepciones y acciones observadas y ejecutadas en cada ciclo de control del robot.

Otro problema de la ordenación de sensores en función de su información mutua con las acciones es que el conjunto resultante parece ser muy redundante (tal y como se puede apreciar en la figura 6.1(b)). Por lo tanto, se hace necesario diseñar un método capaz de encontrar aquellos sensores más relevantes para la tarea mientras se evita una redundancia excesiva entre ellos.

Para ello proponemos generalizar para el aprendizaje en robots el criterio de mínima redundancia máxima relevancia (*minimal-Redundancy-Maximal-Relevance criterion* (mRMR)) presentado por Peng *et al.* en [89]. El criterio de Peng nos parece adecuado pero debemos adaptarlo para el uso en aprendizaje no supervisado en robots, a esta adaptación la llamaremos mRMRδ.

La selección de características significativas siguiendo este método consta de dos etapas. Primero una etapa de ordenación de todas las características, y después una etapa para seleccionar el subconjunto de características más importantes para la tarea.

6.3.1. Ordenando los sensores por mayor relevancia y menor redundancia

Nuestro algoritmo mRMRδ crea una lista de sensores significativos de manera incremental. Se van escogiendo aquellos sensores que proporcionan más información y tienen menor redundancia con el conjunto de sensores seleccionados hasta el momento, S_{sel}. Para ello se combinan dos criterios, uno de máxima relevancia y otro de mínima redundancia [89].

El criterio de máxima relevancia (*Max-Relevance*) selecciona el subconjunto S_{sel} con aquellos sensores $s_i \in S$ que proporcionan la mayor información mutua con las acciones (relevancia):

$$\max_{S_{sel}}[D(S_{sel},A)],\ D = \frac{1}{|S_{sel}|}\sum_{s_i \in S_{sel}} I_m(s_i,A). \tag{6.4}$$

Para evitar excesiva redundancia se añade un criterio de mínima redundancia (denominado *min-Redundancy*) para seleccionar aquellos sensores que aportan información diferenciada:

$$\min_{S_{sel}}[R(S_{sel})],\ R = \frac{1}{|S_{sel}|^2}\sum_{s_i,s_j \in S_{sel}} I_m(s_i,s_j). \tag{6.5}$$

Combinando las ecuaciones 6.4 y 6.5 obtenemos el criterio mRMR [89]. El operador $\Phi(D,R)$ combina D y R para optimizar ambos criterios simultáneamente.

$$\max_{S_{sel}}[\Phi(D,R)],\ \Phi = D - R. \tag{6.6}$$

Para dar mayor importancia a la relevancia que a la redundancia nuestro criterio mRMRδ añade un factor de ponderación $\delta \in [0,1]$ al operador Φ:

$$\max_{S_{sel}}[\Phi(D,R)],\ \Phi = \delta D - (1-\delta)R. \tag{6.7}$$

Se puede ver que el criterio de Peng es un caso específico de mRMRδ donde $\delta = 0,5$. En los resultados se justificará la necesidad de introducir el factor de ponderación δ.

A partir de las ecuaciones anteriores se puede implementar un proceso incremental para ordenar los sensores encontrando aquellos que maximizan $\Phi(.)$. Para ilustrar este proceso incremental supongamos que del conjunto de sensores original $S, |S| = M$, ya se ha seleccionado un subconjunto, $S_{sel}, |S_{sel}| = m-1$, de sensores. El objetivo ahora es seleccionar el sensor m-ésimo del conjunto $S \setminus S_{sel}$ que maximiza $\Phi(.)$. Para ello el sensor que se escoge es aquel que maximiza la siguiente ecuación:

$$\max_{s_j \in S \setminus S_{sel}} \left[\delta I_m(s_j, A) - (1-\delta) \frac{1}{m-1} \sum_{s_i \in S_{sel}} I_m(s_j, s_i) \right], \tag{6.8}$$

Un valor alto de δ previene la inserción de sensores irrelevantes, aunque introduce más redundancia en el conjunto de sensores seleccionados.

Este proceso incremental nos permite ordenar los sensores pero no decidir cuáles debemos escoger para ser utilizados durante el aprendizaje en el robot. Recordemos que este mismo problema ya se planteó anteriormente (sección 6.2.1). En aquella ocasión se decidió incluir un umbral sobre el porcentaje total de información mutua para decidir qué sensores son los seleccionados. En la siguiente sección veremos cuál es el esquema que ahora proponemos para abordar este problema.

6.3.2. Seleccionando el número de sensores significativos

En el trabajo de Peng [89] se ordenan las características empleando información mutua, pero resuelve el problema de determinar el número óptimo de características ($|S_R|$) empleando el error de clasificación. Como hemos dicho anteriormente en nuestro caso no podemos utilizar un algoritmo supervisado. Aplicar la estrategia de Peng en nuestro sistema implicaría probar los M posibles subconjuntos de sensores para ver cuál de ellos conduce a un aprendizaje más eficiente. Esta tarea es completamente inviable, por lo que se hace necesario encontrar un criterio no supervisado.

Nosotros basaremos la decisión del número de sensores típicamente significativos en el valor del operador Φ (ecuaciones 6.7 y 6.8). Se ha observado que Φ es una función que asciende bruscamente y a continuación se estabiliza, acercándose asintóticamente a un valor constante (figura 6.4). Para la selección del número de sensores significativos hemos decidido tomar un umbral del valor de Φ una vez la función está estabilizada. Esto nos permite obtener

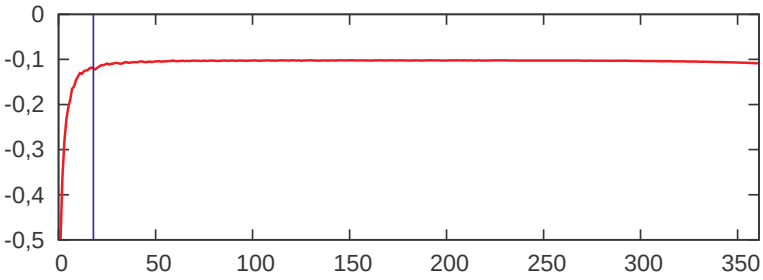


Figura 6.4: Ejemplo de la función Φ en función del número de sensores seleccionados. Se observa un rápido crecimiento y una posterior estabilización de la función. La línea vertical en 22 indica el tamaño del conjunto de sensores significativos seleccionado.

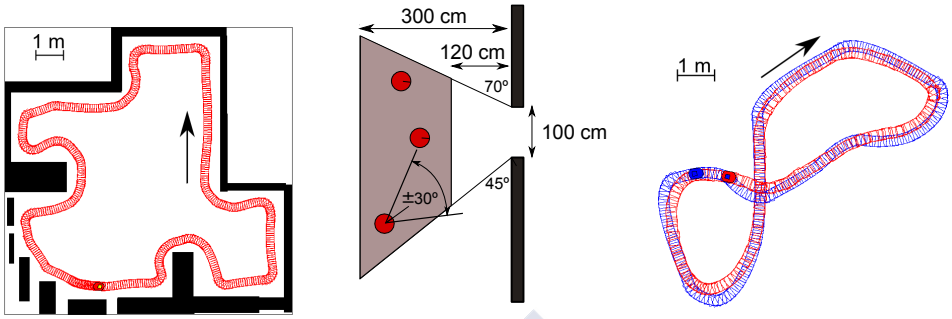
un conjunto de sensores con un bajo número de miembros y donde se alcanza un compromiso entre redundancia y relevancia. La selección del número de sensores aún no está automatizada, si bien sería perfectamente viable aplicar criterios basados en la pendiente de la curva para encontrar de manera autónoma un buen conjunto de sensores.

6.4. Resultados experimentales

En esta sección se presenta el estudio experimental de la influencia de los sensores significativos en el aprendizaje y la ejecución de los comportamientos en un robot. A través de los resultados pretendemos corroborar si esta reducción del número de sensores favorece los procesos de aprendizaje. Dado que únicamente queremos ver su influencia en el aprendizaje las pruebas presentadas en esta sección se realizaron únicamente en simulación. Se han realizado experimentos con tres comportamientos diferentes:

– **Seguir pared** [93]: el robot debe aprender a circular por el entorno manteniendo un cierto margen de distancia con la pared situada a su derecha. El entorno de aprendizaje es el mostrado en la figura 6.5(a).

– **Cruzar puerta** [85]: el robot debe aprender a cruzar un abertura en una pared situada frente a él. En cada episodio del aprendizaje el robot comienza en una posición y orientación aleatorias dentro de los márgenes mostrados en la figura 6.5(b).



(a) (b) (c)

Figura 6.5: Entornos simulados donde el robot aprenderá los diferentes comportamientos: a) comportamiento seguir pared, el robot aprenderá a seguir el contorno situado a su derecha; b) comportamiento cruzar puerta, las posiciones y orientaciones iniciales estarán dentro del área sombreada; c) comportamiento seguir líder, el robot (rojo) aprenderá a seguir al robot líder (azul) que ejecuta el camino mostrado.

– **Seguir líder** [107]: el robot debe aprender a seguir a cierta distancia a otro robot – robot líder. En nuestro caso, y para automatizar el proceso, el robot líder estará siguiendo una ruta prefijada ejecutando un algoritmo de navegación y evitación de obstáculos SND (*Smooth Nearness-Diagram navigation* [36]). En la figura 6.5(c) se puede ver la ruta ejecutada por el líder.

El robot usado para estos experimentos ha sido un Pioneer P3-DX equipado con dos sensores láser SICK LMS-200, uno orientado hacia la parte frontal y otro hacia la parte trasera del robot. Con esta configuración se tiene una cobertura completa de los 360° alrededor del robot – 362 componentes láser en total. No queremos hacer ninguna asunción sobre los sensores necesarios para ejecutar cada comportamiento, deseamos que sea el algoritmo el que encuentre un buen subconjunto de sensores significativos (S_R) para cada una de las tareas.

El período de control del robot ha sido 100 ms. Esto es, se adquieren las medidas procedentes de los sensores y se decide la acción a ejecutar cada 100 ms. Para la simulación hemos usado el entorno de simulación Player/Stage [43]. Para las tareas de seguir pared y cruzar puerta el robot se ha movido a una velocidad lineal constante de 0,15 m/s y el conjunto de acciones disponibles ha sido un grupo de 19 velocidades angulares en el rango $[-0,7, 0,7]$ rad/s. Por el contrario, para el comportamiento seguir líder el robot necesita disponer de diferentes velocidades lineales, por lo que en este caso en particular hemos reducido las velocidades

angulares a un conjunto de 9 velocidades en el rango $[-0,7, 0,7]$ rad/s, cada una de estas velocidades angulares se combina con una velocidad lineal de 0, 0,15, 0,30 y 0,45 m/s para formar el conjunto final de 36 acciones disponibles. El robot líder estuvo dirigido por un algoritmo de navegación SND a 0,3 m/s. Parte de estos resultados han sido publicados en [98].

6.4.1. Selección de los sensores para los procesos de aprendizaje

Se ha aplicado en todos los casos el método basado en información mutua (IM) presentado en la sección 6.2 y el mRMRδ con el fin de poder comparar los conjuntos de sensores obtenidos. En el caso específico del comportamiento seguir pared se probaron diferentes valores de δ, incluyendo $\delta = 0,5$ lo que equivale a aplicar el método original mRMR. En todos estos métodos el cálculo de la información mutua se realizó sobre los vectores de referencia (o vectores prototipo, ver sección 4.1.1) de las redes *Fuzzy ART* obtenidas durante una serie de aprendizajes, ya que se ha comprobado sin ambigüedad que es beneficioso frente al cálculo de la información mutua directamente sobre las entradas sensoriales. De cara a la implementación en un robot real esto no supone un problema ya que al integrar la selección de sensores significativos con el algoritmo de aprendizaje basado en comités se dispondrá de un gran número de redes *Fuzzy ART* (recordemos que cada miembro del comité posee su propia red *Fuzzy ART* con sus propios parámetros, ver capítulo anterior). En concreto se han utilizado 30 redes *Fuzzy ART* por cada comportamiento, lo que proporciona un volumen suficiente de datos sobre los que obtener resultados más robustos y fiables. A continuación se muestran los resultados para los tres comportamientos seleccionados.

Seguir pared

En el comportamiento seguir pared se compararon los resultados obtenidos con los diferentes métodos de selección de sensores presentados en este capítulo: el cálculo de sensores mediante información mutua (sección 6.2.1), la propuesta original de Peng mRMR y por último nuestra modificación mRMRδ. A través del comportamiento seguir pared buscamos un valor de δ adecuado para obtener un buen conjunto de sensores que mejore el entrenamiento. Posteriormente dicho valor de δ se ha utilizado para obtener los sensores significativos del resto de comportamientos.

La figura 6.6(a) muestra la información mutua (IM) de cada componente del láser con la acción del robot. Las componentes se numeran empezando en 0 en el lado derecho del robot e incrementándose en sentido antihorario, siendo 90 la parte frontal del robot y 180 el lado iz-

quierdo con respecto a la dirección de avance del robot. Con el criterio de selección a partir de la información mutua con las acciones [74] todos los sensores relevantes están en el intervalo $[29,121]$ – 93 componentes láser, figura 6.6(c). Dado que este criterio únicamente se centra en la importancia de los sensores respecto a las acciones, consideraremos que los sensores seleccionados son los relevantes para la tarea, siendo todos los sensores no seleccionados irrelevantes. Sin embargo, el conjunto de sensores relevantes puede presentar un alto grado de redundancia, ya que se intuye que las componentes adyacentes de láser proporcionarán información bastante redundante.

Para evitar esta excesiva redundancia se ha aplicado el criterio mRMR. La gráfica resultante de Φ (ecuación 6.7) se muestra en la figura 6.6(b) y el número de sensores seleccionados es 20 (figura 6.6(d)). Con esta técnica se seleccionan algunos sensores irrelevantes para la tarea según su información mutua, tal y como puede verse en 6.6(d), ya que muchos de los sensores seleccionados están por debajo del umbral de relevancia (figura 6.6(a)). Se observa que el conjunto de sensores seleccionados resultante es un subconjunto de sensores equidistantes del conjunto original que abarca los 360° alrededor del robot. La razón de que esto ocurra es que al no existir ninguna ponderación a favor de la relevancia, en la ecuación 6.6 se seleccionan algunos sensores irrelevantes para la tarea pero con poca redundancia antes que sensores más relevantes pero con mayor redundancia con sensores previamente seleccionados.

Aplicando el criterio mRMRδ podemos deshacernos de la mayoría de los sensores irrelevantes que introduce el criterio mRMR. Las figuras 6.7(a) y 6.7(b) muestran la evolución de Φ respecto a los sensores seleccionados según se utilice un valor δ de 0,8 y 0,9 (ecuaciones 6.7 y 6.8). Comparando ambas gráficas se puede observar cómo se obtienen mayores valores de Φ cuando se le da poco peso a la redundancia. Por otra parte, al tener en cuenta nuevos y más redundantes sensores, Φ puede ver disminuido su valor (especialmente si $\delta = 0,9$). Esto significa que nuestra prioridad es encontrar los sensores relevantes antes que evitar toda redundancia. Una vez los sensores más relevantes son seleccionados, el resto de sensores son muy redundantes y poco relevantes, por lo que Φ ve reducido su valor. El número de sensores ($|S_R|$) para $\delta = 0,8$ y $\delta = 0,9$ ha sido 34 y 22 respectivamente (figuras 6.7(c) y 6.7(c)). Si comparamos las figuras 6.6(c) y 6.7(c) podemos ver que utilizando mRMRδ con $\delta = 0,8$ el conjunto de sensores elegido contiene algunos sensores irrelevantes (con baja información mutua con las acciones). Subiendo δ a 0,9 el conjunto resultante contiene en su gran mayoría sensores relevantes y elimina aquellos más redundantes entre sí. Para las pruebas que siguen

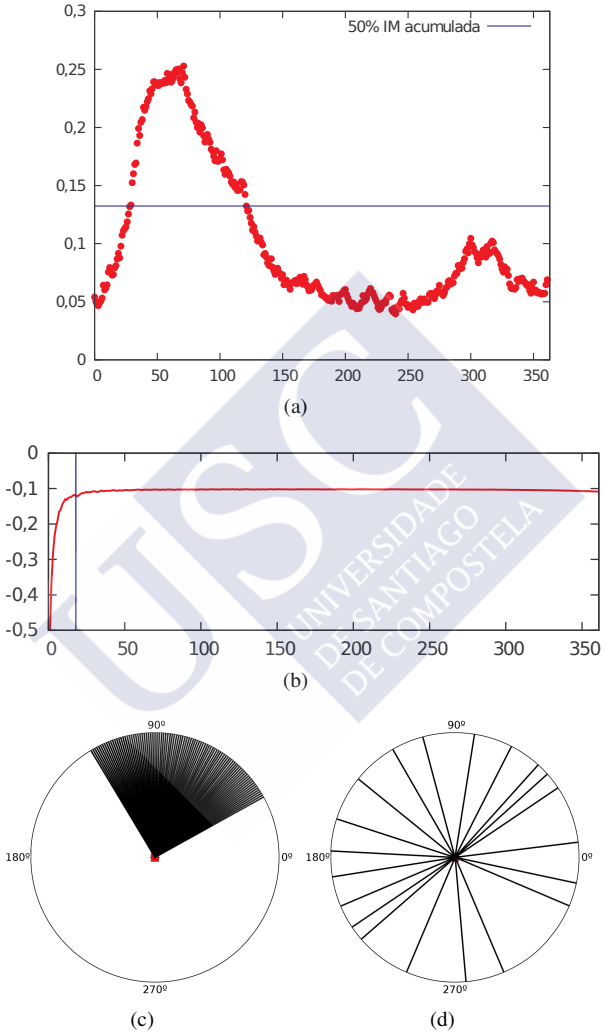


Figura 6.6: Selección de sensores relevantes para el comportamiento seguir pared utilizando dos sensores láser: a) IM entre las componentes láser (362) y las acciones; b) valor de Φ respecto al número de sensores seleccionados cuando se ordenan mediante el criterio mRMR. La línea vertical marca $|S_R|$; c) localización de los sensores más relevantes que acumulan el 50 % del total de información mutua, $|S_R| = 93$; d) localización de los sensores más significativos utilizando el criterio mRMR, $|S_R| = 20$.

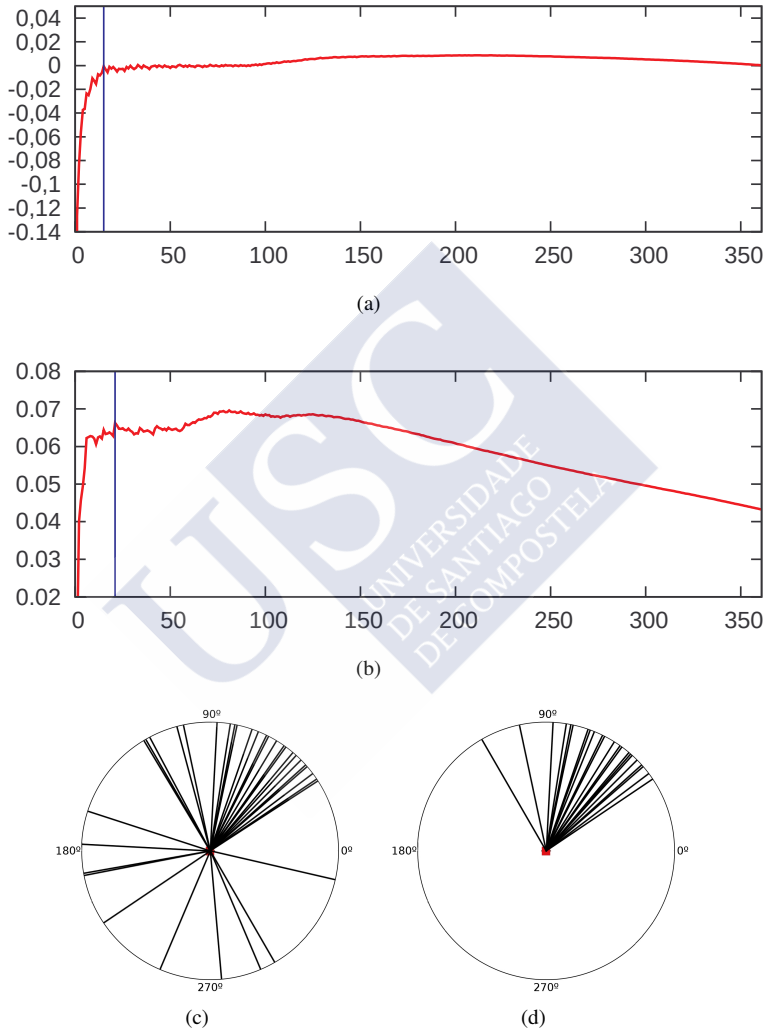


Figura 6.7: Selección de sensores significativos para el comportamiento seguir pared utilizando el criterio mRMRδ. a) Valor de Φ respecto al número de sensores seleccionados con $\delta = 0,8$, $|S_R| = 34$; b) valor de Φ respecto al número de sensores seleccionados con $\delta = 0,9$, $|S_R| = 22$; c) localización de los sensores más significativos con $\delta = 0,8$; d) localización de los sensores más significativos con $\delta = 0,9$.

con otros comportamientos se ha elegido $\delta = 0,9$ a la hora de calcular el conjunto de sensores significativos.

Cruzar puerta

En la figura 6.8(b) se pueden ver los sensores más relevantes – 139 componentes del láser – seleccionados mediante el criterio de información mutua con las acciones. Como era de esperar, contiene muchas componentes contiguas, y por tanto redundantes. Utilizando mRMRδ y $\delta = 0,9$ se han seleccionado únicamente 13 de las componentes del láser – $|S_R| = 13$ – (figura 6.8(c)). En este último caso, el conjunto seleccionado contiene más sensores (componentes del láser) del lado derecho del robot que del izquierdo. La causa este fenómeno puede deberse a que la zona en donde se recoloca el robot al principio de cada episodio de aprendizaje no es simétrica respecto a la puerta, ya que está ligeramente escorada a la derecha (figura 6.5(b)). En cualquier caso, este resultado no debe sorprendernos dado que ya en [54] se demuestra que únicamente los sensores de un lado son necesarios para aprender esta tarea, y nuestros resultados concuerdan con dicha afirmación.

También podemos observar cómo al calcular los sensores más relevantes todas las componentes traseras de la lectura del láser se descartan a excepción de un par de componentes laterales del láser. Intuimos que estas componentes tienen alta relevancia porque cuando el robot está cruzando la puerta son unas de las pocas capaces de detectar el marco de la puerta.

Seguir líder

Los sensores más relevantes obtenidos por el criterio de información mutua (IM) con las acciones se pueden ver en la figura 6.9(b). Se trata de un subconjunto de 58 entradas sensoriales escogidas entre las componentes 60 y 119 del láser. Los sensores seleccionados mediante mRMRδ y $\delta = 0,9$ se pueden ver en la figura 6.9(c). En este caso $|S_R| = 18$ componentes fueron seleccionadas, dos de las cuales no son relevantes según el criterio de información mutua.

Hemos observado que los valores procedentes de algunas componentes son siempre constantes, por lo que estas componentes son claramente irrelevantes (su información mutua con el conjunto de acciones es muy baja) por lo que son rápidamente descartadas. Esta es la razón por la que la figura 6.9(a) sólo contiene 169 componentes situadas en el rango $[4,\ 172]$. El hecho de que esto ocurra se debe a que el único obstáculo en el entorno era el robot líder y se situaba la mayor parte del tiempo frente al robot aprendedor.

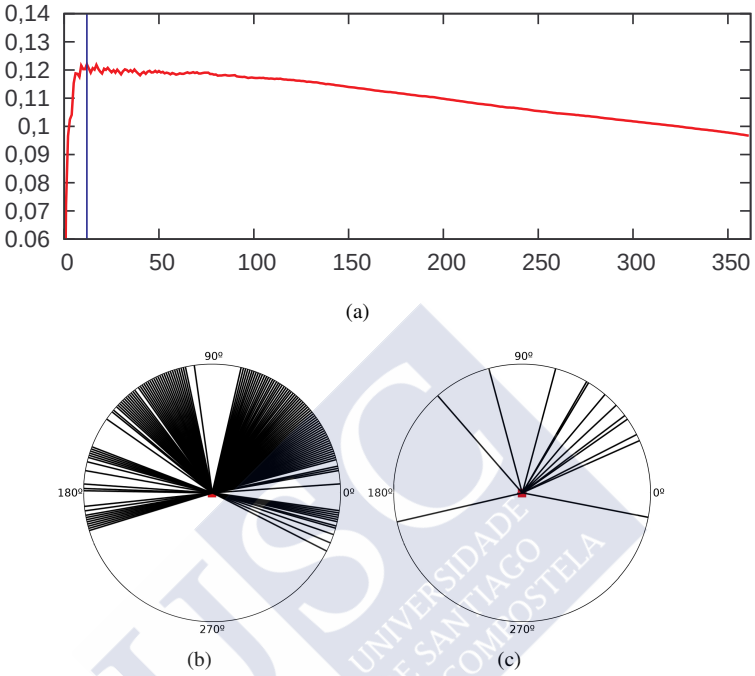


Figura 6.8: Selección de sensores significativos para el comportamiento cruzar puerta. a) Valor de Φ frente al número de sensores seleccionados utilizando mRMRδ con $\delta = 0,9$, $|S_R| = 13$; b) localización de los sensores más relevantes seleccionados utilizando el criterio de información mutua, $|S_R| = 139$; c) localización de los sensores seleccionados siguiendo el criterio mRMRδ con $\delta = 0,9$.

6.4.2. Aplicando los sensores más significativos al aprendizaje

Al igual que ya hicimos anteriormente en la sección 6.2.3, en esta sección se ha analizado si los conjuntos de sensores obtenidos mediante los criterios anteriormente presentados son beneficiosos para el aprendizaje del robot. Es importante destacar que en estas pruebas se ha utilizado el algoritmo *I_Tbf* y no la propuesta basada en comités. Se han realizado 30 experimentos con cada configuración de sensores utilizando siempre la misma configuración de los parámetros: coeficiente de aprendizaje de la red *Fuzzy ART* $\beta = 0,0$, y vigilancia $\rho = 0,9125$. Los resultados de los aprendizajes con todos los comportamientos se pueden ver en la tabla 6.2.

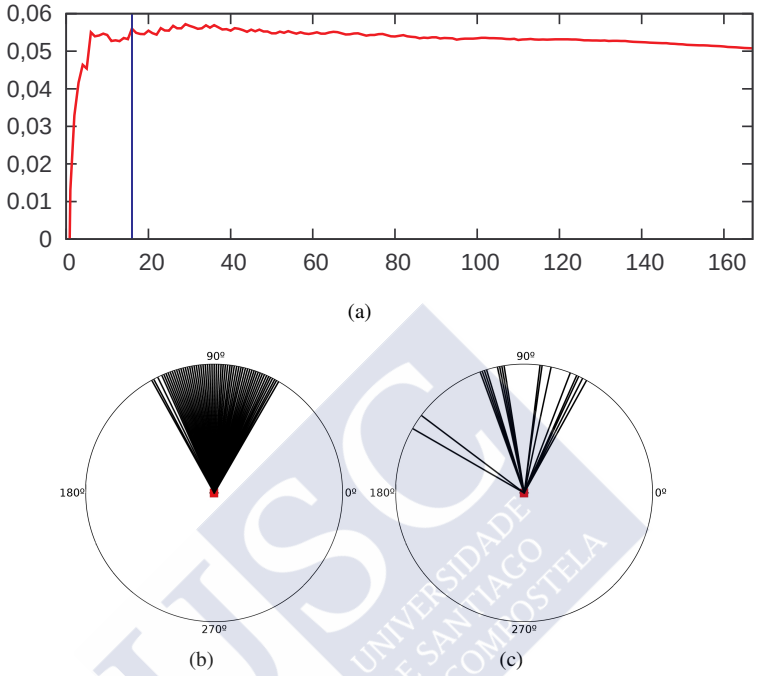


Figura 6.9: Selección de sensores significativos para el comportamiento seguir líder. a) Valor de Φ frente al número de sensores seleccionados utilizando mRMRδ con $\delta = 0,9$, $|S_R| = 18$. No se muestran los sensores que proporcionaron un valor constante durante toda la prueba; b) localización de los sensores más relevantes seleccionados utilizando el criterio de información mutua con las acciones, $|S_R| = 58$; c) localización de los sensores más significativos seleccionados utilizando mRMRδ con $\delta = 0,9$.

En todos los comportamientos el tiempo de aprendizaje se reduce cuando sólo se utilizan los sensores significativos proporcionados por el criterio mRMRδ (S_R). El caso más extremo es el aprendizaje del comportamiento seguir pared, donde el tiempo de aprendizaje se reduce de 122:16 minutos a sólo 08:52 minutos. En este comportamiento todas las componentes del sensor láser detectan obstáculos durante el aprendizaje, así que su variación realmente afecta a la clasificación de estados creada por la red *Fuzzy ART*. Si las componentes que no tienen importancia para la tarea, pero que están variando debido a la naturaleza del entorno, son eliminadas de la configuración sensorial que reciben las redes *Fuzzy ART* el tiempo de aprendizaje se reduce drásticamente. En los comportamientos cruzar puerta y seguir líder la mejora no es tan notable, dado que los entornos de aprendizaje son muy simples y muchas

Tabla 6.2: Resultados del aprendizaje de tres comportamientos con diferentes configuraciones sensoriales. Los tiempos de aprendizaje están expresados en *minutos:segundos*. $\delta = 0,9$.

Comportamiento	**Método de reducción**	$\lvert S_R \rvert$	**Tmpo. aprendizaje**		**Nº estados**	
			Media	σ	**Media**	σ
Seguir pared	Ninguno	362	122:16	54:39	98,8	38,5
	IM	93	21:17	14:11	22,1	9,7
	mRMR	20	51:42	28:20	37,8	15,4
	mRMRδ	22	08:52	06:30	12,7	4,3
Cruzar puerta	Ninguno	362	63:15	31:41	43,8	13,6
	mRMRδ	13	52:55	23:13	45,6	11,8
Seguir líder	Ninguno	362	56:32	44:46	21,1	8,5
	mRMRδ	18	37:26	36:54	33,4	29,4

componentes del vector láser no detectan nada durante gran parte del tiempo. Por otra parte, se ha podido observar que la estabilidad del aprendizaje se ve mejorada si se usan los sensores más significativos, tal y como demuestra la reducción en la desviación estándar en todos los tiempos de aprendizaje.

De los resultados obtenidos en el comportamiento seguir pared utilizando los grupos de sensores proporcionados por la información mutua o por mRMR, podemos concluir que es más importante mantener sólo los sensores relevantes para la tarea que descartar aquellos posibles sensores redundantes si ello conlleva la incorporación de sensores irrelevantes (lo que frecuentemente ocurre con el criterio de selección mRMR). Con la información mutua se mantienen los sensores más relevantes, pero a costa de una mayor redundancia. Sin embargo, el tiempo de aprendizaje ha sido mucho mejor – 21:17 minutos – que el tiempo obtenido con mRMR – 51:42 minutos.

También se observa que el uso de un conjunto de sensores reducido no implica una reducción en el número de estados creados por la red *Fuzzy ART*. Podemos concluir que la diferencia de tiempos de aprendizaje no se debe a la reducción del tamaño del espacio de estados, sino a su ”calidad”, es decir, si distingue o no todas las situaciones relevantes para la tarea. La eliminación en el espacio de entradas de aquellos sensores no relevantes para la tarea facilita esta clasificación.

6.4.3. Generalización de lo aprendido

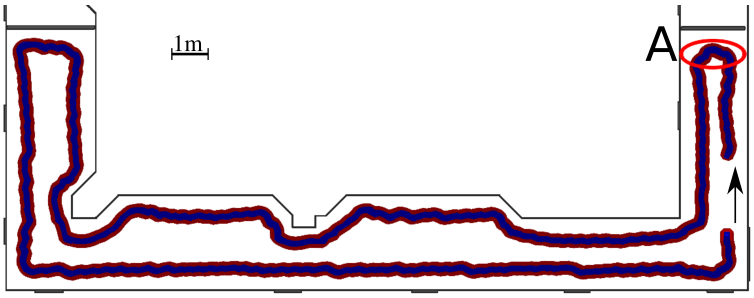
Para probar la robustez de los aprendizajes realizados, las políticas de control obtenidas se han probado en entornos diferentes a aquellos donde se llevó a cabo el aprendizaje. Para cada comportamiento se han probado las 30 políticas de control aprendidas con las 362 componentes de los dos sensores láser y las 30 políticas de control aprendidas empleando únicamente las componentes de los sensores láser seleccionadas mediante el criterio mRMRδ con $\delta = 0,9$. En los nuevos entornos el robot se ha enfrentado a nuevas situaciones que no encontró durante la fase previa de aprendizaje (entornos de la figura 6.5). Dado que en esta segunda tanda de experimentos no hay aprendizaje sino una validación de las políticas de control aprendidas, es necesario evitar que las redes *Fuzzy ART* creen nuevos estados. Para lograrlo se ha modificado ligeramente la regla de resonancia (sección 4.1.2), de modo que si una percepción (entrada de la red *Fuzzy ART*) no está en resonancia con ningún estado, dicha percepción se asignará al estado con la mayor similitud (ecuación 4.5).

Las pruebas de robustez de las políticas de control aprendidas se han realizado en dos fases. En primer lugar, se han validado todas las políticas en entornos simulados diferentes a los utilizados en el aprendizaje de cada comportamiento. En segundo lugar, se han probado en un robot y entorno reales aquellas políticas de control que han superado la primera fase. A continuación se muestran los resultados obtenidos para cada comportamiento.

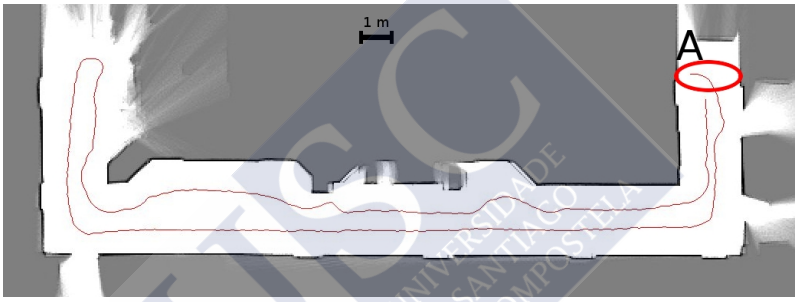
Seguir pared

La prueba de robustez en simulación se ha llevado a cabo en el entorno mostrado en la figura 6.10(a). Empleando todos los componentes de los dos sensores láser (362 medidas por adquisición), sólo 2 de las 30 políticas de control aprendidas – 7 % – han sido capaces de circular por el pasillo sin cometer ningún error. Los fallos cometidos por las diferentes políticas de control se han distribuido de modo casi uniforme por todo el nuevo entorno simulado.

Por otra parte, si durante la fase previa de aprendizaje sólo se han empleado como entradas el conjunto de 22 sensores (componentes del vector láser) más significativos (figura 6.7(d)), entonces el 12 de las 30 políticas de control aprendidas (40 %) se llegaron a ejecutar sin cometer ningún fallo. Hay que destacar que, del resto de políticas de control (18) que no han generalizado correctamente, 14 de ellas sólo han cometido un único fallo y siempre en el mismo punto del nuevo entorno simulado: el giro de 180° al final del pasillo de la derecha. Éste es un pasillo muy estrecho (sólo 2 m de ancho, zona A de la figura 6.10(a)), y se corresponde a una situación que nunca se ha encontrado el robot durante el aprendizaje (entorno simulado



(a)

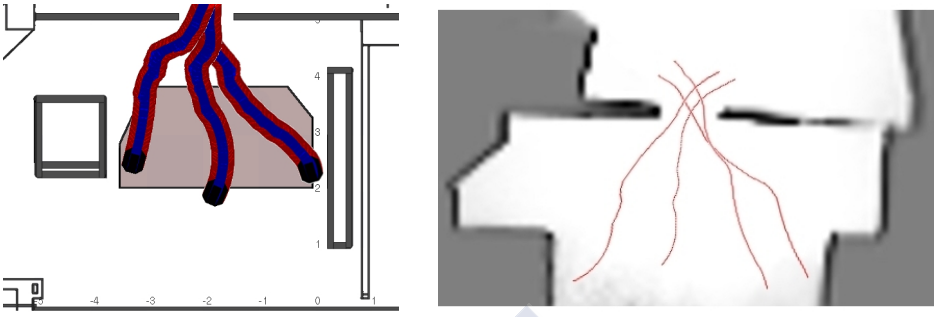


(b)

Figura 6.10: Entornos usados para probar la robustez del comportamiento seguir pared y ejemplos de las trayectorias seguidas por el robot en ambas pruebas. a) Simulado; b) Real.

de la figura 6.5(a)). Si en el entorno simulado de prueba se emplean pasillos de más de 2,5 m de ancho, entonces el porcentaje de políticas de control que funcionan correctamente sube a un muy notable 87 %.

Aquellas políticas de control que han superado satisfactoriamente la fase de simulación, se han probado en un entorno real, parte de un pasillo del Departamento de Electrónica e Computación de la Universidade de Santiago de Compostela (figura 6.10(b)). En esta última prueba las condiciones son más desafiantes, ya que el entorno contiene elementos que no aparecen en simulación, como muebles, puertas, gente o el ruido inherente a los sensores reales. De nuevo, casi todas las políticas de control (87 %) se han ejecutado correctamente, cometiendo error únicamente aquellas que no eran capaces de girar en pasillos de menos de 2,5 m de ancho. En la figura 6.10(b) se puede ver un ejemplo de este último tipo de políticas.



(a) (b)

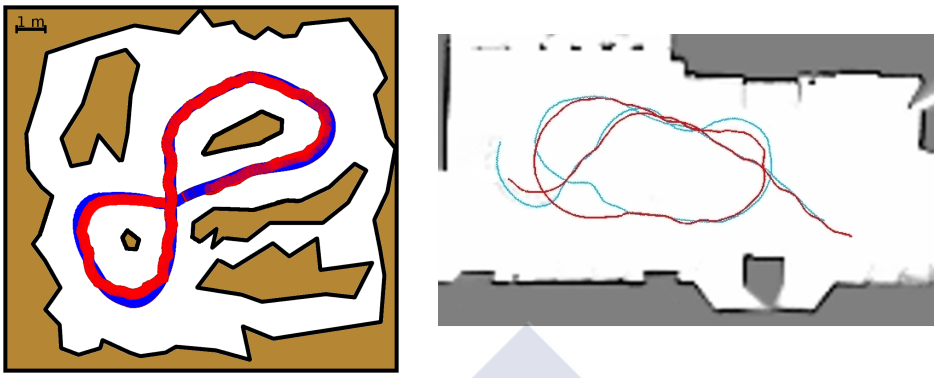
Figura 6.11: Entornos usados para las pruebas de robustez del comportamiento cruzar puerta y algunas trayectorias típicas. a) Simulado; b) Real.

Cruzar puerta

La primera fase de las pruebas de robustez ejecutadas para este comportamiento consistieron en hacer funcionar la política en el entorno simulado mostrado en la figura 6.11(a). Las posiciones iniciales se sitúan en la zona sombreada. Utilizando todos los sensores ninguna política de control ha sido capaz de ejecutar la tarea en el nuevo entorno. Forzando que la posición inicial del robot estuviese más centrada respecto a la puerta, el porcentaje de políticas de control que fueron capaces de cruzar la puerta subió al 30 % (9 de 30).

Por otra parte, 9 de las 30 políticas de control aprendidas empleando únicamente los sensores relevantes obtenidos mediante nuestro criterio mRMRδ (figura 6.8(c)) – 30 % – funcionaron a la perfección en el nuevo entorno. 14 de las 21 políticas que fallaron cometieron errores al partir de la zona situada más a la izquierda del área inicial. Esto se debe a la tendencia, tanto del entorno de aprendizaje como del conjunto de sensores relevantes, de dar más importancia a la derecha del robot. El 100 % de las políticas de control aprendidas con los sensores más significativos han funcionado con éxito cuando la posición inicial está más centrada respecto a la puerta.

A continuación se ejecutaron las pruebas de robustez en un entorno real. Algunas de las trayectorias ejecutadas por el robot real se pueden ver en la figura 6.11(b).



(a) (b)

Figura 6.12: Pruebas de robustez del comportamiento seguir líder y trayectorias ejecutadas por el líder (azul) y el seguidor (rojo). a) Nuevo entorno simulado; b) entorno real.

Seguir líder

Para probar la robustez de las políticas de control aprendidas para este comportamiento se añadieron al entorno de aprendizaje simulado ciertos obstáculos cercanos a la trayectoria del robot líder (figura 6.12(a)). De las 30 políticas aprendidas utilizando como entradas todas las componentes de los dos sensores láser, únicamente 2 (7 %), fueron capaces de continuar siguiendo al robot líder. Por el contrario, de las 30 políticas de control aprendidas empleando los 18 sensores (componentes del sensor láser) más significativos obtenidos mediante el criterio mRMRδ (figura 6.9(c)), 15 (50 %) se han ejecutado correctamente en el nuevo entorno simulado.

La figura 6.12(b) muestra las trayectorias seguidas por el líder y el seguidor durante las pruebas en el entorno real. Se puede apreciar de nuevo cómo la política aprendida en simulación empleando únicamente los sensores relevantes ha sido capaz de seguir al líder mientras éste ejecutaba una trayectoria nueva en un entorno diferente.

6.5. Discusión

En este capítulo se ha visto cómo un excesivo volumen de información sobre el entorno y/o número de sensores puede ser un obstáculo para el aprendizaje en robots. Demasiados sensores pueden introducir en el sistema un porcentaje sensiblemente mayor de información

irrelevante o redundante para la tarea, lo que disminuye la relación señal-ruido, y provoca peores aprendizajes y políticas de control. Determinar qué sensores son relevantes depende de tres factores: el robot, el entorno y la tarea. Al igual que ocurre con el aprendizaje por refuerzo, esos tres elementos pueden sufrir cambios o ser impredecibles, por lo que se hace necesario el uso de una técnica que no requiera la supervisión de un experto si se desea aplicar en robots de servicio, especialmente de servicio personal.

Para permitir que el robot adapte su percepción y encuentre el subconjunto de sensores más significativos se ha aplicado la información mutua para encontrar qué sensores están directamente relacionados con las acciones que el robot está ejecutando, y por lo tanto, con la tarea. La selección de sensores relevantes en base a su información mutua con las acciones ejecutadas permite reducir el tiempo de aprendizaje, pero si se utiliza únicamente este criterio es muy probable que el conjunto de sensores tenga una excesiva redundancia.

Gracias a la información mutua también se puede realizar una discretización de los valores proporcionados por los sensores, permitiendo así a los algoritmos de aprendizaje realizar una clasificación más sencilla de estados. Se ha propuesto la combinación de información mutua con estrategias evolutivas para encontrar de manera automática una discretización de cada sensor. Los resultados obtenidos muestran una reducción significativa del tiempo de aprendizaje. Encontrar el mejor intervalo para cada sensor tiene una alto coste computacional, lo que hace que dicha propuesta sea inviable en un robot aprendiendo en tiempo real.

Para reducir la redundancia se ha generalizado el algoritmo mRMR propuesto por Peng [89] para su uso en aprendizajes en robótica móvil. Nuestra modificación, llamada mRMRδ, introduce un parámetro de ponderación que permite ajustar la importancia relativa entre el criterio de relevancia y el criterio de redundancia.

Por último, se han realizado dos conjuntos de pruebas para evaluar los beneficios de emplear únicamente los sensores más significativos en el proceso de aprendizaje. Para ello se ha entrenado un robot en simulación para adquirir diferentes comportamientos desde cero (seguir pared, cruzar puerta y seguir líder). En primer lugar, se ha evaluado el impacto en el tiempo y en la estabilidad en el proceso de aprendizaje de cada política de control. Los resultados obtenidos nos permiten afirmar que la selección de los sensores obtenida con nuestro algoritmo mRMRδ produce grandes mejoras tanto en el tiempo como en la estabilidad de los aprendizajes en los tres comportamientos.

En segundo lugar, se ha comprobado el grado de robustez de las políticas de control aprendidas ante nuevos entornos, tanto simulados como reales. En este caso, se fija la política de

control para comparar todas las políticas en igualdad de condiciones. Los resultados han demostrado de nuevo que limitar la percepción del robot a los sensores más relevantes obtenidos con mRMRδ incrementa la robustez de la política de control aprendida en comparación con aquellos casos en donde se ha empleando toda la información sensorial disponible. Podemos concluir por tanto que la eliminación de los sensores irrelevantes para la tarea a aprender es muy beneficiosa y que el algoritmo mRMRδ que hemos propuesto es capaz de realizar dicha selección eficazmente y durante las primeras fases del aprendizaje, cuando aún la política de control no ha sido correctamente adquirida.

Capítulo 7

Conclusiones

El futuro de la robótica, y en particular de la robótica de servicio, pasa por disponer de robots adaptables al entorno en el que llevarán a cabo su trabajo. Esto se debe a que no se puede disponer de un controlador válido para todas las variedades de entornos que puede encontrar un robot aún cuando deba resolver la misma tarea en todos ellos. Puesto que no se puede disponer de un controlador perfecto, entendemos que la adaptación del mismo debe ser una característica imprescindible. Más aún, consideramos que dicha adaptación debe poder realizarse durante la ejecución de la tarea y a través de la interacción robot-entorno, pues es deseable que no sea necesaria la presencia de un experto en robótica. Por adaptación se puede entender tanto el aprendizaje de un nuevo controlador, como la modificación de uno ya existente. Independientemente del camino elegido, entendemos que la adaptación de un controlador en robótica debe cumplir una serie de características:

- *Rapidez*: el algoritmo debe ser lo suficientemente rápido como para permitir el aprendizaje en un robot real.
- *Interpretabilidad*: el usuario debe poder entender fácilmente en qué estado se encuentra el aprendizaje.
- *Facilidad de uso*: el robot va a ser utilizado por usuarios sin conocimientos de robótica, por lo tanto el aprendizaje debe ser lo más sencillo posible, sin requerir la intervención de un experto o demasiados ajustes de parámetros.
- *Definición del espacio sensorial*: el propio proceso de aprendizaje debe modificar su forma de percibir el entorno en función de la tarea.

– *Estabilidad*: el comportamiento del robot no puede mostrar grandes fluctuaciones en el caso de la aparición de nuevas situaciones. Lo ya aprendido hasta el momento debe permanecer estable y no verse afectado por errores cometidos en nuevos estados.

– *Continuidad*: no debería haber una diferenciación entre la fase de aprendizaje y la fase de puesta en uso. Esta característica facilitaría un posible aprendizaje continuo durante todo el ciclo de vida del robot.

El paradigma de partida utilizado para lograr esta adaptación ha sido el aprendizaje por refuerzo. El aprendizaje por refuerzo proporciona un mecanismo para el aprendizaje no supervisado de comportamientos, por lo que su aplicación es adecuada para un escenario donde no hay expertos que puedan realizar ajustes al sistema. Pese a todo, el aprendizaje por refuerzo tiene unas limitaciones que impiden o dificultan su aplicación directa en un robot real, y deben ser mejoradas para hacer el sistema viable si se pretende lograr una gran expansión de la robótica.

En este trabajo se ha propuesto un sistema de aprendizaje basado en refuerzo que tiene las características citadas anteriormente. En primer lugar, hemos propuesto un nuevo algoritmo de aprendizaje denominado *Increasing the time before failure* (*I_Tbf*), el cual basa su funcionamiento en una nueva función objetivo a través de la cual el robot es capaz de predecir el tiempo hasta que cometa un error cuando se sigue una determinada política de control. Este algoritmo proporciona diversas ventajas: *I_Tbf* tiene un rendimiento superior al de algoritmos clásicos de aprendizaje por refuerzo, por lo que proporciona aprendizajes rápidos. Por otra parte, el tiempo hasta el fallo proporciona una información mucho más comprensible que la información sobre el refuerzo esperado que proporcionan los algoritmos por refuerzo tradicionales, lo que permite evaluar y comprender mejor el proceso de aprendizaje. Por último, el algoritmo utiliza un único parámetro – el coeficiente de aprendizaje – el cual no necesita grandes ajustes, de hecho, en todas las pruebas presentadas en esta memoria se ha mantenido constante.

El segundo de los problemas abordados ha sido la necesidad que tienen los algoritmos por refuerzo de disponer de un espacio de estados adecuado para la tarea y el entorno. La solución clásica pasa por realizar un cuidadoso estudio del robot, la tarea y el entorno para crear un espacio de estados *ad hoc*. Este proceso es complejo y laborioso, por lo que resulta incompatible con procesos de adaptación que no requieran la intervención de un experto en robótica. La solución propuesta en este trabajo se ha centrado en utilizar una red de neuronas artificiales, *Fuzzy ART*, para crear el espacio de estados de manera dinámica. Combinando

el algoritmo *I_Tbf* con una red *Fuzzy ART* hemos conseguido un sistema de aprendizaje simultáneo de percepción y acción, capaz de aprender desde cero tanto el espacio de estados (entendidos como situaciones relevantes), como las acciones a ejecutar en cada uno de esos estados. Los tiempos de aprendizaje obtenidos mediante esta combinación son similares a los logrados utilizando una representación de estados *ad hoc*, por lo que si se tiene en cuenta el tiempo empleado para construir dicha representación estática, el uso de la red *Fuzzy ART* proporciona una mejora muy significativa al aprendizaje.

Un inconveniente de la red *Fuzzy ART* es que para obtener una correcta clasificación de las percepciones sensoriales en estados es necesario ajustar lo que se conoce como parámetro de vigilancia. Para resolver la dependencia con este parámetro hemos propuesto la modificación de la red *Fuzzy ART*. En particular, aprovechamos la información sobre el tiempo esperado hasta el fallo que proporciona el algoritmo *I_Tbf* para encontrar aquellos estados en los que el tiempo esperado hasta el fallo es muy superior al que finalmente se observa, lo que es un indicio de solapamiento perceptual. Es decir, situaciones que el sistema interpreta como iguales y que en realidad requieren la ejecución de acciones diferentes. Para poder eliminar estos solapamientos perceptuales hemos propuesto insertar, bajo ciertas condiciones, una nueva neurona en la red *Fuzzy ART*. Esta neurona representa un nuevo estado, lo que permite distinguir una nueva situación y aprender una nueva acción asociada. Si bien los resultados obtenidos con la inserción dinámica de neuronas vinculada al aprendizaje son excelentes, esta estrategia no se ha seguido utilizando en la tesis debido a que la siguiente mejora del sistema de aprendizaje hace innecesario el ajuste del parámetro de vigilancia.

El tercer problema abordado en esta tesis ha sido la robustez y estabilidad de los aprendizajes. Hemos propuesto el uso de un comité de aprendedores independientes, cada uno de ellos con su propio espacio de estados y acciones. Esta combinación de aprendedores subóptimos es una estrategia utilizada desde hace tiempo en el ámbito de las redes de neuronas artificiales para reducir la *varianza* mientras se mantiene un bajo *bias*. Esta estrategia proporciona una mayor robustez ante cambios en el entorno, ya que el sistema es mucho más estable. Para el buen funcionamiento del sistema es crítico que el comité de aprendedores disponga de individuos independientes. Por esta razón hemos propuesto que cada miembro del comité tenga su propia red *Fuzzy ART* con un parámetro de vigilancia aleatorio. Otra característica del comité es que sus miembros deben ser aprendedores débiles, por lo que la dependencia del parámetro de vigilancia se reduce aún más dado que no es necesario ajustarlo a un valor óptimo. Se han propuesto dos versiones de los comités: aprendedores de acción donde

cada aprendedor evalúa todo el espacio de acciones, y aprendedores de política, donde cada aprendedor únicamente evalúa un intervalo del espacio de acciones.

Un problema bien conocido de las redes *Fuzzy ART* es que la partición del espacio obtenida depende del orden de aparición de los ejemplos. En nuestro caso dependería del orden de aparición de las entradas sensoriales, que viene determinado por el proceso de exploración realizado durante el aprendizaje. Especialmente en las primeras fases este proceso se verá muy influido por los valores iniciales (aleatorios) del sistema. Tras analizar los resultados obtenidos no consideramos que este hecho afecte negativamente a nuestro sistema. Entendemos que esto es debido a la presencia de diferentes redes *Fuzzy ART* que, al disponer de parámetros de vigilancia diferentes, proporcionan la riqueza perceptual suficiente para mitigar este problema.

El sistema de aprendizaje mediante un comité de aprendedores consigue el aprendizaje rápido y estable de comportamientos, y permite realizar el proceso en el robot real y sin ningún conocimiento previo. Lo único que se requiere es una función de refuerzo y una función de recuperación que recoloque al robot en una posición adecuada para continuar con el aprendizaje tras un error. Dado que el sistema aquí presentado está inspirado para su funcionamiento en la robótica de servicios, hemos aprovechado la presencia del usuario para dotar al sistema de interactividad y que sea el propio usuario el que genere la señal de refuerzo y recoloque al robot en caso de error. Todas las órdenes del usuario se transmiten mediante un *joystick* inalámbrico, por lo que el sistema es sencillo e intuitivo de utilizar. La contrapartida es que nos enfrentamos a un nuevo problema. La señal de refuerzo pasa a tener una fuerte componente estocástica, ya que ante una misma situación el usuario puede proporciona refuerzos muy dispares. Hemos realizado experimentos con usuarios de diferentes edades y niveles educativos que debían enseñarle al robot a seguir una pared dentro de un entorno delimitado. La gran mayoría de los usuarios lograron una convergencia en pocos minutos, por lo que el sistema se muestra robusto ante refuerzos estocásticos, causados por cambios de opinión del usuario durante el aprendizaje.

La última mejora que hemos introducido en nuestro sistema de aprendizaje ha sido la identificación de los sensores más relevantes para la tarea a aprender y/o ejecutar. En este caso hemos decidido aplicar técnicas de información mutua para identificar aquellos sensores que proporcionan información relevante y, a su vez, eliminar aquellos que proporcionan información redundante. De esta forma el sistema es capaz de seleccionar de manera no supervisada únicamente aquellos sensores necesarios para la tarea, lo que acelera enormemente

el aprendizaje y aumenta la robustez del comportamiento aprendido. A la hora de calcular la información mutua es muy importante la frecuencia de aparición de las distintas lecturas sensoriales: existen situaciones muy poco frecuentes pero muy relevantes que podrían pasar desapercibidas. En este sentido hemos comprobado cómo el uso de las redes *Fuzzy ART* "apantalla" este efecto debido a que la inserción de estados no depende de la frecuencia de aparición de sus entradas. Este fenómeno se ha convertido por tanto en un beneficio adicional del empleo de las redes *Fuzzy ART* en nuestro sistema de aprendizaje. A través de los resultados experimentales hemos comprobado cómo la selección de sensores relevantes proporciona una mejora significativa en el tiempo de aprendizaje, especialmente en aquellas situaciones donde existe un gran número de sensores irrelevantes para la tarea.

Con el trabajo presentado en esta tesis hemos conseguido dotar a un robot real de la capacidad de un aprendizaje rápido, continuo y sin la necesidad de grandes ajustes ni de una fase diferenciada de simulación y calibración. Durante toda la tesis hemos empleado el comportamiento seguir pared como banco de pruebas para validar las distintas propuestas presentadas. También hemos probado nuestro sistema con un comportamiento de cruzar puerta y usando un *joystick* para indicar el refuerzo. Por último, como resultado de la colaboración entre la Universidade de Santiago de Compostela y la Universidad de Alcalá, hemos realizado un trabajo en el ámbito del transporte urbano autónomo donde se ha aplicado el aprendizaje por refuerzo siguiendo algunas de las estrategias presentadas en esta tesis, en particular se ha aprendido el comportamiento seguir líder [107, 130, 131]. Como trabajo futuro se planea trasladar el aprendizaje a la maniobra de enlace con el convoy, aprovechando para ello los resultados obtenidos en una tesis recientemente defendida [132].

7.1. Trabajo futuro

Los resultados obtenidos en esta tesis abren un amplio abanico de posibilidades para continuar investigando en el aprendizaje de robots de servicio.

El trabajo más inmediato es la integración del sistema de aprendizaje mediante comités con la selección de los sensores relevantes. Esto permitirá que un robot no sólo adapte su comportamiento sino también su forma de ver el mundo, para utilizar de manera más eficiente su conjunto de sensores.

Se debe estudiar el rendimiento del sistema en robots con un mayor número de grados de libertad como robots humanoides o brazos robóticos. En esta tesis se ha probado el sistema

únicamente con uno o dos grados de libertad. El crecimiento exponencial que supone añadir grados de libertad debe ser estudiado para identificar problemas o posibles mejoras en los algoritmos.

Esta tesis sienta las bases del aprendizaje continuo, de cara a lograrlo se requiere una experimentación exhaustiva en este aspecto. Se deberá probar el funcionamiento del robot durante periodos de tiempo prolongados en ambientes cotidianos.

El uso de un *joystick* para indicar el refuerzo al robot y recolocarlo tras un error permite combinar aprendizaje por demostración con aprendizaje por refuerzo. De esta manera el usuario no sólo le hará saber al robot que lo está haciendo mal, sino que podrá indicar de manera explícita el comportamiento deseado.

Para la realización de nuevas tareas será necesario el uso de diferentes sensores a los usados durante esta investigación. Dispositivos como videocámaras o sensores tridimensionales añaden nuevas posibilidades pero suponen un reto debido a la gran cantidad de datos que aportan. Actualmente se están realizando pruebas con el sensor Kinect de Microsoft, el cual proporciona una imagen tridimensional similar a la obtenida con una cámara de visión estéreo. La identificación de las zonas más importantes de la imagen permite focalizar la atención en esas zonas y reducir la carga de información del sistema.

Una posibilidad interesante es la combinación de nuestro sistema con un entorno inteligente dotado de agentes que observen lo que ocurre en él. Los agentes pueden identificar hechos relevantes que sucedan en el entorno y dirigir al robot hacia ese punto. Tanto para la navegación como para la realización de las tareas el robot utilizaría los comportamientos aprendidos, pero la planificación sería tarea del entorno inteligente.

Publicaciones derivadas de la tesis

Publicaciones en revistas

- Quintía, P.; Iglesias, R.; Rodríguez, M. & Regueiro, C. V.
  Learning on real robots from experience and simple user feedback.
  Journal of Physical Agents, 2013, 7 (1), 57-65

- Quintía, P.; Iglesias, R.; Rodríguez, M. A.; Regueiro, C. V. & Valdés, F.
  Learning in real robots from environment interaction.
  Journal of Physical Agents, 2012, 6 (1), 43-51

- Quintía, P.; Iglesias, R.; Rodríguez, M. A. & Regueiro, C. V.
  Simultaneous learning of perception and action in mobile robots.
  Robotics and Autonomous Systems, 2010, 58, 1306-1315

Publicaciones en congresos internacionales

- Quintía, P.; Iglesias, R.; Rodríguez, M. & Regueiro, C. V.
  Learning on real robots from their direct interaction with the environment
  Advances in Autonomous Robotics - Joint Proceedings of the 13th Annual TAROS Conference and the 15th Annual FIRA RoboWorld Congress, Springer, 2012, 7429, 444-445

- Quintía, P.; Iglesias, R.; Rodríguez, M. & Regueiro, C. V.
  Simultaneous learning of perceptions and actions in autonomous robots.
  ICINCO 2010. 7th International Conference on Informatics in Control, Automation and Robotics, 2010, 395-398

- Quintía, P.; Iglesias, R.; Rodríguez, M. & Regueiro, C. V.
  Variable Vigilance Fuzzy ART applied to robot learning.
  Towards Autonomous Robotic Systems (TAROS'10), 2010

- Kyriacou, T.; Iglesias, R.; Rodríguez, M. & Quintía, P.
  Unsupervised Complexity Reduction of Sensor Data for Robot Learning and Adaptation.
  Towards Autonomous Robotic Systems (TAROS'10), 2010

- Quintía, P.; Iglesias, R.; Rodríguez, M. & Regueiro, C. V.
  Fast robot learning through an ensemble of predictors able to forecast the time interval before failure.
  Towards Autonomous Robotics Systems 2009 (TAROS'09), 2009

- Quintía, P.; Iglesias, R.; Regueiro, C. V.; Cernadas, E. & Rodríguez, M.
  Robot learning from environment interaction and observation of human-behaviour.
  Towards Autonomous Robotic Systems (TAROS'08), 2008

Publicaciones en congresos nacionales

- Quintía, P.; Iglesias, R.; Rodríguez, M. A.; Regueiro, C. V. & Kyriacou, T.
  Selecting the most relevant sensors in a wall following behavior.
  XIII Workshop of Physical Agents (WAF), 2012

- Iglesias, R.; Rodríguez, M.; Quintía, P. & Regueiro, C. V.
  Continuous learning on a real robot through user feedback.
  XIII Workshop of Physical Agents (WAF), 2012

- Iglesias, R.; Rodríguez, M. A.; Quintía, P. & Regueiro, C. V.
  Continual learning in robots from environment interaction.
  XII Workshop of Physical Agents (WAF), 2011

- Rodríguez, M. A.; Iglesias, R.; Quintía, P. & Regueiro, C. V.
  Parallel robot learning through an ensemble of predictors able to forecast the time interval before a robot failure.
  XI Workshop of Physical Agents (WAF), 2010

– Quintía, P.; Regueiro, C. V.; Iglesias, R.; Rodríguez, M.; Gamallo, C. & Cernadas, E.
Control y Generación de Refuerzo a partir de la Observación del Comportamiento Humano.
X Workshop de Agentes Físicos (WAF), 2009

Otras publicaciones derivadas de colaboraciones mantenidas durante la realización de la tesis y que han sido relevantes para la misma

– Valdés, F.; Iglesias, R.; Espinosa, F.; Rodríguez, M. A.; Quintía, P. & Santos, C.
Implementation of robot routing approaches for convoy merging manoeuvres
Robotics and Autonomous Systems, 2012, 60, 1389-1399

– Valdés, F.; Iglesias, R.; Espinosa, F.; Rodríguez, M. A.; Quintía, P. & Santos, C.
Robot Routing Approaches for Convoy Merging Manoeuvers
Towards Autonomous RObotic Systems (TAROS), 2011, 241-252

– Rodríguez, M.; Iglesias, R.; Espinosa, F.; Quintía, P.; Regueiro, C. V. & Valdés, F.
Learning proposal based on reinforcement for collaborative tasks: robot convoy formation
4th European Conference on Mobile Robotics (ECMR'09), 2009

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Bibliografía

[1] Abarbanel, Henry D.I.: *Analysis of observed chaotic data*. Springer, 1996.

[2] Abbeel, Pieter, Adam Coates, Morgan Quigley y Andrew Y. Ng: *An Application of Reinforcement Learning to Aerobatic Helicopter Flight*. En *In Advances in Neural Information Processing Systems*, 2007.

[3] Albus, J. S.: *A new approach to manipulator control: The Cerebellar Model Articulation Controller (CMAC)*. Journal of Dynamic Systems, Measurement, and Control, 97(3):220–227, 1975.

[4] Alpaydin, Ethem: *Introduction to Machine Learning*. The MIT Press, 2nd edición, 2010, ISBN 026201243X, 9780262012430.

[5] Amari, Shun Ichi: *Natural gradient works efficiently in learning*. Neural Comput., 10(2):251–276, 1998, ISSN 0899-7667.

[6] Argall, Brenna D., Sonia Chernova, Manuela Veloso y Brett Browning: *A survey of robot learning from demonstration*. Robotics and Autonomous Systems, 57(5):469 – 483, 2009, ISSN 0921-8890.

[7] Arkin, Ronald C.: *Behavior-Based Robotics*. MIT Press, 1998.

[8] Atkin, R.: *From cohomology in physics to q-connectivity in social science*. International Journal of Man-Machines Studies, 4:341–362, 1972.

[9] Austermann, A. y S. Yamada: *"Good robot", "Bad robot" - Analyzing users' feedback in a human-robot teaching task*. En *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on*, páginas 41 –46, Agosto 2008.

[10] Baird, Leemon C y A Harry Klopf: *Reinforcement learning with high-dimensional, continuous actions*. Informe técnico, Wright Laboratory, Wright-Patterson Air Force Base, 1993.

[11] Bakker, Bram, Viktor Zhumatiy, Gabriel Gruener y Jürgen Schmidhuber: *Quasi-online Reinforcement Learning for Robots*. En *ICRA*, páginas 2997–3002, 2006.

[12] Bekey, George A.: *Autonomous Robots*. MIT Press, 2005.

[13] Bellman, Richard Ernest: *Dynamic Programming*. Princeton University Press, 1957.

[14] Bersekas, Dimitri P. y John N. Tsitsiklis: *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[15] Beyer, Hans Georg y Hans Paul Schwefel: *Evolution strategies - A comprehensive introduction*. Natural Computing, 1:3–52, 2002, ISSN 1567-7818.

[16] Bond, Alan H. y David H. Mott: *Learning of sensory-motor schemas in a mobile robot*. En *7th International Joint Conference on Artificial Intelligence*, páginas 159–161, 1981.

[17] Bradtke, Steven J.: *Reinforcement Learning Applied to Linear Quadratic Regulation*. En *In Advances in Neural Information Processing Systems 5*, páginas 295–302. Morgan Kaufmann, 1993.

[18] Bradtke, Steven J., Andrew G. Barto y Pack Kaelbling: *Linear Least-Squares algorithms for temporal difference learning*. En *Machine Learning*, páginas 22–33, 1996.

[19] Breiman, Leo: *Bagging predictors*. Mach. Learn., 24(2):123–140, aug 1996, ISSN 0885-6125.

[20] Brooks, Rodney A.: *Artificial Life and Real Robots*. En Varela, Francisco J. y Paul Bourgine (editores): *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, páginas 3–10, Cambridge, MA, USA, 1992. MIT Press.

[21] Cao, L. y Francis Tay: *Feature Selection for Support Vector Machines in Financial Time Series Forecasting*. En Leung, Kwong S., Lai Wan Chan y Helen Meng (editores): *Intelligent Data Engineering and Automated Learning - IDEAL 2000. Data Mining, Financial Engineering, and Intelligent Agents*, volumen 1983, capítulo 38, páginas 41–65. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[22] Carpenter, G A y S Grossberg: *ART2: Self-organization of stable category recognition codes for analog input patterns*. En *Proceedings of the IEEE First International Conference on Neural Networks*, volumen II, páginas 727–736, 1987.

[23] Carpenter, Gail A. y Stephen Grossberg: *Absolutely stable learning of recognition codes by a self-organizing neural network*. En *American Institute of Physics (AIP) Conference Proceedings 151: Neural Networks for Computing*, páginas 77–85, 1986.

[24] Carpenter, Gail A. y Stephen Grossberg: *ART 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures*. Neural Networks, 3(2):129–152, 1990, ISSN 0893-6080.

[25] Carpenter, Gail A., Stephen Grossberg y John H. Reynolds: *ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network*. Neural Netw., 4(5):565–588, sep 1991, ISSN 0893-6080.

[26] Carpenter, Gail A., Stephen Grossberg y David B. Rosen: *Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system*. Neural Networks, 4:759–751, 1991.

[27] Cassandra, Anthony R.: *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes*. Tesis de Doctorado, Brown University, Department of Computer Science, Providence, RI, USA, 1998.

[28] Celiberto, L.A., J.P. Matsuura, R. Bianchi y R. Lopez de Mantaras: *Reinforcement Learning with Case-Based Heuristics for RoboCup Soccer Keepaway*. En *IX Latin American Robotics Symposium / I Simposio Brasileiro de Robotica*, páginas 7–13, Fortaleza, Brasil, 2012.

[29] Cichosz, Pawel: *Reinforcement Learning by Truncating Temporal Differences*. Tesis de Doctorado, Department of Electronics and Information Technology, Warsaw University of Technology, 1997.

[30] Comité Español de Automática: *El libro blanco de la Robótica en España: Investigación, tecnologías y formación*. 2011.

[31] Dario, Paolo: *Robot Companions for Citizens*, 2013. `http://www.robotcompanions.eu/`.

[32] Dash, M. y H. Liu: *Feature Selection for Classification*. Intelligent Data Analysis, 1:131–156, 1997.

[33] Dietterich, Thomas G.: *Machine-Learning Research: Four Current Directions*. AI Magazine, 18(4):97–136, 1997.

[34] Domenech, José E., Carlos V. Regueiro, Cristina Gamallo y Pablo Quintía: *Learning Wall Following Behaviour in Robotics through Reinforcement and Image-based States*. En *IEEE International Symposium on Industrial Electronics (ISIE)*, Vigo (Spain), 2007.

[35] Doran, J.: *Planning and robots*. Machine Intelligence, 5:519–532, 1969.

[36] Durham, J. y F. Bullo: *Smooth Nearness-Diagram Navigation*. En *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008)*, páginas 690–695, 2008.

[37] Eiben, A.E. y J.E. Smith: *Introduction to Evolutionary Computing,*. Springer, 2003.

[38] EUROP, European Robotics Technology Platform: *Robotic Visions: To 2020 and beyond*. The strategic research agenda for robotics in Europe, July 2009.

[39] Farias, Daniela de y Nimrod Megiddo: *Exploration-Exploitation Tradeoffs for Experts Algorithms in Reactive Environments*. En Saul, Lawrence K., Yair Weiss y Léon Bottou (editores): *Advances in Neural Information Processing Systems 17*, páginas 409–416. MIT Press, Cambridge, MA, 2005.

[40] Freund, Yoav y Robert E Schapire: *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*. Journal of Computer and System Sciences, 55(1):119 – 139, 1997, ISSN 0022-0000.

[41] Gaskett, Chris, David Wettergreen y Alexander Zelinsky: *Q-Learning in Continuous State and Action Spaces*. En *Proceedings of the 12th Australian Joint Conference on*

Artificial Intelligence: Advanced Topics in Artificial Intelligence, AI '99, páginas 417–428, London, UK, 1999. Springer-Verlag.

[42] Geman, S., E. Bienenstock y R. Doursat: *Neural networks and the bias/variance dilemma*. Neural Computation, 4:1–58, 1992.

[43] Gerkey, Brian P., Richard T. Vaughan y Andrew Howard: *The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems*. En *In Proceedings of the 11th International Conference on Advanced Robotics, 2003*, páginas 317–323, 2003.

[44] Grossberg, Stephen: *Adaptive pattern classification and universal recoding: II. Feedback, expectation, olfaction, illusions*. Biological Cybernetics, 23(4):187–202, 1976.

[45] Grossberg, Stephen: *How does the brain build a cognitives code?* Psychological Review, 87:1–51, 1980.

[46] Hall, Mark A.: *Correlation-based Feature Selection for Machine Learning*. Tesis de Doctorado, The University of Waikato, 1999.

[47] Hartley, Richard y Andrew Zisserman: *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000, ISBN 0521623049.

[48] Hauskrecht, Milos: *Value-Function Approximations for Partially Observable Markov Decision Processes*. Journal of Artificial Intelligence Research, 13:33–94, 2000.

[49] Iglesias, R., M. Fernández-Delgado y S. Barro: *Learning of perceptual states in the design of an adaptive wall-following behaviour*. En *European Symposium on Artificial Neural Networks*, Bruges (Belgium), 2000.

[50] Iglesias, R., U. Nehmzow y S. A. Billings: *Model identification and model analysis in robot training*. Robotics and Autonomous Systems, 56(12):1061–1067, dec 2008, ISSN 0921-8890.

[51] Iglesias, R., C.V. Regueiro, J. Correa, E. Sánchez y S. Barro: *Improving wall following behaviour in a mobile robot using reinforcement learning*. En *ICSC International symposium on engineering of intelligent systems (EIS'98)*, Tenerife (España), 1998. ISBN 3-906454-12-6.

[52] Iglesias, R., M. Rodríguez, C.V. Regueiro, J. Correa y S. Barro: *Combining Reinforcement Learning and Genetic Algorithms to Learn Behaviours in Mobile Robotics*. En *International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, Setúbal, Portugal, 2006.

[53] Iglesias, R., M. Rodríguez, M. Sánchez, E. Pereira y C.V. Regueiro: *Improving reinforcement learning through a better exploration strategy and an ajustable representation of the environment*. En *3rd European Conference on Mobile Robotics (ECMR 07)*, Freiburg (Germany), September 2007.

[54] Iglesias, Roberto, Ulrich Nehmzow, Theocharis Kyriacou y Steve Billings: *Training and Analysis of Mobile Robot Behaviour Through System Identification*. En Marín, Roque, Eva Onaindía, Alberto Bugarín y José Santos (editores): *Current Topics in Artificial Intelligence*, volumen 4177 de *Lecture Notes in Computer Science*, páginas 470–479. Springer Berlin / Heidelberg, 2006.

[55] Iglesias, Roberto, Miguel Angel Rodríguez, Pablo Quintía y Carlos V. Regueiro: *Continual learning in robots from environment interaction*. En *XII Workshop of Physical Agents*, Albacete, 2011.

[56] Iravani, Pejman: *Discovering relevant sensor data by q-analysis*. En Bredenfeld, Ansgar, Adam Jacoff, Itsuki Noda y Yasutake Takahashi (editores): *In RoboCup 2005: Robot Soccer World Cup IX*, Lecture Notes in Computer Science. Springer-Verlag, 2006, ISBN 3-540-35437-9.

[57] Jordan, Michael I. y Robert A. Jacobs: *Hierarchical Mixtures of Experts and the EM Algorithm*. Neural Computation, 6(2):181–214, 1994.

[58] Kaelbling, Leslie Pack, Michael L. Littman y Anthony R. Cassandra: *Planning and acting in partially observable stochastic domains*. Artificial Intelligence, 101(1- 2):99 – 134, 1998.

[59] Kaelbling, L.P., M.L. Littman y A.W. Moore: *Reinforcement learning: A survey*. Journal of Artificial Intelligence Research, 4:237–285, 1996.

[60] Kamio, S. y H. Iba: *Adaptation technique for integrating genetic programming and reinforcement learning for real robots*. Evolutionary Computation, IEEE Transactions on, 9(3):318 – 333, june 2005, ISSN 1089-778X.

[61] Kearns, Michael y Satinder Singh: *Near-Optimal Reinforcement Learning in Polynomial Time*. Mach. Learn., 49:209–232, November 2002.

[62] Kharchenko, S.: *While robots are learning to walk*. Coal International, 2(1), 1983.

[63] Kira, K. y L. A. Rendell: *A practical approach to feature selection*. En *Machine Learning: Proceedings of the Ninth International Conference*, 1992.

[64] Knox, W. Bradley y Peter Stone: *Interactively Shaping Agents via Human Reinforcement: The TAMER Framework*. En *Fifth International Conference on Knowledge Capture*, California (USA), September 2009.

[65] Knox, W. Bradley y Peter Stone: *Reinforcement Learning with Human Feedback in Mountain Car*. En *AAAI Spring 2011 Symposium on Bridging the Gaps in Human-Agent Collaboration*, 2011.

[66] Kober, J. y J. Peters: *Policy search for motor primitives in robotics*. Machine Learning, 84(1-2):171–203, 2011.

[67] Kober, Jens, Betty Mohler y Jan Peters: *Learning perceptual coupling for motor primitives*. En *in Proceedings of the IEEE/RSJ 2008 International Conference on Intelligent RObots and Systems (IROS*, páginas 834–839, 2008.

[68] Kohonen, T.: *Self-Organizing Maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001, ISBN 3540679219.

[69] Kohonen, T., J. Hynninen, J. Kangas, J. Laaksonen y K. Torkkola: *LVQ PAK: The Learning Vector Quantization Program Package*. Informe técnico, 1995.

[70] Kohonen, Teuvo: *Self-organized formation of topologically correct feature maps*. Biological Cybernetics, 43:59–69, 1982, ISSN 0340-1200. 10.1007/BF00337288.

[71] Kollar, T. y N. Roy: *Using reinforcement learning to improve exploration trajectories for error minimization*. En *IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*, páginas 3338 –3343, may 2006.

[72] Konda, V. R. y V. Borkar: *Actor-critic type learning algorithms for Markov decision processes*. SIAM Journal on Control and Optimization, 38(1):94–123, 1999.

[73] Kormushev, P., S. Calinon y D. G. Caldwell: *Robot Motor Skill Coordination with EM-based Reinforcement Learning*. En *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, páginas 3232–3237, Taipei, Taiwan, October 2010.

[74] Kyriacou, Theocharis, Roberto Iglesias, Miguel Rodríguez y Pablo Quintía: *Unsupervised Complexity Reduction of Sensor Data for Robot Learning and Adaptation*. En *Towards Autonomous RObotic Systems (TAROS)*, Plymouth (UK), 2010.

[75] Lagoudakis, Michail G. y Ronald Parr: *Least-Squares Policy Iteration*. Journal of Machine Learning Research, 4:1107–1149, 2003.

[76] León, A., E. F. Morales, L. Altamirano y J. R. Ruiz: *Teaching a Robot New Tasks through Imitation and Feedback*. En *In Proc. Workshop: New Developments in Imitation Learning, as part of the Internacional Conference on Machine Learning (ICML-2011)*, volumen 7042 LNCS de *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, páginas 549–556, 2011.

[77] Liu, Huan y Lei Yu: *Toward integrating feature selection algorithms for classification and clustering*. IEEE Transactions on Knowledge and Data Engineering, 17:491–502, 2005.

[78] Mataric, Maja J.: *The Robotics Primer*. MIT Press, 2007, ISBN 978-0-262-63354-3.

[79] Merriam Webster U.S.: *Webster's Third New International Dictionary*, 3rd edición, 2000.

[80] Moore, B.: *ART 1 and pattern clustering*. En Touretzky, D., G. Hinton y T. Sejnowski (editores): *Proceedings of the 1988 Connectionist Models Summer School*, páginas 174–1985. Morgan Kaufmann, 1988.

[81] Moreno, D. L., C. V. Regueiro, R. Iglesias y S. Barro: *Using prior knowledge to improve reinforcement learning in mobile robotics*. En *Towards Autonomous RObotic Systems (TAROS)*, Essex. UK, 2004.

[82] Moreno, D. L., C. V. Regueiro, R. Iglesias y S. Barro: *Making use of unelaborated advice to improve reinforcement learning: A mobile robotics approach*. En *III*

International Conference on Advances in Pattern Recognition (ICAPR), páginas 89–98, Bath. UK, 2005. Springer Berlin Heidelberg.

[83] Naftaly, Ury, Nathan Intrator y David Horn: *Optimal Ensemble Averaging of Neural Networks*. Network, 8:283–296, 1997.

[84] Nehmzow, Ulrich: *Mobile Robotics. A Practical Introduction*. Springer, 2nd edición, 2003.

[85] Nehmzow, Ulrich, Roberto Iglesias, Theocharis Kyriacou y Stephen A. Billings: *Robot learning through task identification*. Robotics and Autonomous Systems, 54(9):766 – 778, 2006, ISSN 0921-8890. Selected papers from the 2nd European Conference on Mobile Robots (ECMR '05).

[86] Ng, Andrew Y. y Stuart Russell: *Algorithms for inverse reinforcement learning*. En *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[87] Ogale, N. A.: *A survey of techniques for human detection from video*. Informe técnico, University of Maryland, 2006.

[88] Parsons, Lance, Ehtesham Haque y Huan Liu: *Subspace clustering for high dimensional data: a review*. SIGKDD Explor. Newsl., 6:90–105, June 2004, ISSN 1931-0145.

[89] Peng, Hanchuan, Fuhui Long y C. Ding: *Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 27(8):1226 –1238, aug. 2005, ISSN 0162-8828.

[90] Peters, Jan y Stefan Schaal: *Natural Actor-Critic*. Neurocomputing, 71(7-9):1180 – 1190, 2008, ISSN 0925-2312.

[91] Pfeifer, Rolf y Christian Scheier: *Understanding Intelligence*. MIT Press, 1999.

[92] Polikar, R.: *Ensemble based systems in decision making*. Circuits and Systems Magazine, IEEE, 6(3):21–45, quarter 2006, ISSN 1531-636X.

[93] Quintía, Pablo, Roberto Iglesias, Miguel A. Rodríguez y Carlos V. Regueiro: *Simultaneous learning of perception and action in mobile robots*. Robotics and

Autonomous Systems, 58(12):1306 – 1315, 2010, ISSN 0921-8890. Intelligent Robotics and Neuroscience.

[94] Quintía, Pablo, Roberto Iglesias, Miguel A. Rodríguez, Carlos Vázquez Regueiro y Fernando Valdés: *Learning in real robots from environment interaction*. Journal of Physical Agents, 6(1), 2012, ISSN 1888-0258.

[95] Quintía, P., J. E. Domenech, C. V. Regueiro, C. Gamallo y R. Iglesias: *Learning a wall following behaviour in mobile robotics using stereo and mono vision*. En *IX Workshop on Physical Agents (WAF)*, Vigo, Spain, 2008.

[96] Quintía, P., R. Iglesias, M. Rodríguez y C. V. Regueiro: *Simultaneous learning of perceptions and actions in autonomous robots*. En *ICINCO 2010. 7th International Conference on Informatics in Control, Automation and Robotics*, páginas 395–398, Funchal, Portugal, June 2010.

[97] Quintía, P., R. Iglesias, M. Rodríguez y C. V. Regueiro: *Learning on real robots from experience and simple user feedback*. Journal of Physical Agents, 7(1):57–65, 2013.

[98] Quintía, P., R. Iglesias, M. A. Rodríguez, C.V. Regueiro y T. Kyriacou: *Selecting the most relevant sensors in a wall following behavior*. En *XIII Workshop of Physical Agents*, 2012.

[99] Quintía, P., R. Iglesias, M.A. Rodríguez y C. V. Regueiro: *Learning on real robots from their direct interaction with the environment*. En *Advances in Autonomous Robotics - Joint Proceedings of the 13th Annual TAROS Conference and the 15th Annual FIRA RoboWorld Congress*, volumen 7429 de *Lecture notes in Computer Science*, páginas 444–445, Bristol, August 2012. Springer.

[100] Quintía, Pablo, Roberto Iglesias, Carlos V. Regueiro, Eva Cernadas y Miguel Rodríguez: *Robot learning from environment interaction and observation of human-behaviour*. En *Towards Autonomous Robotic Systems (TAROS)*, Edinburgh, Scotland, UK, September 2008.

[101] Quintía, Pablo, Roberto Iglesias, Miguel Rodríguez y Carlos V. Regueiro: *Fast robot learning through an ensemble of predictors able to forecast the time interval before failure*. En *Towards Autonomous Robotics Systems 2009 (TAROS'09)*, 2009.

[102] Quintía, Pablo, Roberto Iglesias, Miguel Rodríguez y Carlos V. Regueiro: *Variable Vigilance Fuzzy ART applied to robot learning*. En *Towards Autonomous RObotic Systems (TAROS)*, Plymouth (UK), 2010.

[103] Quintía, Pablo, Carlos V. Regueiro, Roberto Iglesias, Miguel Rodríguez, Cristina Gamallo y Eva Cernadas: *Control y Generación de Refuerzo a partir de la Observación del Comportamiento Humano*. En *X Workshop de Agentes Físicos*, Cáceres, Spain, September 2009.

[104] Regueiro, C. V., J. E. Domenech, R. Iglesias y J. L. Correa: *Acquiring Contour Following Behaviour in Robotics through Q-Learning and Image-based States*. En *XV Int. Conf. on Computer and Information Sciences Engineering (CISE)*, páginas 403–409, 2006.

[105] Regueiro, C.V.: *Arquitectura fractal de especialistas para robots móviles autónomos*. Tesis de Doctorado, Universidade de Santiago de Compostela, 2002.

[106] Regueiro, C.V., J.E. Domenech, D.L. Moreno y R. Iglesias: *Aprendizaje por refuerzo de un comportamiento visual seguir contorno en robótica móvil*. En *VII Workshop on Physical Agents (WAF)*, 2006.

[107] Rodríguez, M., R. Iglesias, F. Espinosa, P. Quintía, C. V. Regueiro y F. Valdés: *Learning proposal based on reinforcement for collaborative tasks: robot convoy formation*. En *4th European Conference on Mobile Robotics (ECMR'09)*, 2009.

[108] Rodríguez, M., R. Iglesias, C. V. Regueiro, J. Correa y S. Barro: *Autonomous and fast robot learning through motivation*. Robotics and Autonomous Systems, 55:735–740, 2007.

[109] Rodríguez, M., R. Iglesias, C.V. Regueiro, J. Correa y S. Barro: *Improving reinforcement learning in mobile robotics through its combination with generic algorithms*. En *Towards Autonomous RObotics Systems (TAROS)*, Guildford, UK, 2006.

[110] Rodríguez, M. A., R. Iglesias, P. Quintía y C. V. Regueiro: *Parallel robot learning through an ensemble of predictors able to forecast the time interval before a robot failure*. En *XI Workshop of Physical Agents (WAF)*, Valencia, 2010.

[111] Rokach, Lior: *Pattern Classification Using Ensemble Methods*, volumen 75 de *Series in Machine Perception and Artificial Intelligence*. World Scientific Publishing, 2010.

[112] Russell, Stuart J. y Peter Norvig: *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003, ISBN 0137903952.

[113] Sapankevych, N. y R. Sankar: *Time Series Prediction Using Support Vector Machines: A Survey*. Computational Intelligence Magazine, IEEE, 4(2):24–38, may 2009.

[114] Schapire, Robert E.: *The Strength of Weak Learnability*. Machine Learning, 5:197–227, 1990.

[115] Seltzer, Donald S.: *Use of sensory information for improved robot learning*. Manufacturing Engineering Transactions, páginas 64–68, 1980.

[116] Sharkey, Amanda J. C.: *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*. Springer, 1999.

[117] Siciliano, Bruno y Oussama Khatib (editores): *Springer Handbook of Robotics*. Springer, 2008, ISBN 978-3-540-23957-4.

[118] Simons, J., H. van Brussel, J. de Schutter y J. Verhaert: *Self-learning automation with variable resolution for high precission assembly by industrial robots*. IEEE Transactions on Automatic Control, AC-27(5):1109–1112, 1982.

[119] Skinner, Burrhus F.: *Science and Human Behavior*. New York: Macmillan, 1953.

[120] Sobol', I.: *Sensitivity estimates for nonlinear mathematical models*. Matematicheskoe Modelirovanie, 2:112–118, 1990.

[121] Stone, P., R. S. Sutton y G. Kuhlmann: *Reinforcement learning for RoboCup soccer keepaway*. Adaptive Behavior, 13(3):165–188, 2005.

[122] Suay, H. B. y S. Chernova: *Effect of human guidance and state space size on Interactive Reinforcement Learning*. En *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, páginas 1–6, 2011.

[123] Sutton, Richard S. y Andrew G. Barto: *Reinforcement learning: An introduction*. MIT Press, 1998.

[124] Tham, Chen K.: *Reinforcement learning of multiple tasks using a hierarchical CMAC architecture*. Robotics and Autonomous Systems, 15:247–274, 1995.

[125] Thomaz, A. L., G. Hoffman y C. Breazeal: *Reinforcement Learning with Human Teachers: Understanding how people want to teach robots*. En *In Proceedings of the 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2006.

[126] Thomaz, Andrea Lockerd, Guy Hoffman y Cynthia Breazeal: *Real-time interactive reinforcement learning for robots*. En *AAAI Workshop on Human Comprehensible Machine Learning*, 2005.

[127] Thorndike, E.: *Animal Intelligence*. Macmillan, 1911.

[128] Thorndike, E.: *The fundamentals of Learning*. New York: Teachers Collegue Press, 1932.

[129] Thrun, S. y T. M. Mitchell: *Lifelong robot learning*. Robotics and Autonomous Systems, 15(1-2):25–46, 1995.

[130] Valdés, F., R. Iglesias, F. Espinosa, M. Rodríguez, P. Quintía y C. Santos: *Robot Routing Approaches for Convoy Merging Maneuvers*. En *Towards Autonomous RObotic Systems (TAROS)*, páginas 241–252, Sheffield, 2011.

[131] Valdés, F., R Iglesias, F. Espinosa, M. A. Rodríguez, P. Quintía y C. Santos: *Implementation of robot routing approaches for convoy merging manoeuvres*. Robotics and Autonomous Systems, 60:1389–1399, 2012.

[132] Valdés Villarrubia, Fernando: *Estrategia de enrutamiento para la maniobra de enlace a un convoy de vehículos en entornos urbanos, robusta a la incertidumbre en los tiempos de recorrido*. Tesis de Doctorado, Universidad de Alcalá, 2012.

[133] Verschure, P., T. Voegtlin y R. J. Douglas: *Environmentally mediated synergy between perception and behaviour in mobile robots*. Nature, 425(6958):620–624, 2003.

[134] Wang, Y., M. Huber, V. N Papudesi y D. J. Cook: *User-Guided Reinforcement Learning of Robot Assistive Tasks for an Intelligent Environment*. En *Proceedings of the 2003 IEEWRSJ Inn. Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, USA, October 2003.

[135] Watkins, C.: *Learning from Delayed Rewards*. Tesis de Doctorado, University of Cambridge, England, 1989.

Índice de figuras

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Índice de tablas

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Lista de algoritmos

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA