Universidad Autónoma de Yucatán
Facultad de Matemáticas

Sistema Inteligente con Integración Multi-Sensorial y Aprendizaje de Abstracciones

TESIS

Presentado por:
I.E. Edgar de Jesús Ek Chacón

En opción al título:
Maestro en Ciencias de la computación

Asesor:
Dr. Víctor Emanuel Uc Cetina

Mérida, Yucatán, México.
Mayo 17

Dedicatoria

Dedico este trabajo al Dios Todopoderoso, al Dios de Abraham, Jacob e Isaac. Porque él es un buen Padre proveyendo todo en todo lo que hagamos. Por que él es la fuente de toda sabiduría e inteligencia. A él sea la gloria.

Agradecimientos

Mi más profundo agradecimento a Dios que proveyó todo lo necesario para terminar este trabajo, gracias a mis padres por el apoyo y motivación que siempre demostraron durante todo este tiempo, gracias a los compañeros de salón con quienes vivimos buenos momentos, gracias a todos mis maestros que sin duda fueron de mucho apoyo para componer este trabajo, y en especial, a mi asesor de Tesis Dr. Victor Emanuel Uc Cetina que siempre me demostró su apoyo y motivación incondicional. A todos, Dios les bendiga.

Resumen

La necesidad de la autonomía en un agente es para su supervivencia, evitar peligros o simplemente para realizar una determinada tarea por sí mismo en su mundo. Esta capacidad en los robots móviles abre una enorme gama de aplicaciones ya que se pueden desenvolver en muchos entornos ya sea en el suelo, en el aire, bajo el agua o en el espacio exterior. En la mayoría de los robots móviles se han utilizado algoritmos de planeación, pero éstos no han dado resultados precisos en todas las condiciones. La inteligencia artificial tuvo una tendencia de enfoque en los métodos basados en experiencias o algoritmos de aprendizaje que mejoran la autonomía de los robots. En este trabajo se desarrolla una arquitectura de agente inteligente basado en aprendizaje por refuerzo que cuenta con un algoritmo de red neuronal artificial que modela la integración multisensorial y mejora este sistema para percibir suficiente información del ambiente para la toma de decisiones. Se hicieron los experimentos necesarios, los cuales fueron la generación de unas curvas de aprendizaje para el algoritmo Intra-option Q-learning de un sólo paso y generación de curvas de aprendizaje con el nuevo algoritmo generado llamado Intra-option + SOM, entonces haciendo una comparación entre las curvas de aprendizaje del primer algoritmo y las curvas de aprendizaje del segundo se puede observar que el rendimiento es el mismo, con la ventaja de que el segundo algoritmo tiene la capacidad de integración multisensorial.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Los robots autónomos son máquinas inteligentes capaces de realizar tareas en el mundo real por sí mismos sin un control humano explícito. Las tareas que realizan estos robots son muy variados dependiendo del entorno y la aplicación, estos entornos pueden ser lugares terrestres muy peligrosos como las minas, transporte de munición o desactivación de bombas; lugares difíciles de acceder donde son usados helicópteros autónomos, lugares debajo de los oceanos donde la presión hidrostática impide a los humanos llegar, así como lugares en el espacio exterior o en la superficie de algún planeta.

Un ejemplo claro de la necesidad de la autonomía es cuando los Rovers son enviados a explorar la superficie del planeta Marte. Debido a la dificultad de la comunicación existente entre la Tierra y Marte fue necesario desarrollar sistemas capaces de entender órdenes de alto nivel, es decir, que cuenten con un alto grado de autonomía de tal manera que al recibir tan sólo un comando desde la Tierra pueda ejecutar todas las actividades de exploración requeridas en la superficie de Marte. De otra manera sería necesario dejar estas actividades a operadores humanos que los haga desde la Tierra, pero hay enormes desventajas, porque es necesario evaluar la situación del Rover y hacer decisiones usando la inteligencia y creatividad humana, esto lleva mucho tiempo y considerando también que la comunicación puede tomar hasta 20 minutos en promedio de ida y vuelta entre Marte y Tierra. Con la autonomía los robots pueden reaccionar rápidamente ante un peligro o sencillamente realizar de manera óptima una determinada tarea en la superficie de Marte y lo mismo pueden hacer en otros entornos. Esta capacidad abre una enorme cantidad de tareas que pueden llevar a cabo estos agentes en el mundo real.

En la sección 3.1.2 se describen algunos robots que han sido construídos y la mayoría de ellos utilizan algoritmos de planeación para navegación del terreno donde se encuentran y llevar a cabo sus actividades, sin embargo, debido a que el mundo real es un ambiente dinámico los movimientos del robot tienen una cierta cantidad de aleatoriedad, entonces

estas soluciones no han dado resultados precisos en todas las condiciones. Para mejorar la capacidad autónoma de los robots, la inteligencia artificial tuvo una tendencia de enfoque en los métodos basados en experiencias. En general, estos métodos pueden ser computacionalmente menos costosos y más fáciles que los métodos de planeación [22].

Uno de éstos métodos basados en la experiencia es el algoritmo Q-learning, el cual tiene muchas características que lo hacen adecuado para resolver el problema de navegación en un entorno dinámico. En primer lugar, el agente que usa este método es un agente de aprendizaje por refuerzo que no tiene conocimientos previos sobre su entorno de trabajo, aprende sobre el medio ambiente al interactuar con él [22].

En segundo lugar, es un agente de aprendizaje en línea, lo cual significa que aprende a tomar la mejor acción en cada estado o situación por prueba-error. Además, puede elegir acciones de forma aleatoria y calcular el valor para tomar una acción en un estado específico. Mediante la evaluación de cada par estado-acción se puede construir una política para trabajar en el medio ambiente. En el problema de navegación del robot conforme encuentra un camino libre de colisiones también necesita encontrar la mejor acción a tomar en cada estado. El agente obtiene este conocimiento mientras navega por su entorno. Q-learning es una alternativa muy atractiva debido a que es muy simple [22].

El objetivo principal de este trabajo es diseñar una arquitectura de agente inteligente que cuente tanto con la capacidad de integración multisensorial como con la habilidad de aprender tareas a diferentes niveles de abstracción.

El algoritmo desarrollado en este trabajo logra hacer una conexión entre dos áreas del aprendizaje automático, estas son el aprendizaje por refuerzo, el cual nos ofrece un método de aprendizaje muy eficiente llamado Intra-options Q-learning de un sólo paso que tiene la propiedad de aprendizaje de abstracciones; y el aprendizaje no-supervisado, el cual nos ofrece un algoritmo de redes neuronales que tiene la propiedad de integración multisensorial. La integración de estos dos algoritmos habilita a nuestro sistema inteligente la capacidad de percibir su mundo real, aprender de ella interactuando, y con el tiempo por experiencia poder conocer cuáles son las mejores acciones, con diferentes niveles de abstracción, a elegir para lograr de manera óptima su objetivo. Lo novedoso de este sistema basado en aprendizaje por refuerzo es que cuenta con un algoritmo de red neuronal artificial que modela la integración multisensorial que mejora el sistema para percibir suficiente información para la toma de desiciones o la elección de la acción a ejecutar. Las acciones que puede ejecutar este sistema pueden ser de diferentes niveles de abstracción. Una manera de entender esto es por ejemplo, un robot que tiene la tarea de abrir una puerta, un nivel bajo de abstracción de aprendizaje es que el robot aprende esta actividad como una serie de acciones, entonces desde un determinado punto camina una serie de pasos (acciones) a la derecha, y luego otra

serie (acciones) a la izquierda, y así hasta llegar a la puerta, luego alzar (acción) su brazo hasta la manija, agarrarla (acción), girarla (acción), y jalar (acción) la puerta para abrirla; un nivel alto de abstracción es por ejemplo aprender que abrir una puerta consiste en esa serie de acciones pero aprenderlo como una sola acción a ejecutar, abrir puerta.

1.1. Planteamiento del Problema

Un sistema común en aprendizaje por refuerzo consiste de un agente que interactúa en un ambiente para aprender a lograr un objetivo de una manera óptima. Algunos ejemplos de aplicaciones en aprendizaje por refuerzo son *el acrobot* y *el elevador* [45]; en el primero se usa Sarsa y en el segundo Q-learning como métodos de aprendizaje, ambos robots usan la arquitectura común de un agente inteligente, Fig. 2.1. El elevador utiliza una red neuronal de retropropagación para representar la función valor-acción, debido a que éste maneja variables continuas, por lo tanto, utiliza un vasto número de estados que son imposibles de administrarlos en una tabla Q. El acrobot utiliza variables continuas para determinar su estado, pero no utiliza ningún tipo de red neuronal artificial. La formulación de estos sistemas siempre incluye tres aspectos: percepción, acción y objetivo. Haciendo enfoque a la percepción el elevador, utiliza una red neuronal pero sólo para almacenamiento de los valores Q. Sin ningún objetivo de modelar alguna función cerebral, el acrobot simplemente no utiliza ninguno [45].

El presente estudio tiene como propósito diseñar una arquitectura de agente inteligente que pueda percibir su mundo real por medio de una red neuronal artificial que modela una función del cerebro llamada *integración multisensorial*, el cual es una capacidad que tienen los humanos y otros animales para integrar señales de varios sentidos (visión, audición, etc.) que vienen del mundo exterior para localizar objetos con precisión y luego tomar decisiones propias en diferentes niveles de abstracción a partir de estas señales integradas.

Para hacer frente a este diseño de arquitectura utilizamos dos métodos, el método de aprendizaje por refuerzo que dota al agente para la toma de decisiones llamado *Intra-option Q-learning* y el método de aprendizaje no supervisado que da la capacidad de integración multisensorial llamado red neuronal artificial *mapa auto-organizado*.

1.1.1. Objetivo General

- Diseñar una arquitectura de agente inteligente que cuente tanto con la capacidad de integración multisensorial como con la habilidad de aprender tareas a diferentes niveles de abstracción.

1.1.2. Objetivos Específicos

- Identificar un modelo neuronal artificial que permita integrar la información multisensorial de un agente inteligente que aprende por refuerzo.
- Determinar una forma eficiente de utilizar la integración multisensorial con un agente que aprende por refuerzo tareas con diferentes niveles de abstracción, mediante métodos de procesos de decisión semi-Markov.
- Generar varios conjuntos de ejemplos con un determinado método de distribución estadístico para experimentación.
- Evaluar el desempeño del nuevo algoritmo sobre cada uno de los conjuntos de ejemplos generados.

1.1.3. Justificación

La autonomía en la robótica es la clave para la solución idónea de muchas tareas que son tediosas, difíciles y peligrosas, las cuales están presentes en la industria, transporte, entornos peligrosos, exploración espacial, etc.

- En la industria, por ejemplo, un proyecto reciente de Carnegie Mellon ha demostrado que los robots pueden eliminar la pintura de grandes barcos, cincuenta veces más rápido que las personas. Prototipos de robots mineros autónomos son más rápidos y precisos que los humanos para el transporte de mineral en minas subterráneas [44].
- En el transporte tiene varias facetas que van desde helicópteros autónomos que entregan objetos en lugares que podrían ser difíciles de acceder por otros medios, a sillas automáticas que transportan personas que son incapaces de controlarlas por sí mismas [44].
- En entornos peligrosos, algunos países han utilizado robots para transportar munición y desactivar bombas [44].
- En la exploración espacial los robots han ido donde nadie ha ido antes, incluyendo la superficie de Marte. Los brazos robóticos asisten a los astronautas en la colocación y recogida de satélites y la construcción de la Estación Espacial Internacional. Los robots también ayudan a explorar por debajo del mar, y son rutinariamente utilizados para obtener mapas de barcos hundidos [44].

- Para atacar el problema de navegación de terreno en robots móviles se han utilizado algoritmos de planeación, pero no han dado resultados precisos en todas las condiciones. La inteligencia artificial tuvo una tendencia de enfoque en los métodos basados en experiencias o algoritmos de aprendizaje que mejoran la autonomía de los robots [22].

La mayoría de los robots existentes para estas tareas son teleoperados, es decir, un humano los opera mediante control remotos, y algunos tienen cierto nivel de autonomía. Proveer una autonomía completa a este tipo de robots es un siguiente paso muy importante. La razón de la autonomía (entendida como la independencia del robot con respecto al control humano) es que el supervisor humano puede encontrarse a miles de kilómetros del sistema robótico y el robot debe reaccionar o tomar decisiones propias frente a un entorno hostil o en continuo cambio [44].

Conseguir que un algoritmo sea capaz de modelar el culículo superior en sus funciones como la integración multisensorial y la generación de órdenes motoras es un valioso aporte para que un agente pueda navegar en un mundo desconocido, aprender de ella y tomar decisiones propias para lograr objetivos.

Conocemos que en el mundo físico es necesario sensores en un robot para percibir el mundo real, sin embargo, según Etzioni y Weld, 1994, también los softbots o agentes de software perciben de manera similar que los robots, sólo que un mundo de bits. Entonces al igual que los robots, los softbots necesitan tomar sus propias decisiones en ambientes muy complejos o ambientes con una basta cantidad de información como es en la internet. Por lo tanto, este algoritmo aportado no se limita su aplicación sólo en agentes que viven en el mundo real si no también en agentes que habitan en un mundo de bits.

Este trabajo contiene los siguientes capítulos: en el Capítulo 2 Marco Teórico, se describen los métodos usados que sirvieron de base para generar el algoritmo desarrollado, en el Capítulo 3 Estado del Arte, se describen algunos robots y softbots haciendo enfoque sobre sus capacidades autónomas, en el Capítulo 4 Metodología se tiene el desarrollo del trabajo, se tiene el Capítulo 5 Experimentos, en el cual se generaron las curvas de aprendizaje del algoritmo desarrollado, en el Capítulo 6 Discusión de Resultados se describe cómo se llega al objetivo planteado, por último en el Capítulo 7 Conclusiones se tiene la conclusión y el trabajo futuro.

Capítulo 2

Marco Teórico

2.1. Aprendizaje Automático

El aprendizaje automático se refiere a un conjunto de algoritmos computacionales diseñados para lograr que un sistema computacional pueda mejorar automáticamente su desempeño con la experiencia. Es un campo interdisciplinario y subárea esencial de la inteligencia artificial [32].

Dependiendo de la forma en que se defina experiencia se puede hablar de tres tipos generales de aprendizaje automático:

- En aprendizaje supervisado un maestro proporciona un conjunto de ejemplos con la respuesta correcta o etiquetas, sobre éstos el algoritmo hace una generalización para responder bien a futuros ejemplos de entrada. Es también llamado aprendizaje basado en ejemplos [30].

- En aprendizaje no supervisado un maestro proporciona los ejemplos sin las respuestas correctas o etiquetas, sin embargo, el algoritmo intenta encontrar similitudes entre ellos, así los ejemplos similares son agrupados [30].

- El aprendizaje por refuerzo es aprender a tomar decisiones. Qué acción tomar ante situaciones. Es aprendizaje por interacción, similar a la forma que aprenden las personas, prueba-error. Consiste de un agente que interactúa en un ambiente para aprender a lograr un objetivo de la manera más óptima. El aprendizaje basado en ejemplos es inadecuado para aprendizaje por interacción. En problemas interactivos generalmente es impráctico obtener ejemplos para un comportamiento deseado para todas las situaciones, en las cuales el agente pueda actuar. En un ambiente desconocido el agente debería aprender de su propia experiencia [45].

2.2. Aprendizaje por refuerzo

En aprendizaje por refuerzo, un agente toma decisiones en función de una señal del ambiente llamado estado. El estado es cualquier información que es disponible para el agente, puede ser proporcionado por algún sistema de pre-procesamiento. La representación del estado o la información proporcionada para el agente debería ser tan buena para informar al agente, por ejemplo, cuando vemos una escena de una manera muy rápida podemos reconstruir dicha escena detalladamente, o cuando vemos un objeto y enseguida vemos en otro lado, sabemos que el objeto sigue allí. En un nivel artificial un sistema puede medir una posición en dos momentos diferentes y producir un estado que represente información acerca de la velocidad. Lo que queremos, idealmente, es un estado que resuma todas las sensaciones pasadas de una manera compacta, que retenga toda la información relevante. Una representación de estado que retiene toda la información relevante es llamada Markov, o tiene la propiedad de Markov. La propiedad de Markov es importante en aprendizaje por refuerzo porque las decisiones y los valores son asumidos a ser una función sólo del estado actual. [45]

2.3. Procesos de Decisión Markov

Una tarea en aprendizaje por refuerzo que satisface la propiedad de Markov es llamada un *proceso de decisión Markov* (Markov decision process, *MDP*, en inglés). Si los estados (un estado es denotado por s) y las acciones (una acción es denotado por a) son finitos, entonces es llamado *proceso de decisión Markov finito* o *MDP finito*. El MDP es particularmente importante en aprendizaje por refuerzo, es todo lo que se necesita para entender el 90 % del aprendizaje por refuerzo moderno.

Un proceso de decisión Markov consiste de un *agente* que aprende interactuando con su ambiente en un tiempo t = 0, 1, 2,... En cada paso t el agente percibe el estado s del ambiente, $s_t \in S$, donde S es el conjunto de estados del ambiente; y sobre eso elige una acción a primitiva, $a_t \in A_{st}$, donde A_s es el conjunto de acciones en el estado s. En respuesta a cada acción a_t el ambiente produce en el paso $t+1$ una recompensa numérica r_{t+1}, y un estado siguiente s_{t+1}, el cual es s'.

Sea $A = \cup_{s \in S} A_s$ la unión de todos los conjuntos de acciones. Si S y A son finitos, entonces la dinámica de transición del ambiente puede ser modelado por la transición de probabilidad p de un estado a otro en cada paso,

$$p^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\} \; . \tag{2.1}$$

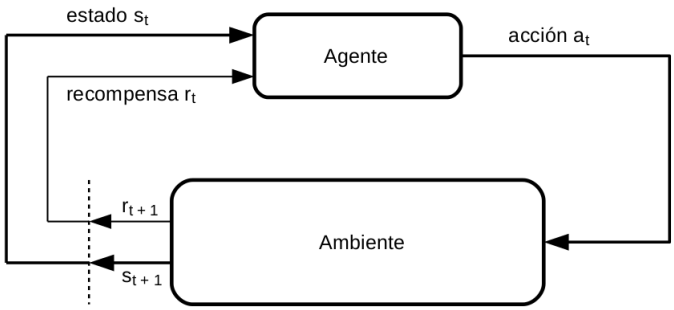


Figura 2.1 Interacción del agente con su ambiente en aprendizaje por refuerzo.

y el valor esperado E de la siguiente recompensa r tambien en cada paso,

$$r_s^a = E\{r_{t+1}|s_t = s, a_t = a\} \ . \tag{2.2}$$

para toda s, $s' \in S$ y $a \in A_s$. Estos dos conjuntos de cantidades constituyen el modelo de paso por paso del ambiente.

El objetivo del agente es aprender una política Markov, esto es una función entre un conjunto de estados y un conjunto de acciones, a cada estado le corresponde una acción de acuerdo a una probabilidad $\pi : S \text{ x } A \to [0,1]$, que maximiza la recompensa en cada estado:

$$V^\pi(s) = E\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ...|s_t = s, \pi\} \ . \tag{2.3}$$

$$= E\{r_{t+1} + \gamma V^\pi(s_{t+1})|s_t = s, \pi\}$$

$$= \sum_{a \in A_s} \pi(s,a) \left[r_s^a + \gamma \sum_{s'} p_{ss'}^a V^\pi(s') \right] \ . \tag{2.4}$$

donde $\pi(s,a)$ es la probabilidad con el cual la política π elige una acción $a \in A_s$ en el estado s y $\gamma \in [0,1]$ es un parámetro de descuento. Esta cantidad, $V^\pi(s)$, es llamada *valor* de estado s bajo la política π y V^π es llamada *función de valor-estado* para π. La función de valor óptimo proporciona el valor de cada estado bajo una política óptima:

$$V^*(s) = \max_\pi V^\pi(s) \tag{2.5}$$

$$= \max_{a \in A_s} E\{r_{t+1} + \gamma V^*(s_{t+1})|s_t = s, a_t = a\}$$

$$= \max_{a \in A_s} \left[r_s^a + \gamma \sum_{s'} p_{ss'}^a V^*(s') \right] \ . \tag{2.6}$$

Una política que alcanza el máximo en la ecuación 2.5 es por definición una política óptima. Así, dado V^*, una política óptima es fácilmente formado por elegir en cada estado s cualquier acción que alcanza el máximo en la ecuación 2.6. Planeación en aprendizaje por refuerzo se refiere a usar modelos de ambientes para calcular funciones de valor, por lo tanto, optimizar o mejorar políticas. Las ecuaciones de *Bellman* son particularmente útiles en este sentido, tales como las ecuaciones 2.4 y 2.6, las cuales recursivamente relacionan funciones de valor. Si tratamos los valores, $V^\pi(s)$ o $V^*(s)$, como desconocidos, entonces un conjunto de ecuaciones Bellman, para toda $s \in S$, forman un sistema de ecuaciones cuya única solución es de hecho V^π o V^* como dado por las ecuaciones 2.3 ó 2.5. Este hecho es clave para la forma en que todos los métodos en diferencia-temporal y programación dinámica estiman funciones de valor.

Hay funciones de valor y ecuaciones Bellman para los pares estado-acción, las cuales son particularmente importantes para métodos de aprendizaje.

El valor de tomar una acción a en el estado s bajo la política π, denotado $Q^\pi(s,a)$, es la recompensa a largo plazo con un factor de descuento empezando en el estado s, tomando a, y por lo tanto siguiendo π:

$$Q^\pi(s,a) = E\{r_{t+1} + \gamma(r_{t+2}) + \gamma^2(r_{t+3}) + ...|s_t = s, a_t = a, \pi\} \tag{2.7}$$

$$= r_s^a + \gamma \sum_{s'} p_{ss'}^a V^\pi(s') \tag{2.8}$$

$$= r_s^a + \gamma \sum_{s'} p_{ss'}^a \sum_{a'} \pi(s',a') Q^\pi(s',a'). \tag{2.9}$$

Este es conocido como la función valor-acción para la política π. La función valor-acción óptimo es:

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) = r_s^a + \gamma \sum_{s'} p_{ss'}^a \max_{a'} Q^*(s',a') \tag{2.10}$$

Finalmente, muchas de las tareas son naturalmente episodios, envolviendo repetidos intentos o episodios, cada uno termina con un reinicio a un estado estándar o un estado de distribución. Las tareas por episodio incluyen un estado terminal especial, llegando en este estado termina el episodio actual. El conjunto de estados regulares más el estado terminal es denotado S^+. Así, el s' en $p_{ss'}^a$ está en el rango general sobre el conjunto S^+ más que sólo en el S. En una tarea por episodio, los valores son definidos por una recompensa acumulada hasta la finalización más que sobre el futuro infinito [38].

2.4. *Options*

Options es una generalización de acciones primitivas para usarlos como acciones que tienen una duración extendida en el tiempo. Por ejemplo, las options recoger objeto y viajar a una ciudad lejana tienen como acciones primitivas movimientos en las articulaciones y activación de los musculos. Formalmente, una option consiste de tres componentes principales: una política $\pi : S \text{ x } A \to [0,1]$, una condición de finalización $\beta : S^+ \to [0,1]$ y un conjunto de inicialización I, $I \subseteq S$.

Una option $\langle I, \pi, \beta \rangle$, está disponible en un estado s_t si y sólo si $s_t \in I$. Si se elige la option entonces las acciones son seleccionadas de acuerdo a una política π hasta que la option termina estocásticamente de acuerdo a una condición de finalización β. En particular, *Markov option* funciona como sigue. Primero se selecciona la acción a_t de acuerdo a una probabilidad de distribución $\pi(s_t, \cdot)$. Como resultado de la acción el ambiente hace una transición al estado s_{t+1}, donde la option termina o continúa de acuerdo a una probabilidad $\beta(s_{t+1})$, determinando a_{t+1} de acuerdo a $\pi(s_{t+1}, \cdot)$, la option posiblemente termine en s_{t+2} de acuerdo a $\beta(s_{t+2})$ y así sucesivamente. Cuando una option termina el agente tiene la oportunidad de elegir otra option.

Por ejemplo, una option llamado *abrir puerta* puede consistir de una política para alcanzar, asir y girar la manija de una puerta, una condición de finalización para reconocer que la puerta ha sido abierta y un conjunto de inicialización que restringe la consideración que una puerta está presente. En tareas por episodio, la finalización de un episodio también termina la option actual [38].

2.5. Procesos de Decisión Semi-Markov

Options está íntimamente relacionado con las acciones y también presente en un tipo especial de problema conocido como *proceso de decisión semi-Markov* (*semi-Markov decision process*, SMDP, en inglés). De hecho, cualquier MDP con un conjunto de options es un SMDP. El siguiente teorema deja más claro este hecho y establece explícitamente las condiciones y consecuencias.

Teorema 1 (MDP + Options = SMDP)**.** *Para cualquier MDP y cualquier conjunto de options definido sobre aquel MDP, el proceso de decisión que selecciona sólo entre estas options, ejecutando a cada una hasta la finalización, es un SMDP.*

Prueba. Un SMDP consiste de: 1) un conjunto de estados, 2) un conjunto de acciones, 3) para cada par de estado y acción hay una recompensa numérica, y 4) una distribución

conjunta bien definida para la transición de un estado a otro. Para nuestro caso el conjunto de estados es S y el conjunto de acciones es el conjunto de options. La recompensa, el estado siguiente y la función de transición son definidos por el estado y la option por el MDP, por la política π y condición de finalización β de la option. Estos valores esperados y distribuciones son bien definidos porque MDPs son Markov, y las options son semi-Markov; así que el estado siguiente, recompensa y el tiempo son dependientes sólo sobre la option y el estado en el cual fue inicializado. Las transiciones en el tiempo son discretos pero sólo en uno de los intervalos reales que son permitidos en SMDPs. □

La forma apropiada del modelo de options en la teoría de SMDP es análoga a r_s^a y $p_{ss'}^a$, definidos recientemente para acciones. Para cada estado en el cual una option puede ser inicializado, este tipo de modelo predice el estado en el cual la option finalizará y el total de recompensa recibida a lo largo del camino. Las recompensas son descontadas en una forma particular. Para cualquier option o, sea $\varepsilon(o,s,t)$ que denota el evento o, lo cual significa inicializar una option o en el estado s en el tiempo t. Entonces la recompensa en el modelo de o para cualquier estado $s \in S$ es

$$r_s^o = E\{r_{t+1} + \gamma r_{t+2} + ... + \gamma^{k-1} r_{t+k} | \varepsilon(o,s,t)\} \tag{2.11}$$

donde $t+k$ es el tiempo aleatorio en el cual o termina. La predicción del estado en el modelo de o para el estado s es

$$p_{ss'}^o = \sum_{k=1}^{\infty} p(s',k)\gamma^k \tag{2.12}$$

para todo $s' \in S$, donde $p(s',k)$ es la probabilidad que la option finalice en s' después de k pasos. Así, $p_{ss'}^o$ es una combinación de la probabilidad donde s' es el estado en el cual o finaliza junto con una medida de retardo γ. Llamamos este modelo *multi-time model* porque describe el resultado de una option en diferentes tiempos, apropiadamente combinados [38].

2.6. Intra-option Q-learning de un sólo paso

Los métodos *Intra-options* son potencialmente más eficientes que SMDP porque ellos extraen más ejemplos de entrenamiento de la misma experiencia.

Por ejemplo, supongamos que se está aprendiendo a aproximar la *función valor-option óptimo* $Q_O^*(s,o)$ y que la option o es Markov. Basado en la ejecución de o, de t a $t+k$, el método SMDP extrae sólo un ejemplo de entrenamiento para $Q_O^*(s,o)$. Como o es Markov es inicializado en cada uno de los pasos entre k y $k+1$. Los brincos de s_i hasta s_{t+k}, son

experiencias válidas con o, experiencias que pueden ser usados para una mejor estimación en $Q_O^*(s_i, o)$. O considerar otra option que es muy similar a o la cual tendría que seleccionar la misma acción pero podría terminar un paso más tarde $t+k+1$ que $t+k$ que la primera option. Formalmente, es una option diferente y no fue ejecutada, sin embargo, toda esta experiencia puede ser usada para aprender lo más relevante. Frecuentemente una option puede aprender algo de experiencia cuando se ejecuta otra option y sólo son ligeramente relacionados porque tienen la misma acción que se está ejecutando. Esta es la idea del entrenamiento *off-policy* hacer uso de cualquier experiencia para aprender tanto como sea posible de todas las options no importando cuál sea su rol para generar experiencia.

Es conveniente introducir una nueva notación para el valor del par estado-option $U_O^*(s,o)$ dado que la option es Markov y es ejecutada hasta llegar al estado.

$$U_O^*(s,o) = (1-\beta(s))Q_O^*(s,o) + \beta(s)\max_{o' \in O} Q_O^*(s,o') \,. \tag{2.13}$$

Entonces se puede escribir la ecuación de Bellman que relaciona $Q_O^*(s,o)$ para esperar valores de $U_O^*(s',o)$, donde s' es el sucesor inmediato de s después de inicializar Markov option $o = \langle I, \pi, \beta \rangle$ en s:

$$\begin{aligned} Q_O^*(s,o) &= \sum_{a \in A_s} \pi(s,a) E\{r + \gamma U_O^*(s',o) | s,a\} \\ &= \sum_{a \in A_s} \pi(s,a)[r_s^a + \sum_{s'} p_{ss'}^a U_O^*(s',o)] \end{aligned} \tag{2.14}$$

donde r es la recompensa inmediata hasta llegar en s'.

Consideremos ahora los métodos de aprendizaje basados en las ecuaciones de Bellman. Suponer que la acción a_t es tomada en el estado s_t para producir el siguiente estado s_{t+1} y una recompensa r_{t+1}. La acción a_t fue seleccionada en forma consistente con la política Markov π de una option $o = \langle I, \pi, \beta \rangle$. Esto significa que a_t fue seleccionada de acuerdo a una distribución $\pi(s_t, \cdot)$. Entonces en la siguiente ecuación de Bellman se aplica Diferencia-Temporal con off-policy en cada paso

$$Q(s_t,o) \leftarrow Q(s_t,o) + \alpha[(r_{t+1} + \gamma U(s_{t+1},o)) - Q(s_t,o)] \,. \tag{2.15}$$

donde

$$U(s,o) = (1-\beta(s))Q(s,o) + \beta(s)\max_{o' \in O} Q(s,o') \,. \tag{2.16}$$

Este método es llamado *One-step Intra-option Q-learning*, la cual aplica esta regla $Q(s_t,o)$ de actualización para toda option o que es consistente con toda acción a_t seleccionada [38].

2.7. Aprendizaje no supervisado

Considere una máquina (o un organismo vivo) que recibe una secuencia de entradas $x_1, x_2, ..., x_t$ donde x_t es una entrada sensorial en el tiempo t. Estas entradas son llamadas *datos*, que podrían corresponder a una imagen sobre la retina, los pixeles en una cámara, o una forma de onda de sonido. Entonces el objetivo del aprendizaje no supervisado es extraer patrones sobre estos datos [19]. Se conocen tres puntos de vista del aprendizaje no supervisado:

- Desde una perspectiva estadística, el aprendizaje no supervisado es visto como un problema de modelación de datos. Así, el campo del aprendizaje no supervisado se ha basado en los métodos estadísticos para la estimación de densidad [18].
- Desde una perspectiva de la teoría de la información, aprendizaje no supervisado a menudo es visto como el problema de maximizar la información contenida en una representación sensorial. El objetivo de un agente es comunicar eficientemente los datos a un receptor, por lo tanto, producir una representación compacta de los datos [18].
- Desde un punto de vista de la estadística física, el ambiente y el agente son sistemas combinados que pueden ocupar muchos estados. Los estados corresponden a patrones de datos sensoriales del ambiente y su representación interna en el agente. Una energía es asociada con cada combinación de estados entre el ambiente y el agente. Esta energía toma un modelo generativo: un estado con baja energía corresponde a combinaciones de datos sensoriales y una representación interna que tienen una alta probabilidad, y viceversa. El objetivo del aprendizaje es encontrar parámetros del sistema que minimice la energía sobre todo el conjunto de datos [18].

Tomando en cuenta estos puntos, en este trabajo nos enfocamos más al problema de aprendizaje no supervisado desde una perspectiva de la teoría de la información.

2.8. Auto-organización, propiedad del cerebro humano

El cerebro humano cumple con la función de hacer representaciones compactas de los datos. Se conoce desde hace mucho tiempo que determinadas áreas del cerebro, especialmente la corteza cerebral, son organizadas de acuerdo a la información recibida de diferentes modalidades sensoriales (visual, auditivo, somatosensorial, etc.). Esta organización es una representación reducida de la información percibida sin perder el conocimiento de sus

interrelaciones y puede ser llevado a cabo por un simple proceso de auto-organización. En un proceso de auto-organización varios mapas son formados y son capaces de describir las relaciones topológicas de la información recibida usando un medio de representación en una o dos dimensiones. Tal mapeo de reducción de dimensiones parece ser una operación fundamental en la formación de abstracciones [25].

La auto-organización ocurre en una variedad de sistemas, pero una manera de ejemplificarlo consideramos una bandada de pájaros que vuelan en formación, cada pájaro no puede conocer su ubicación exacta entonces cómo pueden mantener su formación. Algunas simulaciones demuestran que, si cada pájaro intenta estar detrás y a la derecha de otro, pero en diagonal, y vuela a la misma velocidad entonces ellos pueden hacer perfectas formaciones, no importa cómo ellos empiecen y que obstáculos encuentren en el camino, así se puede lograr un orden global en toda la bandada por medio de interacciones locales de cada pájaro observando el que está a su derecha (o izquierda).

2.9. Mapa Auto-organizado Kohonen

El mapa auto-organizado de Kohonen es un tipo de red neuronal artificial que es entrenado usando aprendizaje no supervisado para producir una representación de baja dimensión (típicamente dos dimensiones) a partir de un conjunto de ejemplos de alta dimensión. El objetivo del aprendizaje es categorizar los ejemplos que se introducen en la red.

La red neuronal descrita en esta sección tiene la propiedad de auto-organización, también es llamada *preservación de mapas topológicos* y asume una estructura topológica entre unidades clústers. Esta propiedad es observada en el cerebro, pero no se encuentra en otra red neuronal artificial. Hay m unidades clústers en un arreglo de dos dimensiones; las señales de entrada son n-tuplas.

El vector de pesos de una unidad clúster sirve como un ejemplar y es asociado con un patrón de entrada. Durante el proceso de auto-organización, el vector de pesos de las unidades clústers hacen una correspondencia al patrón de entrada de acuerdo a una medida de similitud (típicamente es la distancia mínima euclidiana), la unidad clúster más cercano es el ganador. Son actualizados los pesos de la unidad clúster ganador y los clústers vecinos (de acuerdo a una topología). Los pesos de los clústers vecinos normalmente no son cercanos al patrón de entrada. Por ejemplo, en un arreglo de unidades clúster en una dimensión, el vecindario de radio R alrededor de la unidad clúster J consiste de todas las unidades j tales que $max(1, J-R) \leq j \leq min(J+R, m)$.

En las siguientes secciones se presenta la arquitectura y algoritmo que puede ser usado para agrupar un conjunto p de ejemplos de entrada, cada uno dado por el vector con valores continuos $x = (x_1, ..., x_i, ..., x_n)$ hacia m unidades clústers [12].

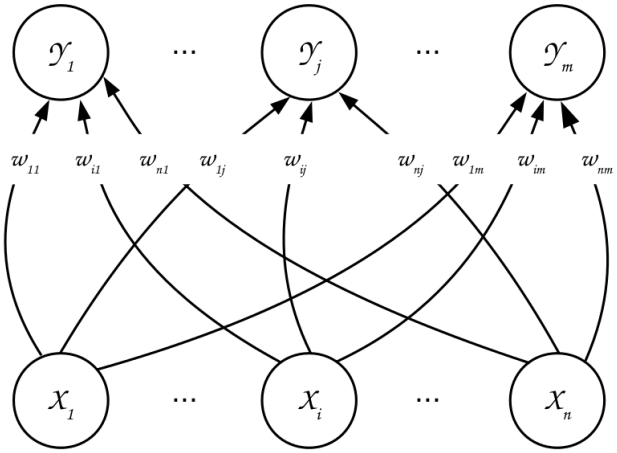


Figura 2.2 Mapa Auto-organizado Kohonen.

2.9.1. Arquitectura

La arquitectura de la red neuronal Mapa Auto-organizado es mostrada en la Fig. 2.2. En la Fig. 2.3 se muestra una topología de una dimensión (con diez clústers "$*$") y los vecinos del clúster designado por "#" con radio R = 2, 1 y 0. En la Fig. 2.4 se muestra el vecindario de radio R = 2, 1 y 0 para una topología rectangular, y en la Fig. 2.5 para una topología hexagonal ambos en dos dimensiones y 49 unidades clúster. En cada ilustración, la unidad ganadora está indicada por el símbolo "#" y los otros son denotados por "$*$". Note que cada unidad tiene ocho vecinos más cercanos en la topología rectangular pero sólo seis en la hexagonal. Las unidades ganadoras más cercanas a la orilla del arreglo tendrán un vecindario con pocas unidades [12].

```
*   *   * { * ( * [ # ] * ) * } *   *
    {} R = 2 () R = 1 [] R = 0
```

Figura 2.3 Topología de una dimensión de unidades clústers.

2.9.2. Algoritmo

Los pasos del algoritmo está descrito en Algoritmo 1.

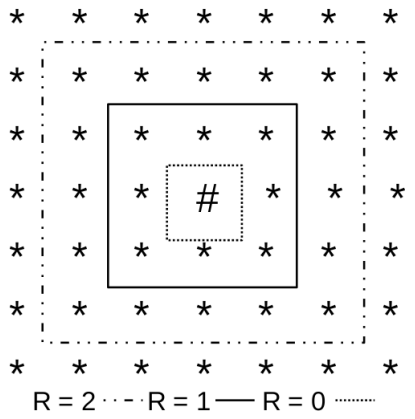


Figura 2.4 Topología rectangular en dos dimensiones.

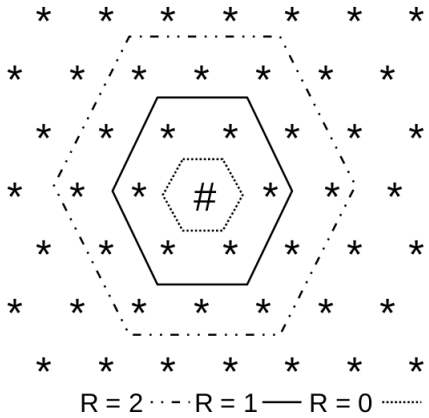


Figura 2.5 Topología hexagonal en dos dimensiones.

La razón de aprendizaje α es una función que decrece lentamente en el tiempo (o épocas de entrenamiento). Kohonen indica que una función que decrece linealmente es satisfactorio para cuestiones de cómputo; un decremento geométrico produciría resultados similares.

El radio del vecindario alrededor de la unidad clúster también decrece conforme al progreso del proceso de clusterización o *clustering*.

La formación de un mapa ocurre en dos fases: la formación inicial de un orden correcto y la convergencia final. La segunda fase se alarga más y requiere un pequeño valor para la razón de aprendizaje. Puede ser necesario muchas iteraciones a través del conjunto de entrenamiento, al menos en algunas aplicaciones, esto lo indica Kohonen.

A los pesos iniciales pueden ser asignados valores aleatorios. Si alguna información está disponible relacionada a la distribución de clúster que puede ser apropiado para un problema en particular, los pesos iniciales pueden ser tomados para reflejar aquel conocimiento previo [12].

Algoritmo 1 Algoritmo Mapa Auto-organizado.

Paso 0. *Inicialización de los pesos w_{ij}*
Fijar los parámetros de vecindario
Fijar el parámetro de aprendizaje
Paso 1. *Mientras la condición de parada sea falsa hacer Pasos 2-8*
Paso 2. Para cada vector de entrada x, hacer Pasos 3-5
Paso 3. Para cada j calcular:

$$D(j) = \sum_i (w_{ij} - x_i)^2.$$

Paso 4. Encontrar índice J tal que $D(J)$ es un mínimo.
Paso 5. Para todas las unidades j en un vecindario específico de J, y para todo i:

$$w_{ij}(new) = w_{ij}(old) + \alpha[x_i - w_{ij}(old)].$$

Paso 6. Actualizar la razón de aprendizaje
Paso 7. Reducir el radio del vecindario a un número específico de veces.
Paso 8. Evaluar la condición de parada.

2.10. Integración Multi-sensorial

Un problema que todos los organismos superiores encaran es cómo integrar información de múltiples modalidades sensoriales. Naturalmente la información llega de diferentes modalidades sensoriales que son de importancia para el comportamiento, tal como la localización de un depredador, la orientación del cuerpo o la identificación lingüística de un enunciado.

La integración de varias modalidades sensoriales ha sido estudiada ampliamente en psicofísica y neurociencia [18].

2.10.1. Psicofísica

Desde un punto de vista psicofísico, las interacciones sensoriales revelan que las percepciones sensoriales a menudo son moduladas por otras modalidades de entrada. Estas interacciones a menudo pueden ser influenciadas por causas secundarias. Por ejemplo, el sistema vestibular del oído, por alterar la orientación percibida del cuerpo respecto la gravedad, puede afectar la percepción de la ubicación del sonido o la orientación de una línea que se muestra visualmente. Revisamos la integración del estímulo visual y auditivo, haciendo enfoque a dos fenómenos clásicos, el "efecto McGurk" y el "efecto ventrílocuo" que ejemplifican muy bien la integración viso-auditiva en la percepción de la voz y la ubicación en el espacio, respectivamente.

El "efecto McGurk". Demuestra la influencia de la visión en la percepción de la voz. Un sujeto escucha unas sílabas pero cuando ve la pronunciación percibe otra sílaba diferente,

entonces pide que le aclaren cuál fue aquella sílaba. La sílaba reportada no fue el que se escuchaba ni el que se veía en la pronunciación, pero fue una entre esas dos. Por ejemplo, cuando sólo escuchamos podemos percibir la sílaba "ba", mientras cuando vemos la pronunciación de la boca percibimos "ga", en realidad se percibiría la sílaba "da".

El "efecto ventrílocuo". Es tal vez la ilusión espacial más estudiada de la integración de los estímulos visual y auditivo. Como su nombre lo indica, el efecto surge cuando la percepción de la ubicación del sonido cambia a un estímulo visual que no es donde se origina. Como muchas ilusiones, este efecto refleja algunas propiedades básicas del sistema perceptivo. Una de las propiedades es el dominio de la visión en nuestro campo de percepción.

2.10.2. Neurociencia

La integración de información de múltiples modalidades sensoriales y la plasticidad en la relación entre los sentidos plantea problemas interesantes a la neurociencia. En varias áreas del sistema nervioso central se lleva a cabo la convergencia visual, auditiva e información prioceptiva tales áreas son la formación reticular, la corteza visual, ganglios basales, varias regiones del cerebelo, algunos núcleos del tálamo y el culículo superior. El culículo superior es de las áreas de integración multisensorial y sensoriomotor mejores estudiadas del sistema nervioso.

El culículo superior. El culículo superior en mamíferos y su homólogo, techo óptico, en otros vertebrados, es una estructura del mesencéfalo implicado en la atención y la orientación. El culículo superior está compuesto por siete capas de células divididos en dos partes: las capas superficiales (I-III) y las capas profundas (IV-VII). Las capas superficiales reciben las señales visuales que vienen directamente de la retina y de la corteza visual. Las capas profundas reciben señales visuales, somatosensoriales, auditivas, entre otras. Más del 50 % de las neuronas de la capa profunda son multisensoriales, la combinación más común es visual-auditiva, un 30 % del total. Es importante notar que la convergencia multisensorial parece tener lugar en la capa profunda. Las salidas del culículo superior proyectan al tronco cerebral y la medula espinal que son áreas directamente involucrados en posicionar los órganos sensoriales periféricos. Aunque comúnmente es considerado parte del sistema de control del movimiento del ojo, el culículo superior juega un papel importante en los movimientos para la orientación de la cabeza, extremidades; en algunas especies, el movimiento de las orejas y los bigotes.

2.10.3. Ventajas de la integración multisensorial

Desde una perspectiva computacional del sistema sensoriomotor humano, es natural preguntar cuales son las ventajas de diseñar un sistema multisensorial. El estudio de la robótica sugiere tres principales ventajas de combinar múltiples fuentes de información.

- Múltiples sensores proporcionan *redundancia*, los cuales pueden reducir incertidumbre en la estimación sensorial e incrementar la fiabilidad en el caso de fallos sensoriales.
- Se puede conseguir *información complementaria* de diferentes sentidos. La integración de información a través de varios sentidos puede proporcionar información que es imposible obtener usando sólo uno.
- Más *información oportuna* puede obtenerse a través del paralelismo, por la latencia que puede tener cada sentido.

Aunque los tres factores pueden ser muy importantes para sistemas multisensoriales de organismos biológicos, nos enfocamos en el primer factor [18].

2.11. Modelación de la auto-organización por el algoritmo de redes neuronales mapa auto-organizado

La integración multisensorial (Multisensory Integration, MSI, en inglés) es una capacidad que tiene los humanos y otros animales de integrar señales de varios sentidos (visual, auditivo, tactil, etc.) que vienen del mundo exterior para localizar objetos con precisión Fig. 2.6, esta capacidad puede ser modelado por el algoritmo de redes neuronales mapa auto-organizado.

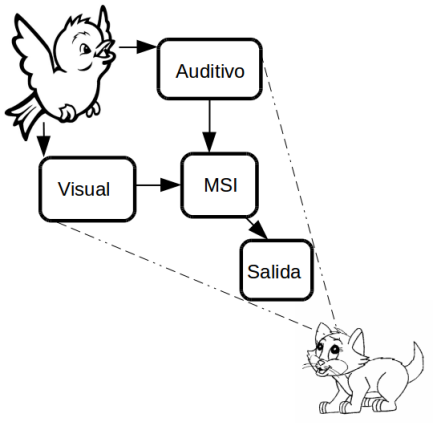


Figura 2.6 Integración multisensorial.

El culículo superior (CS) está en el área medio del cerebro el cual recibe varios tipos de señales y los integra para localizar un origen común de cada señal de entrada. El CS utiliza esta información integrada para generar órdenes motoras, por ejemplo, escuchar el canto de un pájaro y percibir sus movimientos en la periferia del campo de visión, puede guiar, por ejemplo, el enfoque de los ojos, es decir, movimientos rápidos que los dirigen hacia el origen del estímulo [3].

El problema de integración de diferentes señales, puede ser descrito tomando como ejemplo la visión y la audición: considerar que el estímulo audio-visual esta en frente, la visión proporcionará una estimación muy exacta de la ubicación del objeto mientras que la audición puede estar un poco alejado de la estimación exacta. Si ambas señales serán integradas -y deberían ser, porque la audición proporciona información valiosa- entonces necesitarán tener diferentes pesos. La fiabilidad de los diferentes tipos de señales no es igual en todo el espacio, por ejemplo, la precisión de la visión es mucho más grande en el centro del campo visual que en la periferia [3].

2.12. Simulación de datos

Hay muchos tipos de simulación. Los científicos del clima usan la simulación para modelar la interacción entre la superficie terrestre, los oceános y la atmosfera. Los astrofísicos usan la simulación para modelar la evolución de las galaxias. Los biólogos para modelar el desarrollo de una epidemia y los efectos de los programas de vacunación. Los ingenieros lo usan para estudiar la seguridad y la eficiencia del combustible de un automóvil y el diseño de un avión. En estas simulaciones de los sistemas físicos los científicos modelan la realidad y usan una computadora para estudiar el modelo bajo varias condiciones.

La simulación nos da un completo control sobre las características de un modelo de población, del cual son extraídos los datos simulados. Simular datos con características determinadas significa tener un entendimiento de aquellas características. Aplicar métodos estadísticos para simular datos nos ayuda a un mejor entendimiento de los métodos y principios subyacentes.

La *simulación de datos* significa generar una muestra aleatoria de una distribución con propiedades conocidas.

Simulación de datos con distribución discreta. Cuando un conjunto de posibles muestras es finito o contablemente infinito (como los enteros), asignando una probabilidad a cada muestra crea una distribución de probabilidad discreta y la suma de todas las probabilidades es la unidad, algunos métodos son: la distribución Bernoulli, Binomial, Geométrico, Uniforme, Tabulado y Poisson.

Simulación de datos con distribución continua. Cuando el conjunto de muestras posibles son infinitos (como un intervalo o un conjunto de números reales), asignando una probabilidad a cada muestra, y por supuesto la suma de todas las probabilidades es uno, crea una distribución de probabilidad continua, algunos métodos comunes son: la distribución normal, uniforme y exponencial.

Para trabajar con simulaciones estadísticas es esencial saber cómo calcular densidad de probabilidad, densidad acumulada, cuantiles y valores aleatorios para distribuciones estadísticas, sin embargo, en este trabajo sólo nos enfocamos a la función de densidad probabilística, el cual definimos a continuación [47].

2.12.1. Función de Densidad Probabilística

Una función de densidad probabilística (FDP) devuelve la densidad de probabilidad de un valor específico.

Para muchas variables aleatorias continuas, podemos definir una función extremadamente útil para calcular probabilidades de eventos asociados a la variable aleatoria.

Definición 2.1. *Sea $F(x)$ una función de distribución para una variable aleatoria continua X. La **función de densidad de probabilidad** para X es dado por*

$$f(x) = \frac{dF(x)}{dx} \tag{2.17}$$

donde la derivada existe [31].

En otras palabras, la FDP de una variable continua aleatoria es la derivada de su **función de densidad acumulada** (FDA). Por el Teorema Fundamental del Cálculo sabemos que el FDA de $F(x)$ de una variable aleatoria continua X puede ser expresada en términos de su FDP:

$$F(x) = \int_{-\infty}^{x} f(t)\,dt. \tag{2.18}$$

donde f denota la función de densidad probabilística de X.

La formulación de la FDP por el Teorema Fundamental del Cálculo nos permite derivar las siguientes propiedades:

Propiedades 2.2. *Si $f(x)$ es una función de densidad de probabilidad para una variable aleatoria continua X entonces*

2.2a $F(b) = Pr(X \leq x) = \int_{-\infty}^{x} f(t)\,dt.$

2.2b $f(x) \geq 0$ *para cualquier valor de x.*

2.2c $\int_{-\infty}^{\infty} f(t)\,dt = 1.$

La propiedad 2.2a como hemos visto, es sólo una aplicación del Teorema Fundamental del Cálculo.

La propiedad 2.2b establece que para que una función sea FDP, debería ser positiva. Esto tiene sentido dado que las probabilidades son siempre números positivos. Más precisamente, sabemos que la FDA de $F(x)$ es una función no-decreciente de x. Así, su derivada $f(x)$ es positiva.

La propiedad 2.2c establece que el área entre la función y el eje x debería ser 1, o que todas las probabilidades deberían sumar 1, esto debería ser verdadero dado que

$$\lim_{x \to -\infty} f(x) = 0$$

y

$$\lim_{x \to \infty} f(x) = 1.$$

La FDP nos da una interpretación geométrica muy útil de la probabilidad de un evento: la probabilidad que una variable aleatoria continua X es menos que algún valor x_0, es igual al área bajo la FDP $f(x_0)$, sobre el intervalo (-∞, x_0], como muestra la siguiente gráfica.

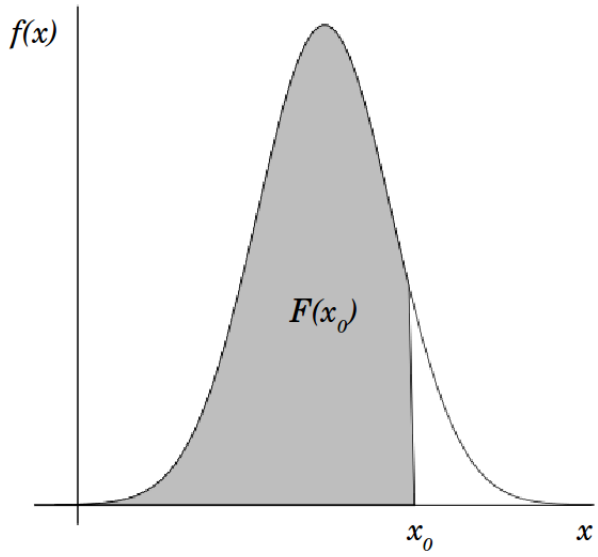


Figura 2.7 Función de densidad probabilística.

De otra manera dado un intervalo [a,b], la probabilidad que esté una variable aleatoria continua X es:

$$Pr(a \leq X \leq b) = \int_a^b f(x)\,dx$$

2.12.2. Distribución Multivariada

Sólo unos pocos métodos de distribución univariada estándar tienen extensiones a métodos multivariados, sin embargo, la simulación de datos con métodos multivariados es una técnica muy importante ya que es muy útil si se tiene variables con una estructura de correlación prescrita. De otra manera, si se tiene variables que no tienen correlación entre sí se puede usar los métodos de distribución univariada.

Hay varios métodos multivariados entre ellos la distribución multinomial, la distribución normal y sus versiones, entre otras. En este trabajo nos enfocamos al método de simulación de datos de distribución normal multivariado, específicamente se usan sólo dos variables y no están correlacionados, se podría usar los métodos univariados para generar los datos, sin embargo, este trabajo mira más allá y en un trabajo futuro se podría hacer simulaciones con datos correlacionados [47].

Distribución Normal Bivariada

Es una distribución estadística con función de densidad probabilística

$$p(x_1,x_2;\xi_1,\xi_2,\sigma_1,\sigma_2,\rho) = \left[2\pi\sqrt{1-\rho^2}\right]^{-1} \exp\left[-\frac{1}{2(1-\rho^2)}\left\{\left(\frac{x_1-\xi_1}{\sigma_1}\right)^2 - 2\rho\left(\frac{x_1-\xi_1}{\sigma_1}\right)\left(\frac{x_2-\xi_2}{\sigma_2}\right) + \left(\frac{x_2-\xi_2}{\sigma_2}\right)^2\right\}\right], \tag{2.19}$$

donde $E[X_j] = \xi_j$, var $(X_j) = \sigma_j^2$ $(j = 1,2)$, y la correlación entre X_1 y X_2 es ρ. En este caso la matriz de varianza-covarianza sería

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}.$$

Esta es la *distribución normal bivariada*.

Para muchos propósitos, es suficiente estudiar la distribución estandarizada. Esta se obtiene haciendo $\xi_1 = \xi_2 = 0$ y $\sigma_1 = \sigma_2 = 1$ en la función 2.19.

$$p(x_1,x_2;\rho) = \left[2\pi\sqrt{1-\rho^2}\right]^{-1} \exp\left\{-\frac{1}{2(1-\rho^2)}(x_1^2 - 2\rho x_1 x_2 + x_2^2)\right\} \tag{2.20}$$

Si en la función 2.19 se tiene $\sigma_1 = \sigma_2$ y $\rho = 0$ se llama función de densidad *normal circular*, Fig. 2.8. No debe confundirse con la distribución normal circular univariada. Si $\rho = 0$ y $\sigma_1 \neq \sigma_2$ entonces se llama *normal elíptico*, Fig. 2.9 [39].

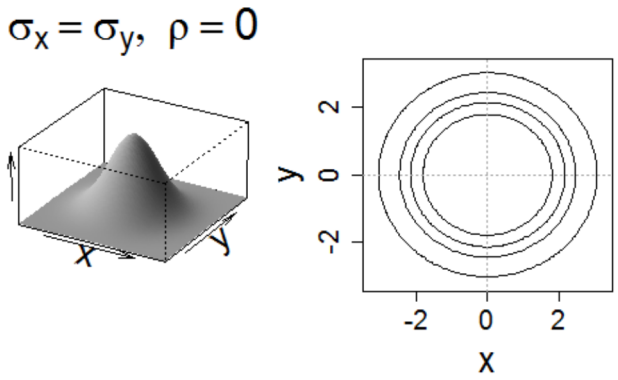


Figura 2.8 Función de densidad normal circular.

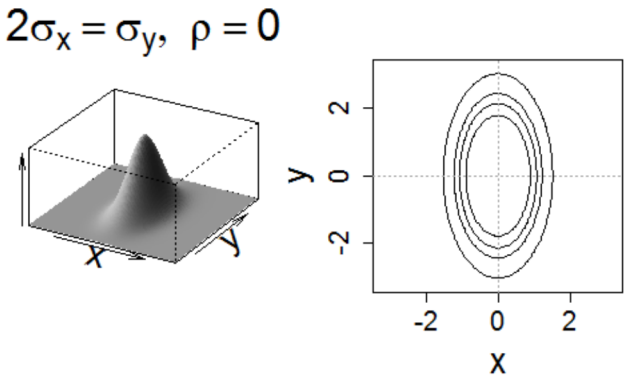


Figura 2.9 Función de densidad normal elíptico.

2.12.3. Algunas consideraciones para la simulación de datos

Eficiencia y Eficacia

Una simulación eficiente es aquella que optimiza los recursos computacionales para terminar en un tiempo razonable. Una simulación eficaz es aquella que produce los resultados estadísticos deseados. La eficiencia es alcanzada a través de una buena práctica de programación. La eficacia es alcanzada a través de un sólido entendimiento de los conceptos estadísticos y métodos. El diseño de un estudio de simulación afecta su eficiencia y su eficacia. Uno de los primeros parámetros a decidir es el número de ejemplos para usar en una simulación. Si se usa muy pocos ejemplos el programa corre demasiado rápido, pero la incertidumbre en las estimaciones hace que la simulación sea inútil. Si se usa demasiados ejemplos, el error estándar de las estimaciones es muy pequeño pero el programa requerirá horas o días para terminar. Claramente, la mejor elección está entre estos dos extremos.

El efecto del número de ejemplos

La clave para lograr un mejor resultado en un estudio de simulación son los experimentos computacionales. Los principios de diseño estadístico pueden ser usados para mejorar la eficiencia, reproducibilidad y documentación del estudio de simulación.

El mejor consejo es empezar con una simulación pequeña y simple. Se debería usar un número pequeño de muestras que el número que se intenta usar en el estudio final.

Generalmente, en un estudio de investigación se maneja diez mil o más ejemplos. Sin embargo, no hay que ser esclavo de algún número en particular. El mejor enfoque es entender qué se está intentando estimar y reportar no sólo puntos estimados si no también errores estándares y/o intervalos de confianza.

Desventajas de la simulación

La simulación es una técnica poderosa, pero tiene limitaciones, los cuales son la dificultad en la generalización de los resultados, organizar los datos y aplicar los resultados en datos reales. Una crítica común de los estudios de simulación es que no puede ser generalizado más allá de los modelos de población y parámetros utilizados en el estudio. Además, los datos reales son raramente distribuidos de acuerdo a cualquier distribución que pueda ser nombrado, por tanto, cómo generalizaremos a los datos reales.

Una simulación puede justificarse para una determinada distribución, pero no para otras, así que una simulación proporciona evidencia que algo es verdadero, pero se necesita tener cuidado no hacer una extrapolación más allá de los casos que fueron simulados. Podríamos simular datos con distribuciones no normales, tales como exponencial, gamma y beta, sin embargo, manejar estas distribuciones al mismo tiempo complica presentar los resultados de una manera ordenada. A pesar de sus limitaciones, la simulación sigue siendo una poderosa técnica que está en la caja de herramientas de los programadores estadísticos. La simulación de datos nos da la capacidad de comparar dos o más técnicas de simulación para estudiar su rendimiento en un mismo conjunto de datos con propiedades conocidas [47].

2.13. Ecolocalización

Los murciélagos tienen un oído muy sensible para enfrentar sus actividades nocturnas. El rango de sensibilidad varía por especie; el más bajo puede ser 1kHz para algunas especies y para otras el más alto alcanza hasta 200kHz. Los murciélagos que han detectado hasta 200 kHz no pueden escuchar bien bajo los 10 kHz. En cualquier caso, el rango más sensitivo de los murciélagos es más estrecho: entre los 15 kHz y 90kHz.

Los murciélagos navegan alrededor de los objetos y ubican sus presas usando la ecolocalización que consiste en que el murciélago produce un sonido corto y fuerte, para luego evaluar el eco que rebota de algún objeto. Los murciélagos cazan insectos voladores; estos insectos

devuelven un débil eco del sonido. El tipo de insecto y su tamaño puede ser determinado por la calidad del eco y el tiempo que toma para rebotar, hay dos tipos: la frecuencia constante (constant frequency, CF, en inglés) es usado para detectar un objeto y la frecuencia modulada (frequency modulated, FM, en inglés) es usado para evaluar su distancia. FM y CF son dos diferentes tipos de eco que informa al murciélago sobre el tamaño y la distancia de la presa, respectivamente. Los sonidos producidos por el murciélago tardan sólo unas pocas milésimas de segundos; los silencios entre las llamadas dan tiempo para escuchar la información de regreso en una forma de un eco. Evidencias sugieren que los murciélagos usan el cambio de pitch (se puede entender como un sonido base), producido por el efecto Doppler, para evaluar su velocidad de vuelo en relación a objetos alrededor. La información respecto a tamaño, forma y textura es construída para formar una imagen de sus alrededores y la ubicación de su presa. Usando estos factores un murciélago puede exitosamente seguir los cambios en los movimientos y por lo tanto cazar su presa.

Los murciélagos generalmente son muy buenos para ubicar objetos con la ecolocalización incluso se ha pensado que los murciélagos son ciegos porque han volado exitosamente en el aire sin la vista, sin embargo, es todo lo contrario, aunque hay dos grupos de murciélagos los cuales son los microquirópteros y los megaquirópteros, que tienen una visión dicromática (en blanco y negro) y una visión en colores, respectivamente [41].

En el caso de los microquirópteros la retina está compuesto principalmente de células que sólo pueden percibir cuando la luz del ambiente es muy baja y estos son estrictamente nocturnos [41].

Aunque la visión de los murciélagos no es tan buena como la de un roedor aun así es muy importante, ya que les sirve para la orientación mientras vuelan, para evitar predadores y para cazar presas muy grandes en los cuales la ecolocalización a veces no es muy útil [41] [33].

Otros animales que pueden escuchar el ultrasonido son los perros y las ballenas dentadas, incluyendo los delfines, éstos últimos usan el ultrasonido para orientarse y capturar sus presas [7].

2.14. Validación cruzada

La validación cruzada consiste en no utilizar todo el conjunto de datos para el entrenamiento del algoritmo. Algunos de los datos es removido antes de empezar con el proceso de entrenamiento. Entonces cuando el algoritmo ya está entrenado los datos que fueron removidos pueden ser usados para probar el rendimiento del modelo aprendido. A esto se le llama el método de evaluación de validación cruzada.

Hay varios tipos de validación cruzada entre ellos está *K-fold cross validation*, Fig. 2.10, este consiste de un conjunto de datos que es dividido en k subconjuntos. Cada vez, uno de los k subconjuntos es usado como el conjunto de prueba y los otros k-1 subconjuntos son usados como el conjunto de entrenamiento. Entonces es calculado el error medio a través de todos los k procesos de entrenamiento y prueba. Una de las ventajas de este método es que no hay problema en la manera en que los datos son divididos. Todos los ejemplos son probados exactamente una vez, y ser usados como ejemplos de entrenamiento k-1 veces. La varianza de la estimación es reducida conforme k incrementa. Una de las desventajas del método es que el entrenamiento del algoritmo tiene que ser ejecutado k veces, eso significa también hacer k veces el cómputo para la evaluación [40].

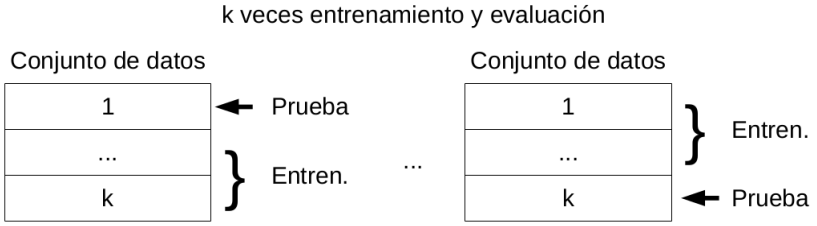


Figura 2.10 Validación cruzada *K-fold*.

Capítulo 3

Estado del arte

3.1. Autonomía

Un agente no necesariamente es un programa de computadora, puede ser un robot, un maestro que enseña en una escuela o incluso una bacteria biológica. Los agentes de software son por definición programas de computadoras, pero no todos los programas son agentes, un programa debe cumplir ciertos requisitos para ser un agente [16].

Un agente es un sistema capaz de actuar de manera autónoma en un ambiente dinámico e impredecible, por lo tanto, debería ser capaz de percibir el ambiente, razonarlo y actuar sobre ella por sí mismo. Los agentes pueden ser solamente sistemas de software puro, en los cuales sus percepciones y acciones son cadenas binarias codificadas. Cuando los agentes son implementados en hardware y son colocados en el mundo real se vuelven robots [17].

La necesidad de la autonomía en un agente es para su supervivencia o evitar peligros en un ambiente que está en constante cambio, ya sea en el mundo real o en un ambiente de bits. Un claro ejemplo son los robots Rover de exploración que son enviados a Marte, aunque no son completamente autónomos tienen cierto nivel de autonomía para llevar a cabo sus actividades en la superficie marciana con tan sólo recibir unos cuantos comandos de instrucción al día desde la tierra. Es imposible tener los Rovers teleoperados en tiempo real debido a que un haz de luz le lleva 20 minutos en promedio viajar desde la Tierra a Marte, por lo tanto la comunicación sería muy retardada. La autonomía es de mucha importancia para la NASA y otras instituciones que cooperan con ella, en un futuro próximo alrededor del 2022, se espera lanzar un Rover capaz de colectar muestras de la superficie de marte y regresarlas a la tierra, para el proceso de colección de muestras es necesario un nivel de autonomía sin precedentes [2].

Similarmente, en los softbots la autonomía tampoco deja de ser importante para enfrentar ambientes de software muy complejos tales como una estructura de archivos de un sistema

operativo, una base de datos o internet. Tales ambientes son muy difíciles para un usuario humano administrarlos de manera eficiente por la basta información que contienen. Por ejemplo, Sumpy (Song, Franklin y Negatu, 1996) es un agente inteligente autónomo en el sistema operativo UNIX que ayuda a administrar el espacio en disco duro haciendo las tareas de compresión y respaldo de archivos, entre otras [43]. MARVIN (Multi-Agent Retrieval Vagabond on Information Networks, en inglés) es un softbot multiagente diseñado por el Hospital Universtario de Ginebra, Suiza, desde 1996. Es utilizado para recuperar e indexar sólo información médica multilingüe de internet, reduciendo a un espacio de búsqueda sólo sobre temas de medicina. También ha sido aplicado en otros campos tales como la biología molecular [4].

De acuerdo con (Stan Franklin y Art Graesser, 1996) y (Frank C. Keil, 1992), la clasificación natural de los agentes (no incluido los biológicos) se divide entre los agentes de software (softbots) y los agentes que viven en el mundo real (robots) [16], [24].

3.1.1. Agentes de software

Los agentes de software o softbots, tal como los robots que están en el mundo real, también tienen la capacidad de percibir, razonar (hasta cierto punto) y ejecutar acciones en su ambiente, es decir, realizar tareas con algún grado de autonomía. Tomamos como ejemplo los agentes de información (uno de los tipos de clasificación de los agentes de software) para clarificar cómo funcionan estos agentes de software debido a que no tienen efectores y sensores físicos como los robots [42].

Los agentes de información han surgido debido a la gran demanda de herramientas que ayudan a manejar el crecimiento explosivo de la información que estamos experimentando actualmente y que seguiremos experimentando en la internet. Cabe recalcar que hay otros tipos de agentes como los colaborativos o de interface que también pueden ser considerados como agentes de información pero éstos surgieron bajo diferentes condiciones. Los agentes de información que mencionamos aquí surgen debido a una explosión de información en la internet en diferente grado. Cumplen con las tareas de gestionar, manipular o recopilar información de muchas fuentes distribuidas [34].

Un agente de información típicamente trabaja en conjunto con un navegador, usa una gran cantidad de herramientas de internet tales como spiders y motores de búsqueda con el fin de recopilar información. Un Spider es un indexador capaz de hacer búsquedas en la internet de manera profunda y almacenar las topologías de la www en una DBMS (database management system, en inglés), además, el índice completo de URL's lo almacena en el WAIS, el cual permite a los usuarios descubrir la información y resuelve el acceso a la red

sin tener en cuenta su ubicación física. Otros motores de búsqueda/indexación o spiders tales como Lycos or Webcrawler pueden ser usados similarmente para crear indexaciones [34].

A continuación, se presenta un ejemplo de prototipo de un softbot (agente de información) de Etzioni y Weld, 1994, el cual su ambiente es la internet. Es un agente que permite al usuario hacer consultas de alto nivel. Es capaz de usar búsquedas e inferencia para determinar cómo satisfacer la consulta en internet. Al hacerlo, es capaz de tolerar ambigüedades, omisiones y los inevitables errores del usuario en la consulta. En su artículo, Etzioni y Weld usan una fuerte analogía a un robot real conforme describe su softbot de interface para internet. Por ejemplo, describe los efectores del softbot incluyendo *ftp*, *telnet*, *mail* y otros comandos de manipulación de archivos incluyendo *mv* o *compress*. Los sensores proporcionan información al softbot acerca del mundo exterior y ellos son las herramientas para percibir internet tales como *archi*, *gopher* y *netfind* entre otros comandos de Unix tales como *mv* y *compress*; netfind, por ejemplo, se usa para determinar la dirección de correo de los usuarios [34].

Los agentes de software se pueden dividir de muy diversas formas, se mencionan aquí sólo algunas.

1. Agentes de información
   - Jasper
2. Agentes de entretenimiento
   - Julia
3. Agentes según el ambiente de interacción
   a) Sistema de archivos en un sistema operativo
   - Sumpy

   b) Internet
   - MARVIN
4. Agentes de interface
   - Letizia
5. Agentes de aprendizaje automático
   - Magi
   - UNA

6. Asistente personal inteligente
    - Siri
    - Cortana
    - Google Now

Jasper

Jasper (Joint Access to Stored Pages with Easy Retrieval, en inglés), (John Davies, et al., 1995) es un agente inteligente que cae dentro de la clasificación de los agentes de información por sus características y bajo las condiciones por las que surge, básicamente por que realiza tareas a través de internet para satisfacer una consulta hecha por uno o varios usuarios, estas tareas son almacenamiento, recuperación y resumen de información, además de informar a otros agentes acerca de la información encontrada. Este agente también tiene la capacidad de mejorar su desempeño entre ellos buscar la retroalimentación de parte de los usuarios.

El enfoque de este agente es muy diferente a otros motores de búsqueda, porque después de encontrar información útil en la internet la almacena de una forma especial para su fácil recuperación y también la de informar a otros usuarios que pueden estar interesados.

En lugar de copiar toda información útil encontrada, los agentes Jasper sólo almacenan información relevante que pueden ser palabras clave, un resumen, título del documento, URL y fecha y hora de acceso. Esta meta-información se utiliza para indexación de información y sobre la cual se realiza una solicitud de recuperación en otro momento.

Los agentes de Jasper también tienen la capacidad de aprender los intereses de los usuarios observando su comportamiento. A medida que el usuario almacena más páginas, su perfil es modificado por Jasper para reflejar mejor los temas de su interés.

El fin de Jasper es permitir al usuario alcanzar un conjunto más amplio de información de su interés mediante la meta-información.

Muchas de las herramientas de información de Internet tienen las cuatro funciones principales que son búsqueda, almacenamiento, acceso y organización. El agente inteligente Jasper tiene otras dos novedosas funcionalidades: en primer lugar, el intercambio automatizado de información entre agentes; en segundo lugar, la capacidad del agente para modelar los intereses del usuario basándose en su interacción con el sistema. Esta funcionalidad no es común en ninguna otra herramienta de Internet [11].

Julia

Julia (Mauldin 1994) es un softbot que se conecta en un mundo virtual llamado MUD (Multi-User Domain, en inglés), el cual es un ambiente de realidad virtual multiusuario

donde la comunicación es de sólo texto, comenzaron como juegos, pero su uso ha sido reemplazado por ambientes más interactivos. Está diseñado para hacerse pasar como un humano (originalmente desarrollado para pasar la prueba de Turing) en el ambiente de interacción restringido MUD, cumpliendo con las tareas de entretener e informar, además suele provocar reacciones emocionales intensas en aquellos con los que interactúa. Estas reacciones suelen provocarse porque muchas de las respuestas de Julia son deliberadamente humorísticas.

Julia (no se ejecuta en el mismo servidor que el MUD) se conecta al MUD como cualquier jugador por medio de telnet. Cuando ella está conectada e interactúa con los jugadores en este tipo de ambiente es muy difícil darse cuenta que se trata de un softbot, de esta manera Julia cumple con sus objetivos de pasarse a ser como un humano, entretener e informar de manera efectiva.

Uno de los comportamientos muy útiles de Julia es hacer mapas de las interconexiones entre cuartos (los cuartos son objetos muy importantes en cualquier MUD), con esto ella puede mantener una estadística acerca de aquellos cuartos y puede responder preguntas importantes para los jugadores tales como cuántos cuartos hay, cuántos jugadores están en los cuartos, cuáles cuartos son más ruidosos debido a la conversación, o una sugerencia de parte de Julia para visitar un cuarto, también puede responder preguntas que normalmente serían muy difíciles con herramientas estándares, como la última ubicación o último momento de conexión de los jugadores, Julia tiene la capacidad de entender quién es ella y donde está ubicada.

Además, tiene otras capacidades, incluyendo el adivinar el género de otros jugadores, basado en sus nombres, también puede cumplir con otros roles como la de entregar mensajes (si consideramos que un MUD no tiene un sistema de correos), en este caso Julia resuelve el problema de comunicarse con otra persona cuando no está conectada.

Estructura general de Julia

Julia es dividido en varias subsecciones:

- Codigo para interactuar con el ambiente MUD por sí mismo.
- El analizador, el cual hace coincidir las declaraciones entrantes de los jugadores y elige una respuesta adecuada.
- La base de datos que consiste de mapas de interconexión de cuartos, comentarios de jugadores, descripciones de sí mismos, etc.
- El código para realizar ciertas actividades autónomas, tales como la exploración del MUD para construir mapas.

Acciones autónomas

La acción autónoma más importante de Julia está relacionada en construir mapas del ambiente MUD. Necesita este mapa para navegar ya que ella nunca se teletransporta, para Julia este mapa es muy importante para responder preguntas que hacen los jugadores en relación a ubicaciones de otros.

El proceso de construcción de mapas se vuelve relativamente simple en que cada cuarto es representado con un nombre, una descripción y su conjunto de salidas, los cuales apuntan a nombres de otros cuartos. Julia tiene un código de caso especial para analizar las descripciones de las habitaciones, dado que ellos pueden ser ambiguos. Ella también observará los números de cuartos y preferirá los que están pintados por el MUD ya que son únicos. Si aun con estos datos falla la construcción del mapa, utilizará una variedad de heurísticas para extraer los nombres de las habitaciones de sus descripciones, y también tiene algunos casos especiales de código para ciertos casos problemáticos en algunos MUD en específico.

Cuando Julia está mapeando cuartos, ella da prioridad para explorar salidas que no han sido exploradas, también intentará el 1 % de las veces pasar por una salida previamente explorada para capturar modificaciones hechas por los jugadores, dado que los jugadores pueden inesperadamente agregar una salida a un cuarto previamente explorada, Julia toma cuidado para encontrar todas las salidas de cualquier cuarto esto lo hace por sí misma.

En cualquier momento, alguien puede preguntar a Julia la ruta de un cuarto a otro entonces ella usa el estado actual de su base de datos para calcular la ruta más corta usando un algoritmo muy conocido de recorrido gráfico (graph-traversal algorithm, en inglés). Un MUD típico tiene miles de cuartos y todavía decenas de miles de salidas, pero hacer un cálculo de un camino más corto es computacionalmente barato [10], [15].

Letizia

Letizia es un agente autónomo de interface que realiza sugerencias al usuario en tiempo real acerca de sitios Web que podrían ser de su interés.

La Web es un ambiente común para agentes autónomos de interface. Los usuarios de la web sienten la necesidad de algún tipo de asistencia inteligente, dado que la manipulación directa en la interface siguiendo links en un navegador causa la sensación de ahogarse en información irrelevante.

Letizia es un agente autónomo de interface que hace búsquedas a través de la web de manera continua y cooperativa entre el usuario y el navegador. El agente almacena las URLs elegidas por el usuario y hace una lectura a las páginas para compilar un perfil de acuerdo a los intereses del usuario. Para analizar las páginas se usa una simple medición de la frecuencia de palabras clave. Letizia está siempre activo, buscando en la Web lo que está cercano a lo

que esta buscando el usuario, sus búsquedas siempre están en paralelo con las del usuario. Letizia usa la interface de Netscape para presentar sus resultados, utilizando una ventana independiente en la que el agente navega por páginas que podrían interesar al usuario [28].

Magi

Los siguientes dos agentes Magi y Una han sido probados con dos algoritmos de aprendizaje automático los cuales son el algoritmo de inducción de reglas CN2 y el algoritmo k-vecinos más cercanos. CN2 es un algoritmo de aprendizaje supervisado que construye reglas a partir de un conjunto de ejemplos preclasificados. El algoritmo k-vecinos más cercanos realizan una clasificación comparando una instancia nueva con instancias previamente clasificadas.

Magi ayuda al usuario a clasificar los correos que le llegan. En esencia, el sistema es un aprendiz que autónomamente observa y analiza el comportamiento del usuario al tratar con el correo, por ejemplo, puede ser alguna versión modificada de *Xmail*. Para cada sesión se crea un logfile, el cual contiene las acciones del usuario y los mensajes en los que se realizaron las acciones. Periódicamente, son extraídas las características de estos mensajes en el logfile para generar el perfil del usuario. Las características extraídas de los correos entrantes son probados por el motor de clasificación, y se genera un índice de confianza. La predicción es considerada válida si su nivel de confianza es más grande que un umbral llamado umbral predictivo. Las predicciones válidas son almacenadas por el agente para su presentación al usuario.

Se informa al usuario si han llegado nuevos mensajes cuando utilice la aplicación. En este punto, el usuario puede instruir al agente para que realice las acciones sugeridas o pueda elegir entre las acciones predichas. Para cada tipo de acción que se puede predecir, existe un umbral de confianza. Sólo se invocarán acciones con calificaciones superiores a esta. La razón detrás de esto es que ciertas acciones predichas, si son incorrectas, pueden ser toleradas, como almacenar un mensaje en el buzón equivocado. Sin embargo, las acciones como las que eliminan correo o envían mensajes a otros destinatarios son más críticas. Por lo tanto, el agente requiere un mayor nivel de confianza en tales predicciones antes de realizarlas. Por lo tanto, este segundo valor umbral se utiliza para determinar las predicciones que requieren la confirmación del usuario antes de ser realizadas.

Un navegador predictivo permite al usuario monitorear las acciones tomadas por el agente y, por lo tanto, establecer confianza en la capacidad predictiva del agente. El navegador muestra un resumen de las acciones previstas e indica aquellas que tienen una calificación de confianza lo suficientemente alta. El usuario puede confirmar las predicciones con calificaciones bajas, o rechazar las predicciones altamente calificadas, superando así la decisión del

agente. Esta retroalimentación podría utilizarse para ajustar el umbral de confianza para cada clase de acción, como la eliminación, etc., [36].

UNA

UNA ayuda al usuario identificando noticias interesantes en USENet, el cual consistente en un sistema global de discusión en Internet donde los usuarios pueden leer o enviar mensajes (denominados artículos). A medida que el usuario lee cada artículo en una versión modificada de un navegador, proporciona una clasificación para indicar su nivel de interés en él. La clasificación es un número entero en el rango de 1-6. Una calificación de 1 indica que el usuario encontró el artículo extremadamente aburrido o poco interesante, mientras que una calificación de seis indica al agente que el usuario encontró el artículo muy interesante. Los detalles del artículo y las calificaciones se adjuntan a un logfile. Cuando el usuario sale de la interfaz, las características se extraen de este logfile, y se utilizan para generar el perfil de usuario, el cual es utilizado por el motor de clasificación para clasificar futuros artículos de noticias.

Periódicamente (por ejemplo, cada hora) se ejecuta un daemon, que identifica los grupos de noticias a los que el usuario está suscrito, y consulta al servidor de noticias para recuperar todos los nuevos artículos publicados en cada grupo de noticias. Las características se extraen de estos nuevos artículos de la misma manera que para los datos de entrenamiento. Los artículos se clasifican y los resultados pasan a la etapa de predicción, que interpreta los resultados de la clasificación, generando una predicción (en la escala 1-6) del interés del usuario en el artículo.

Cuando el siguiente usuario lee las noticias, puede elegir entre dos modos: modo agente o modo navegador. En el modo navegador, no hay intervención del agente en la presentación de artículos al usuario; Se presentan todos los artículos, independientemente si el agente tiene calificado que son de interés o no. Sin embargo, en el modo agente, el agente marca los nuevos artículos que ha predicho como no interesantes (con calificación 1-3) como si ya se hubieran leído. Todos los artículos que se han pronosticado como interesantes (es decir, con una calificación 4-6) o aquellos que para el agente no pudo generar una predicción, se dejan como no leídos. Con este método, los artículos que se cree que tienen poco o ningún interés se filtran para que no sea mostrado al usuario.

Una ventana que muestra una representación gráfica del estatus del agente se ejecuta de manera permanente en el escritorio del usuario. Con simplemente mirar la ventana puede ver el estatus del agente que ha recibido o no nuevos artículos y si algunos de estos artículos han sido clasificados como interesantes. La ventana puede representar cuatro estados del agente: inactivo, de aprendizaje, apagado o emocionado. El agente se considera inactivo si el daemon

no se está ejecutando y no se han publicado nuevos artículos. El agente está aprendiendo si se está generando un perfil a partir de las observaciones del usuario. El icono apagado indica que se han publicado nuevos artículos en al menos un grupo de noticias y que los artículos se han clasificado como poco interesantes, mientras que cuando aparece el ícono emocionado, se han detectado algunos artículos interesantes [36].

Siri, Cortana y Google Now, limitaciones

Siri de Apple empezó en 2011 una tendencia actual de agentes virtuales, agentes virtuales inteligentes, agentes virtuales móviles, asistente virtual personal, asistente de software inteligente o asistente personal inteligente, estos son algunos términos que se utilizan para referirse a este tipo de sistemas. Tienen la habilidad de interpretar el lenguaje natural de los usuarios, por ejemplo, abrir una aplicación en específico o responder a una pregunta [37].

Un asistente personal inteligente (IPA, en inglés) tiene un conjunto mínimo de programas disponibles para ejecutar, éstos se consideran como acciones, los cuales pueden ser programas para hacer llamadas, enviar mensajes, configurar recordatorios, control de la aplicación de música, interacción con las herramientas de navegación, etc [37].

En cuanto a la capacidad de contestar preguntas el agente soporta desde preguntas factuales generales, tales como "¿Cuál es la capital de Italia?" hasta preguntas de acuerdo a la situación, tal como "¿Cuál es el restaurante chino más cercano?". Esta habilidad de responder a preguntas situacionales se debe a que el agente tiene acceso al GPS y agendas de usuario, mientras que la habilidad de manejar las preguntas factuales se debe a que pueden lanzar otras aplicaciones y tener acceso a servicios de terceros tales como Yelp o Wolfram Alfa [37].

Actualmente, entre los agentes que están teniendo más presencia en el mercado de las tecnologías de la información son los asistentes personales inteligentes, empresas como Microsoft, Apple y Google han lanzado su propio sistema tales como Cortana, Siri y Google Now, respectivamente.

Aunque estos sistemas pareciera que son un gran avance para la inteligencia artificial aún no tienen una inteligencia suficiente para un dialogo profundo con el usuario comparando con Samanta en la película "Her". La mayoría de los IPAs utilizan estrategias de diálogo basadas en reglas tales como confirmación y verificación para asegurar la comprensión con el usuario. Sin embargo, estas estrategias no son suficientes respecto a las amplias preferencias del usuario, contexto y además el ruido de voz del usuario. Los asistentes personales actuales son similares a sistemas de pregunta-respuesta o mando-control. Además, ellos tienden a tratar la mayoría de los enunciados como consultas y comandos individuales. En caso de que

el asistente virtual sea incapaz de entender el contexto, su estrategia es presentar al usuario los resultados de una búsqueda en la web [37].

Lo que realmente se tiene con los IPAs es una colección de funcionalidades (aplicaciones) muy diversos y pueden ser extremadamente útiles pero el todo no es más que la suma de sus partes. Una colección de funcionalidades cada uno con un dominio específico no incrementa la inteligencia [9].

En un mediano plazo, esto podría desbordar. El número de funcionalidades distintas soportadas por la plataforma puede aumentar más allá de lo que puede ser manejado por una arquitectura en específico, lo que puede dar como resultado que el usuario frecuentemente sea dirigido a una funcionalidad que no era la que buscaba: mientras más categorías haya, mayor es el riesgo de caer en la categoría equivocada [9].

Entonces la gran pregunta es si algún día se llegará a construir un agente con una inteligencia artificial “real”. Otros están haciendo la misma pregunta: por ejemplo, los inventores de Siri, ahora están trabajando en Viv Labs, parece que han encontrado el problema y afirman que la solución está en "simplificar radicalmente el mundo, proporcionando una interfaz que es inteligente a todo", como todo proyecto se espera que esto resulte un éxito, y no al contrario [9].

3.1.2. Robots

Los agentes que viven en el mundo real o robots se pueden clasificar también de muchas formas. Se mencionan algunas.

1. Robots Móviles

   a) Robots Espaciales Autónomos

      - Sojourner
      - Spirit
      - Oportunity
      - Rover de la misión MSR (Mars Sample Return, en inglés) en 2020

   b) Robots de Servicio Autónomos

      - Roomba
      - Windows Cleaner

   c) Humanoides

      - ASIMO-Honda

- Humanoide H7, Universidad de Tokyo.
- Humanoide de Toyota.

2. Manipuladores

Robots Móviles

Una de las capacidades que abre una enorme gama de aplicaciones es la movilidad de una máquina por sí misma, es decir, autonomía. Los robots móviles son máquinas que se mueven de manera autónoma en el suelo, en el aire, bajo el agua o en el espacio exterior. En general, se mueven por sí mismos, por medio de una fuente de energía (baterías, por ejemplo), sensores y el sistema computacional necesario que los guía en sus movimientos. Sin embargo, las tareas que realiza estos robots son generalmente supervisadas por un humano. Esta supervisión puede adoptar diversas formas, dependiendo del entorno y la aplicación. Es común utilizar el llamado "control de supervisión" para un monitoreo de alto nivel de los movimientos del robot, movimientos autónomos. En otros casos, se proporciona una interfaz de entrada de comandos de parte del humano esto constituye el llamado "vehículo operado remotamente" (Remotely Operated Vehicle, ROV, en inglés), de allí el nombre Rover. En este caso, el Rover se conecta de manera alámbrica o inalámbrica con el fin de tener comunicación con el operador. Es evidente que un mayor nivel de autonomía es una tendencia importante en las tecnologías relacionadas a estos sistemas robóticos y el modo de operación está siendo gradualmente reemplazado por el control de supervisión [2], [5].

Robots espaciales autónomos

Los Rovers que la NASA (National Aeronautics and Space Administration, en inglés) envía a la superficie de Marte son vigiladas desde la tierra por medio del *control de supervisión* y son capaces de realizar operaciones autónomas (por medio de algoritmos de planeación) muy bien definidas como la navegación de terreno [29].

Por ejemplo, en Julio de 1997, el Rover Sojourner de la misión Mars Pathfinder fue el primero en ser enviado exitosamente a Marte con capacidades autónomas que consistían en navegación del terreno, respuesta de contingencia y gestión de recursos. Describiendo un poco más sobre el caso de la navegación de terreno, se utiliza un sistema de planeación capaz de encontrar caminos con menos obstáculos y con un mejor terreno, por medio de análisis de imágenes, para que sea mínimo:

- El riesgo de vuelco del Rover.
- El tiempo de desplazamiento de un punto a otro.
- El riesgo de deslizamiento debido a las pendientes y el mal contacto con el suelo.

- El riesgo de quedar atascado en suelo suelto.

Aunque el Rover tenía estas característcias de autonomía con cierto grado era necesario enviarle comandos cuando se encontraba con eventos inesperados, por lo que los controladores desde la tierra sufrieron fatiga por proporcionar instrucciones muy detalladas diariamente y la función de respuesta de contingencia fue desactivada cuando la ejecución de las operaciones del Rover fue anormal [27].

Por otro lado, el software de navegación autónoma de Spirit y Oportunity (ambos tocaron la superficie de Marte en Enero de 2004) de la misión MER (Mars Exploration Rover, en inglés) llamado GESTALT (Grid-based Estimation of Surface Traversability Applied to Local Terrain, en inglés) es más avanzado que el del Rover Sojourner porque tienen una mejor capacidad de análisis de imágenes, el Sojourner sólo podía hasta 20 puntos en cada paso, los Rover de la misión MER más de 16 mil puntos y como resultado tienen una mayor capacidad de identificar y evitar obstáculos y seguir en el camino más seguro y rápido; este software utiliza el algoritmo de planeación Field D*. Este software también permite largos tiempos de navegación autónoma del Rover con tan sólo recibir unos comandos enviados desde la Tierra, de hecho, una de las restricciones de la misión MER es que sólo iba a durar 90 soles y una comunicación de dos veces por día [20], [46], [13].

Las capacidades autónomas de los Rovers MER les han permitido explorar mucho la superficie marciana, y los próximos Rovers se basarán en estas capacidades para asegurar el mismo nivel de rendimiento. Sin embargo, las futuras misiones requerirán más autonomía para lograr un éxito mínimo. El objetivo de la misión MSR (Mars Sample Return, en inglés) en 2020 es aterrizar un Rover para que este pueda obtener un conjunto de muestras de la superficie marciana y colocarlos en un vehículo con capacidad de ascenso a un orbitador de Marte para que este pueda regresarlos a la Tierra. Desafortunadamente, la vida útil del combustible del vehículo que ascenderá es aproximadamente de un año. En consecuencia, el Rover MSR debe ser capaz de conducir a muchos lugares de hasta 5 km del sitio de aterrizaje en un período de tiempo muy corto [1].

Uno de los enfoques para lograr que la misión tenga éxito es dotar al Rover con un tipo de autonomía que lo regresaría automáticamente a objetivos previamente vistos, esto incrementaría las muestras que serían devueltas a la tierra. A esta capacidad se conoce como "reconocimiento y retorno", específicamente consiste en que un equipo de científicos pueden ordenar a un Rover para recopilar datos sobre numerosos objetivos científicos y que disponga también de datos sobre otros objetivos relacionados a ellos. Las capacidades actuales en los Rovers de la misión MER requieren que los operadores designen trayectorias de regreso a objetivos y vuelvan a seleccionar los objetivos resultando en operaciones de varios días marcianos que podrían reducirse a un solo día con la capacidad reconocimiento y retorno [1].

Robots de servicios autónomos

En años recientes, ha incrementado la investigación en robots enfocados en realizar tareas de manera autónoma tales como limpieza de pisos y tareas relacionadas a éstas en áreas públicas. De todos los trabajos de investigación, el trabajo en robots móviles se ha centrado en el desarrollo de diversas metodologías que permita moverse de un punto a otro. El problema de limpieza de pisos es especialmente interesante y desafiante debido a su aplicación en diversos campos industriales y públicos. Aunque la mayoría de las máquinas de limpieza siguen siendo guiadas por operadores humanos, ya existen varios robots de limpieza semiautónomos para uso comercial [35].

Un robot de limpieza debe navegar por todo el espacio de trabajo. Es decir, debe haber un algoritmo de navegación de cobertura completa en el que el robot de limpieza navegue por un espacio de trabajo completo en un entorno desconocido utilizando únicamente datos de sensores. Para que un robot de limpieza pueda navegar por completo en un entorno desconocido, se necesitan dos requisitos básicos, representación de mapas y planificación de rutas. El primer requisito es proporcionado por el sistema sensorial que recopila información del robot mismo y del ambiente circundante. El último requisito se ha logrado con la planeación de movimientos que permite al robot navegar por completo en el espacio de trabajo. Generalmente se utilizan algoritmos de planeación para lograr estas tareas, por ejemplo, el método de planificación de trayecto basado en la transformación a distancia (The distance-transform-based path-planning method, en inglés) ó también pueden emplearse métodos de lógica difusa [35], [21].

Humanoides

Los humanoides son máquinas que tienen la forma y función de los seres humanos. Representan el diseño de un sistema completo, combinando la investigación cognitiva con la navegación, la percepción y la manipulación [5].

Uno de los desafíos actuales de los humanoides es la manipulación móvil, esta habilidad se logra al combinar la parte inferior del cuerpo del robot capaz de ubicarse en cualquier lugar con facilidad, y el cuerpo superior del robot que con sus extremidades es capaz de realizar trabajo de valor agregado, es decir, más eficiente que una persona. La parte inferior del cuerpo les permite mantener el equilibrio y acercarse al campo de trabajo mientras que las extremidades de la parte superior del cuerpo se ocupan de maniobrar o realizar tareas. Las personas son ejemplos ideales de manipuladores móviles [5].

Para mejorar la autonomía y la funcionalidad general de estos robots, se necesitan sensores confiables, mecanismos de seguridad, y es necesario herramientas de software integradas. Pero la tecnología más importante para los humanoides es el desarrollo de

algoritmos de planeación que guiará sus movimientos para maniobrar y evitar obstáculos durante la navegación [26].

Se han desarrollado algoritmos de planeación para tareas tales como planeación de pasos (Footstep Planning, en inglés), manipulación de objetos y del movimiento de todo el cuerpo del robot. Para el primer caso se puede adoptar un enfoque de programación dinámica, el cual puede ser generalizado con un algoritmo clásico de búsqueda A*. Para la manipulación existe un algoritmo de planeación eficiente para atacar este tipo de problemas llamado RRT (Rapidly-exploring Random Trees, en inglés) el cual ha sido utilizado para resolver de manera eficiente varios rompecabezas con movimientos bien conocidos tales como el alfa (The "Alpha Puzzle" , en inglés) y la brida (The "Flange", en inglés) los cuales son considerados difíciles debido a que son necesarios movimientos muy minuciosos. Y para el movimiento del cuerpo completo del robot tambien se ha empleado el algortimo RRT [26].

Para la navegación del humanoide Asimo de Honda se utiliza el *Footstep Planning* para evitar obstáculos en el suelo. Para el robot humanoide H7 desarrollado en el Laboratorio JSK de la Universidad de Tokyo se usa también un algoritmo de planeación para calcular sus movimientos para agarrar un objeto parcialmente obstruido por otro objeto en el suelo. Así también el humanoide de Toyota utiliza un algoritmo de planeación para agarrar objetos [8].

Algoritmos de navegación en Robots móviles

Para resolver el problema de navegación en los robots móviles se han utilizado algoritmos de planeación como podemos darnos cuenta, también algoritmos de lógica difusa, pero debido a que el mundo real es un ambiente dinámico los movimientos del robot tienen una cierta cantidad de aleatoriedad, entonces estas soluciones no han dado resultados precisos en todas las condiciones. Para mejorar la capacidad autónoma de los robots, la inteligencia artificial tuvo una tendencia de enfoque en los métodos basados en experiencias. En general, estos métodos pueden ser computacionalmente menos costosos y más fáciles que los métodos de planeación [22].

Uno de éstos métodos basados en la experiencia es el algoritmo Q-learning, el cual tiene muchas características que lo hacen adecuado para resolver el problema de navegación en un entorno dinámico. En primer lugar, el agente que usa este método es un agente de aprendizaje por refuerzo que no tiene conocimientos previos sobre su entorno de trabajo, aprende sobre el medio ambiente al interactuar con él [22].

En segundo lugar, es un agente de aprendizaje en línea, lo cual significa que aprende a tomar la mejor acción en cada estado por prueba-error. Además, puede elegir acciones de forma aleatoria y calcular el valor para tomar una acción en un estado específico. Mediante la evaluación de cada par estado-acción se puede construir una política para trabajar en el medio ambiente. En el problema de navegación del robot conforme encuentra un camino libre

de colisiones también necesita encontrar la mejor acción a tomar en cada estado. El agente obtiene este conocimiento mientras navega por su entorno. Q-learning es una alternativa muy atractiva debido a que es muy simple [22].

Manipuladores

Manipuladores, robots industriales o robots estáticos, en sus inicios no tenían la habilidad de percibir ni tampoco razonar, sólo tenían la capacidad de realizar tareas muy específicas preprogramadas. Actualmente, son equipados con visión computacional y otros sensores para percibir su ambiente y tener cierta autonomía con la ayuda de un procesador que cuenta con algún algoritmo especial [6].

Capítulo 4

Metodología

4.1. Analogía al mundo real

Para tener una idea más clara en cuanto a la mecánica de nuestra experimentación hacemos una analogía de nuestro agente con un murciélago en cuanto a las herramientas que utiliza para cazar una presa.

Como se ha mencionado en la sección 2.13 acerca de la ecolocalización, los murciélagos emiten un sonido que choca y rebota con el objeto o presa, regresando una parte de este sonido a los oídos del depredador, el cual utiliza para determinar la distancia y el tamaño del objeto. Este procedimiento lo hace varias veces hasta ubicar con precisión y atrapar la presa, Fig. 4.1.

Aunque los quirópteros (murciélagos estrictamente nocturnos) utilizan la ecolocalización como herramienta principal de localización para cazar sus presas, la visión también proporciona información valiosa aunque no tan fiable como la primera [41], [33]. En esto consiste la integración multisensorial, usar información de varios sentidos con diferentes niveles de fiabilidad, integrarlos y tomar decisiones sobre esta información integrada.

De la misma manera que el depredador ubica su presa a través de sus sentidos y ejecuta acciones para atraparlo, en este trabajo se presenta un robot (agente) equipado con tres sensores, que pueden ser la combinación de los siguientes telémetros (sensores de distancia) Tabla 4.1, para determinar su ubicación dentro de un cuarto con paredes Fig. 4.2a:

- **Sonar**: emite ondas de sonido, el cual es reflejado por un objeto recibiendo el sensor un eco. El tiempo y la intensidad de la señal de regreso indica la distancia del objeto.
- **Radar**: se basa en emitir una onda electromagnética, el cual es reflejada por un objetivo y recibida por el sensor para calcular una distancia.

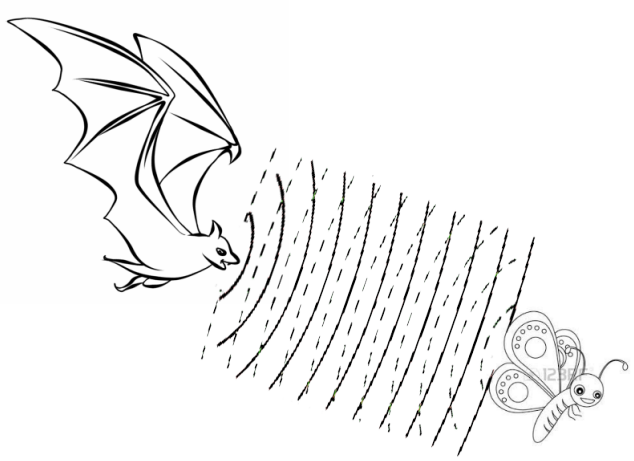


Figura 4.1 Ecolocalización.

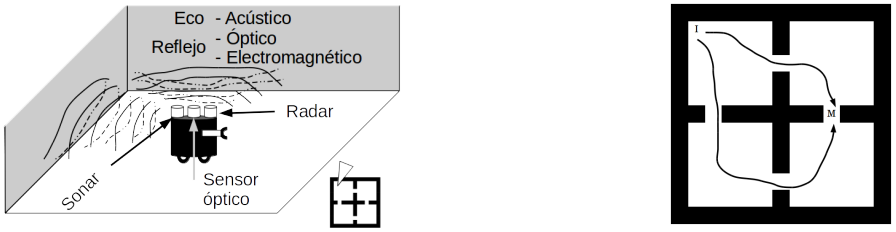
- **Sensor óptico**: similarmente al sonar, con la diferencia de que emite una señal de luz, mide el tiempo de regreso de la luz reflejada para calcular una distancia.

Cada uno de estos sensores tienen un error en la estimación de la distancia, así como la diferencia entre la precisión del oído y la visión de los murciélagos nocturnos. Se incluyen otros dos sensores que funcionan de manera similar en la estimación de la distancia, el primero lo llamamos **Sensor ideal** porque tienen cero error y el segundo lo llamamos **Peor Sensor** porque tiene el peor error. El sonar, el radar y el sensor óptico tienen error pequeño, mediano y grande, respectivamente, como se observa en la Tabla 4.1.

Tabla 4.1 Sensores del robot y sus errores en la estimación de la distancia.

Tipo	**Error**
Sensor ideal	Error cero
Sonar	Error pequeño
Radar	Error mediano
Sensor óptico	Error grande
Peor sensor	Error peor

El objetivo del robot no es atrapar una presa sino ejecutar acciones para moverse de un punto inicial a un punto meta de una manera óptima Fig. 4.2b.



(a) Robot y sus sensores.

(b) Punto inicial y meta en el mundo continuo.

Figura 4.2 En a) se observa que el robot utiliza telémetros para determinar su ubicación en su mundo. En b) muestra dos posibles trayectorias desde el punto inicial hasta el punto meta.

Cabe mencionar que los tipos de sensores mencionados en esta sección sólo es una etiqueta para diferenciar las distribuciones estadísticas utilizadas para la generación de datos mas no tratamos de simular física y exactamente estos sensores en nuestra simulación, sólo es una analogía en cuanto al error que puede tener cada sensor o sentido.

El modelo de ambiente y agente propuestos fueron implementados en el lenguaje de programación C++, en el cual también fue simulado la interacción del agente inteligente con su ambiente.

4.2. Método de aprendizaje Intra-option Q-learning de un sólo paso

Nuestra formulación del mundo o ambiente del agente es tomada en [38] aunque ligeramente alterado para nuestro propósito. Como se ha mencionado cualquier proceso de decisión Markov con un conjunto de options formalmente forma un proceso de decisión semi-markov. Los procesos de decisión semi-Markov tienen la desventaja de aprender con una sóla option a la vez. Si las options son Markov, es decir, que siguen una política π para la ejecución de sus acciones primitivas, entonces podemos considerar los métodos Intra-option, los cuales son más eficientes que los métodos de procesos de decisión semi-Markov porque ellos pueden extraer más ejemplos de entrenamiento para muchas options diferentes a partir de una sóla experiencia o acción. La regla de actualización que se emplea es la siguiente:

$$Q(s_t,o) \leftarrow Q(s_t,o) + \alpha[(r_{t+1} + \gamma U(s_{t+1},o)) - Q(s_t,o)] \,. \tag{4.1}$$

donde

$$U(s,o) = (1-\beta(s))Q(s,o) + \beta(s)\max_{o' \in O} Q(s,o') \; . \tag{4.2}$$

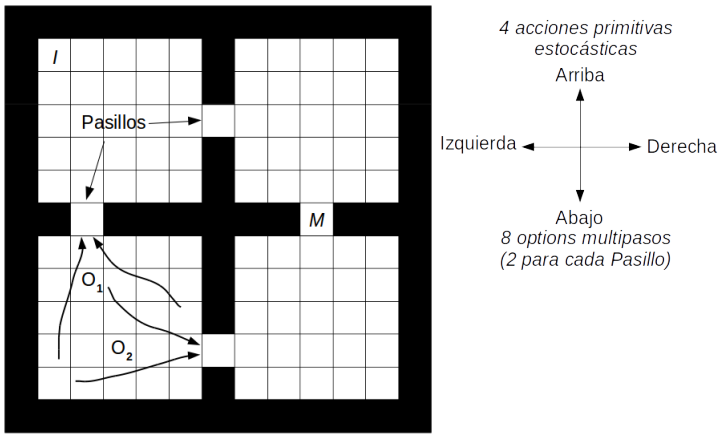


Figura 4.3 Mundo de cuadrículas con acciones estocásticas que avanzan de una celda a otra y options de pasillos que avanzan de un cuarto a otro. Se ejemplifican dos options etiquetados como O_1 y O_2. La M indica la meta del robot en este experimento.

Acciones. Como una ilustración de aprendizaje con options, se considera un mundo de cuadrículas de cuatro cuartos como se muestra en la Fig. 4.3. Las celdas del mundo corresponden a estados del ambiente y el conjunto de acciones primitivas es $\boldsymbol{A}$. La probabilidad de seleccionar una acción es de acuerdo con la política Markov π de una option $o = \langle I, \pi, \beta \rangle$. Si el agente toma un movimiento que choca contra la pared entonces permanece en la misma celda. Sin embargo, esto no ocurrirá porque la política de una option es determinístico en este experimento, Fig. 4.4. Se considera que la recompensa es cero en todos los estados excepto en el estado M, en el cual es 1.

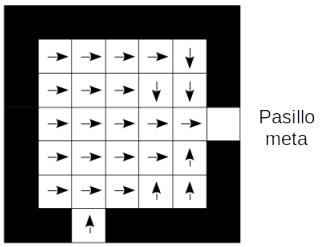
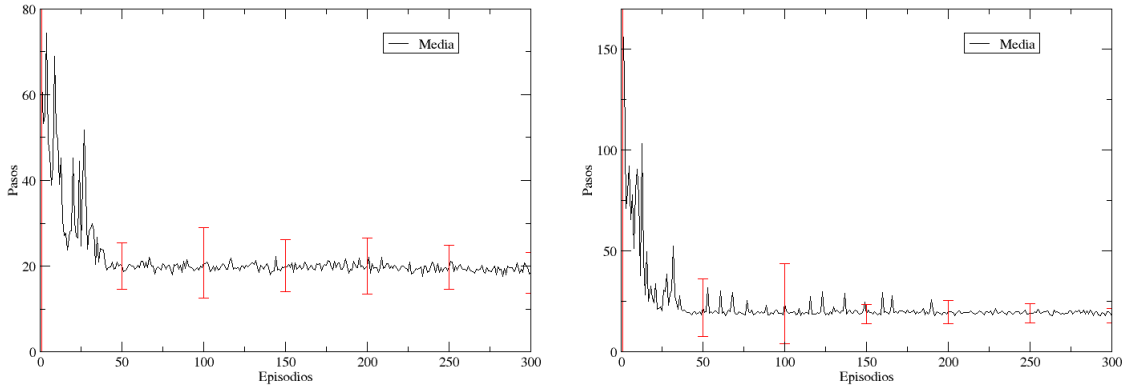


Figura 4.4 Política de una de las ocho options.

Options. En cada uno de los cuatro cuartos se proporciona dos options que llevan a un pasillo, está diseñado para que el agente lo tome en cualquier parte del cuarto para que lo guíe hacia otro cuarto. El conjunto de options disponibles es $\boldsymbol{H}$. La política π de una option sigue el camino más corto dentro del cuarto hasta su pasillo objetivo, minimizando ir hacia

otros pasillos equivocados. Se optó por el método *value iterative* de programación dinámica para construir la política de cada option. Por ejemplo, la política de una option de pasillo se muestra en la Fig. 4.4. La condición de terminación $\beta(s)$ para cada una de las option de pasillo es cero para todos los estados s dentro del cuarto y 1 para los estados fuera del cuarto incluyendo los estados de pasillo. El conjunto de estados iniciales I comprende los estados dentro del cuarto más el pasillo que no es objetivo para esa option. Estas options son determinísticas y Markov, además la política de una option no está definido fuera de su conjunto de estados iniciales. En todo este trabajo el factor de descuento es γ igual a 0.9.

En aprendizaje con Intra-options en este trabajo se utiliza el conjunto de options $\boldsymbol{O} = \boldsymbol{H}$. El valor de α que toma es 0.25. Las options de pasillo fueron seleccionadas de acuerdo al método ε-*greedy*, es decir, se selecciona la option que es máximo para un determinado estado $Q(s_t, o_t) = max_{o \in O_{s_t}} Q(s_t, o)$ pero con probabilidad ε se selecciona cualquier option entre todas las options disponibles en el estado. La probabilidad de elegir una option al azar es ε igual a 0.1 en este experimento. El estado inicial para cada episodio es en la esquina superior izquierda, estado I. Los valores Q para todos los estados son inicializados con el método típico de inicialización de parámetros $[-0.05, 0.05]$ [14].



(a) Mejor curva de aprendizaje.

(b) Peor curva de aprendizaje.

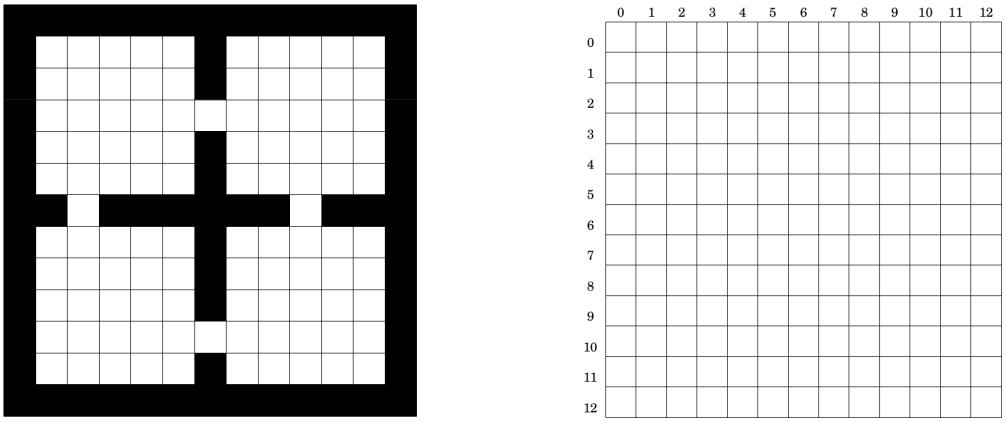
Figura 4.5 Rendimiento del Intra-option Q-learning en el mundo de cuatro cuartos donde el objetivo es **M** y con el conjunto de options **H**. Cada curva de aprendizaje es el promedio de 30 experimentos, las barras son la desviación estándar. En a) los primeros episodios están alrededor de los 60 pasos y la tendencia estable está alrededor del episodio 50. En b) los primeros episodios están alrededor de los 150 pasos y la tendencia estable está alrededor del episodio 200.

En la Fig. 4.5 se muestran las curvas de aprendizaje generadas con el Intra-option Q-learning de un sólo paso, se muestra el mejor y peor comportamiento que se pudo observar, el primero alcanza un rápido aprendizaje estable y el otro tarda hasta el episodio 200.

4.3. Generación de datos

En esta sección se generan los datos a partir de un mundo continuo para el entrenamiento de la red neuronal artificial mapa auto-organizado.

Este es un mundo de cuadrículas, Fig. 4.6a, que puede ser visto como un mundo discreto con filas y columnas, las intersecciones de éstas forman un número finito de estados del mundo. En la Fig. 4.6b se puede observar la ausencia de las divisiones de los cuartos, sin embargo, no significa que ya no se tomen en cuenta, más bien se hace para facilitar la generación de ejemplos de entrenamiento en todo el mundo de cuadrículas.

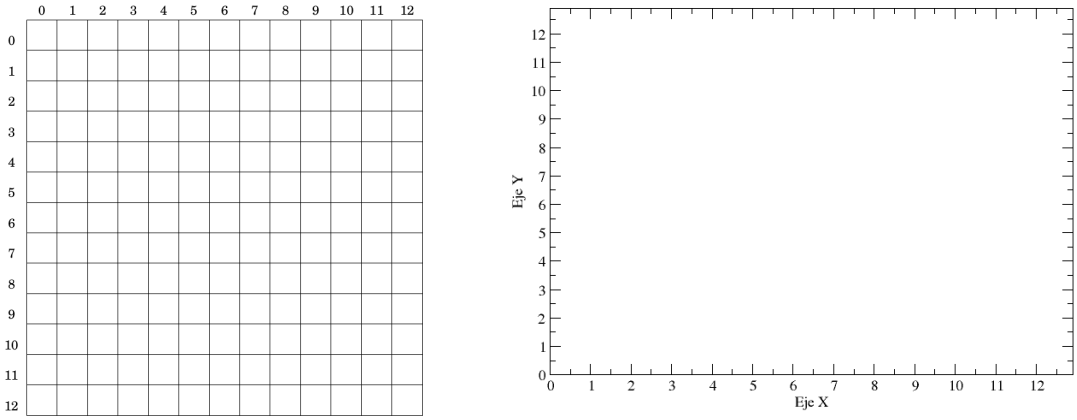


(a) Mundo. (b) Mundo discreto.

Figura 4.6 En a) se observa un mundo que tiene obstáculos o muros donde el agente inteligente aprenderá interactuando. En b) muestra que el mundo está compuesto por un conjunto de estados discretos sin importar si son muros o no.

Hasta aquí se tiene un mundo discreto pero el objetivo de este trabajo es presentar un agente inteligente que posee sensores que perciben datos de un mundo real y que sea capaz de aprender interactuando en un mundo continuo, entonces el mundo discreto Fig. 4.7a se puede convertir en un mundo continuo Fig. 4.7b, viéndolo como un plano en 2D, los ejes x e y toman valores dentro del conjunto A, donde $A = \{x | x \in \Re : 0 \leq x < 13\}$, podemos observar que el mundo discreto fue volteado en el eje y para producir el mundo continuo, sin embargo, esto no afectará para la generación de ejemplos. En el mundo continuo hay un número infinito de estados que pueden ser percibidos.

Los ejemplos de entrenamiento son extraídos del mundo continuo por secciones, las cuales equivalen a los estados discretos del mundo discreto, es decir, la primera sección equivale a extraer ejemplos en el plano 2D donde los ejes x e y toman valores en el conjunto B, donde $B = \{x|x \in \Re : 0 \leq x < 1\}$, la segunda sección en el plano 2D donde el eje x toma valores en el conjunto C, donde $C = \{x|x \in \Re : 1 \leq x < 2\}$ e y los valores en el conjunto D, donde $D = \{x|x \in \Re : 0 \leq x < 1\}$ y así para cada sección, esto es importante tomando en cuenta que los ejemplos es como si fueran extraídos desde un estado discreto en particular pero esto será en el mundo continuo. Es fundamental no perder de vista los estados discretos porque la función de actualización $Q(s,o)$ del Intra-option Q-learning siempre se aplicará al número finito de estados del mundo discreto, extenderlo a estados continuos es posible [45] pero no es el objetivo de este trabajo. Por otro lado, la red neuronal artificial mapa auto-organizado será entrenada con datos del mundo continuo, más adelante se explicará la interacción entre estos dos algoritmos.



(a) Mundo discreto. (b) Mundo continuo.

Figura 4.7 La ausencia de los muros en a) significa que se generará los ejemplos en todo el mundo de cuadrículas, pero estos ejemplos se generarán en el mundo continuo, b).

En la sección 4.1, hemos presentado a nuestro agente inteligente con tres sensores, los cuales pueden ser cualquiera de los telémetros mencionados en la Tabla 4.1 y que proporcionan información o estado de la ubicación (x,y) del agente, con diferente error cada uno. Específicamente, un sensor puede ser modelado con una función de distribución normal bivariada, dado que nuestro agente inteligente se mueve en un mundo continuo en dos dimensiones. A continuación, se presenta una tabla donde se muestra la relación entre un sensor y una función de distribución normal bivariada.

Tabla 4.2 Modelado de un sensor con una función de distribución normal bivariada.

Sensor	**Función de distribución normal bivariada**
Posición real	μ (media)
Error	Σ (matriz de covarianza)

En la sección 2.12.2 se mencionó básicamente dos tipos de función de distribución normal bivariada o función de densidad, la normal circular y la normal elíptica, aunque estos dos tipos pueden seguir variando dependiendo del valor que tome ρ. Sin embargo, en todo este trabajo estaremos utilizando la función de densidad circular, donde ρ siempre será cero y σ_1, σ_2 siempre iguales, por lo tanto, las variables aleatorias continuas siempre serán independientes y con una misma probabilidad de ser tomadas, respectivamente.

A continuación, se modelan los siguientes sensores que tienen una posición real y un error, con una función de densidad circular con μ y Σ, respectivamente.

Tabla 4.3 Modelación del sensor ideal con error cero con una función de distribución normal bivariada.

Sensor ideal	**Función de distribución normal bivariada**
Posición real $x = 1.5, y = 1.5$	$\mu = [1.5, 1.5]$
Error mediano, $\sigma_x = \sigma_y = 0$	$\Sigma = [0,0;0,0]$

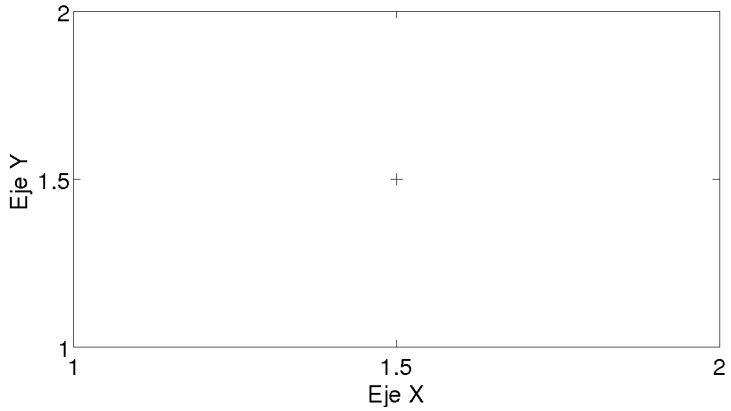


Figura 4.8 Ejemplos extraídos con el sensor ideal con error cero, se observa que siempre se generan los ejemplos en la ubicación real.

En la Tabla 4.3, se muestra las características del sensor ideal con error nulo. Este es el sensor perfecto que siempre va a proporcionar la ubicación real del agente inteligente. Simula un sensor en el mejor de los casos.

Tabla 4.4 Modelación del sonar con error pequeño con una función de distribución normal bivariada.

Sonar	**Función de distribución normal bivariada**
Posición real $x = 1.5, y = 1.5$	$\mu = [1.5, 1.5]$
Error pequeño, $\sigma_x = \sigma_y = 0.001249375$	$\Sigma = [0.001249375, 0; 0, 0.001249375]$

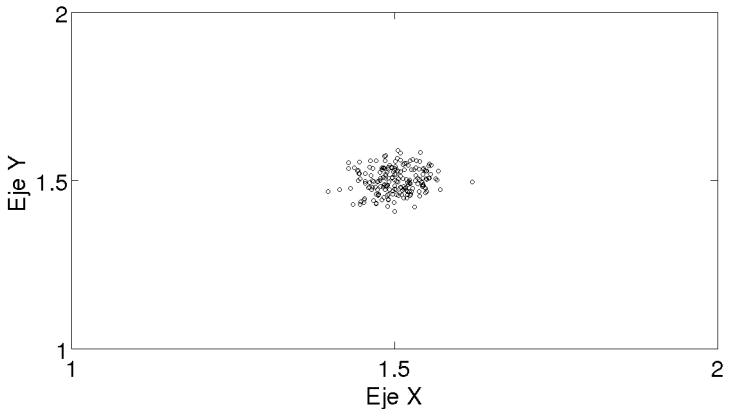


Figura 4.9 Ejemplos extraídos con la función de densidad normal circular que modela un sonar con error pequeño. Se observa que los ejemplos son generados en un área pequeña proporcional a una desviación estándar pequeña.

La Tabla 4.4 muestra los valores que tomarán las variables de la función de densidad circular para ajustarse al comportamiento de un sonar con un error pequeño, este error es relativo. Sin embargo, tomando en cuenta que los ejemplos son extraídos desde el centro de un estado en el mundo discreto pero en el mundo continuo, Fig. 4.6b, se considera que un error es pequeño cuando σ_x y σ_y (ambos siempre son iguales) son lo suficientemente pequeños, de tal manera que, la probabilidad de obtener una muestra es siempre en el centro del estado, y por consiguiente con una pequeña desviación estándar, Fig. 4.9. La técnica es lo mismo en la Tabla 4.5 y 4.6 que muestran las características para el radar con error mediano y el sensor óptico con error grande, la Fig. 4.10 y la Fig. 4.11 las ilustran, respectivamente.

Tabla 4.5 Modelación del radar con error mediano con una función de distribución normal bivariada.

Radar	**Función de distribución normal bivariada**
Posición real $x = 1.5, y = 1.5$	$\mu = [1.5, 1.5]$
Error mediano, $\sigma_x = \sigma_y = 0.0049975$	$\Sigma = [0.0049975, 0; 0, 0.0049975]$

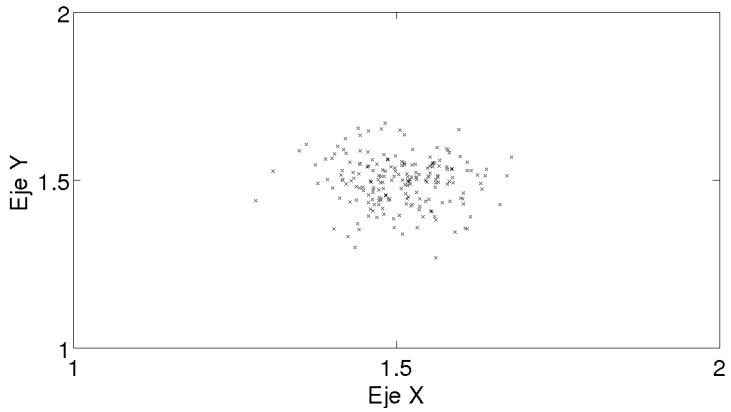


Figura 4.10 Ejemplos extraídos con el radar con error mediano.

Tabla 4.6 Modelación del sensor óptico con error grande con una función de distribución normal bivariada.

Sensor óptico	**Función de distribución normal bivariada**
Posición real $x = 1.5, y = 1.5$	$\mu = [1.5, 1.5]$
Error mediano, $\sigma_x = \sigma_y = 0.01999$	$\Sigma = [0.01999, 0; 0, 0.01999]$

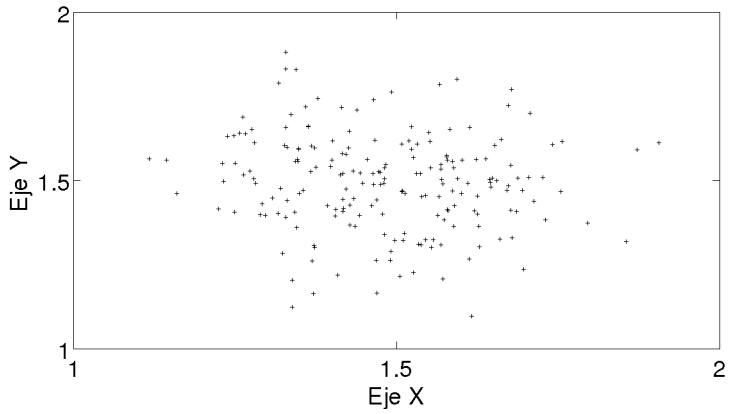


Figura 4.11 Ejemplos extraídos con el sensor óptico con error grande.

Se tiene un último sensor que tiene un error peor, esto significa que algunos de los ejemplos extraídos quedan fuera del estado, Fig. 4.12. También puede entenderse que la función toma muestras de los estados adyacentes. Este es un sensor que tiene un ruido en el peor de los casos.

Tabla 4.7 Modelación del peor sensor con error peor con una función de distribución normal bivariada.

Peor sensor	**Función de distribución normal bivariada**
Posición real $x = 1.5, y = 1.5$	$\mu = [1.5, 1.5]$
Error mediano, $\sigma_x = \sigma_y = 0.07996$	$\Sigma = [0.07996, 0; 0, 0.07996]$

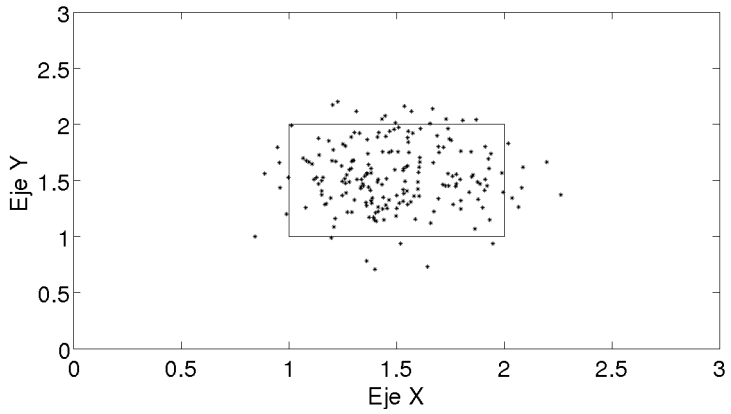


Figura 4.12 Ejemplos extraídos con el peor sensor con error peor.

Estos son los cinco tipos de sensores que serán utilizados durante toda la experimentación en este trabajo.

Se procede a construir siete conjuntos de datos que serán utilizados para el entrenamiento de la red neuronal SOM, cada conjunto corresponde a una experimentación. Cada conjunto de datos es construído utilizando tres sensores, los cuales pueden tomar cualquiera de los tipos de sensores previamente mencionadas o los tres pueden tener un sólo tipo, dependiendo de las características de cada experimentación.

La siguiente tabla muestra las características que tiene cada conjunto de ejemplos en relación a los sensores con los que fueron generados.

Tabla 4.8 Conjuntos de datos que son generados de acuerdo a la combinación de tipos de sensores.

Conjunto de datos	**Combinación de sensores**
1	Los tres son sensores ideales.
2	Los tres son sonares.
3	Los tres son radares.
4	Los tres son sensores ópticos.
5	Los tres sensores son peores sensores.
6	Se tiene un sonar, un radar y un sensor óptico.
7	Se tiene un radar, un sensor óptico y un peor sensor.

En la Fig. 4.13 se muestra el conjunto de datos número 7 que será utilizado para entrenar la red neuronal SOM que corresponde a la experimentación 7. Cabe mencionar que cada uno de los tres sensores generaron 200 casos, 600 tomando en cuenta los tres sensores, y como se tienen 169 estados o secciones, entonces tenemos un conjunto de datos que tiene 101,400 muestras extraídas del mundo continuo; y así se hace para el resto de los conjuntos de datos.

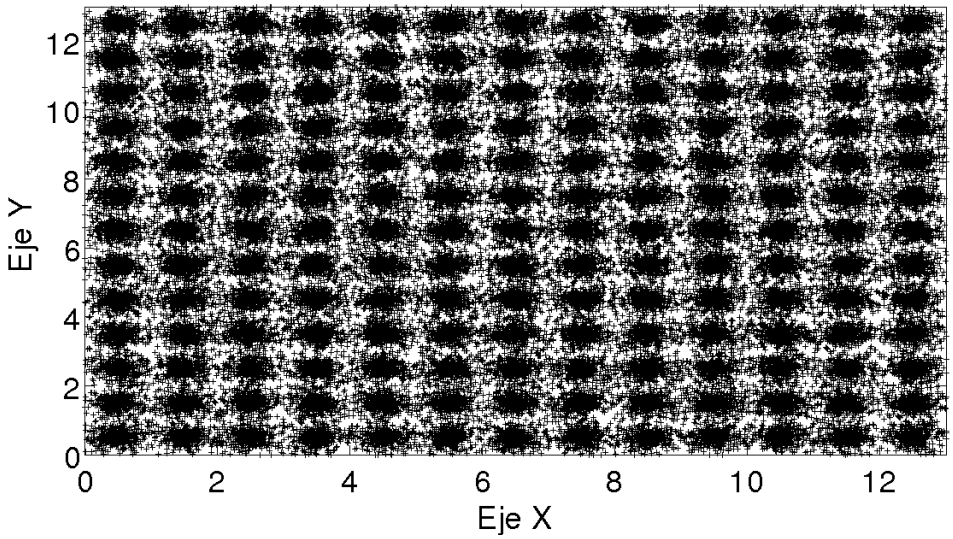


Figura 4.13 Uno de los conjuntos de ejemplos para la experimentación.

4.4. Descripción de la red neuronal SOM implementada

La red neuronal SOM fue implementada de acuerdo a la sección 2.9 y también de acuerdo a los siguientes parámetros.

Se implementa con una arquitectura de acuerdo con la Fig. 4.14; x_1 y x_2 son el patrón de entrada, el cual tomará los valores de la posición x e y que proporciona un sensor, respectivamente. Entonces, cada uno de los m clústers tendrá un vector de pesos con dos elementos que será asociado a un patrón de entrada de acuerdo a una medida de similitud, en esta implementación se utiliza la distancia mínima euclidiana. La topología empleada es la rectangular, Fig. 4.15, con radio R.

La siguiente razón de aprendizaje α es decreciente geométricamente

$$\alpha(0) = a \tag{4.3}$$

$$\alpha(t+1) = 0.5\alpha(t) \tag{4.4}$$

donde t es el número épocas de entrenamiento.

Los pesos iniciales son aleatorios en el intervalo [-0.05, 0.05]. Este es el método comúnmente usado para la inicialización de los pesos, existen otros métodos que son ligeramente mejores; sin embargo, tienen la desventaja de que es necesario ajustar parámetros por prueba y error [14].

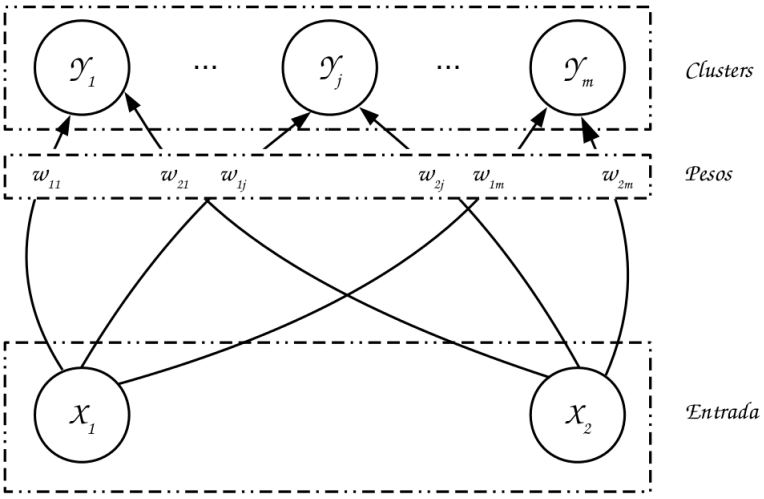


Figura 4.14 Arquitectura de la red neuronal mapa auto-organizado.

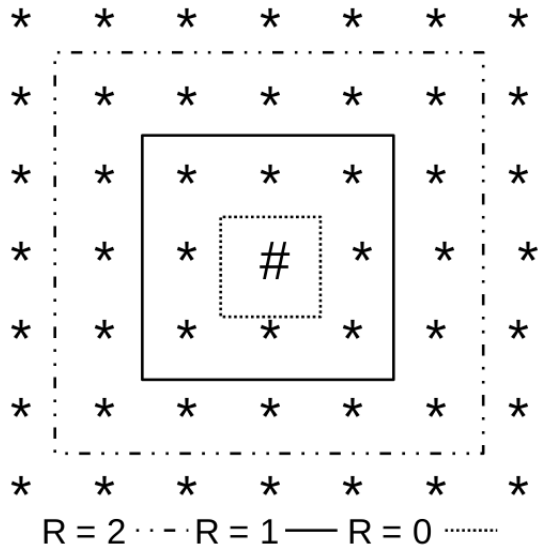


Figura 4.15 Topología rectangular con radio uno.

4.5. Validación cruzada

Para el entrenamiento y evaluación de nuestro algoritmo de aprendizaje SOM se aplicó el método validación cruzada *K-fold cross validation*. El conjunto de datos con el que se entrena y evalúa contiene siempre $101,400$ muestras. Este conjunto fue dividido en 6 subconjuntos como se muestra en la Fig. 4.16; por lo tanto, se hizo 6 veces el entrenamiento y evaluación de la red neuronal, a partir de estos 6 procesos se obtiene la media de la eficiencia del algoritmo.

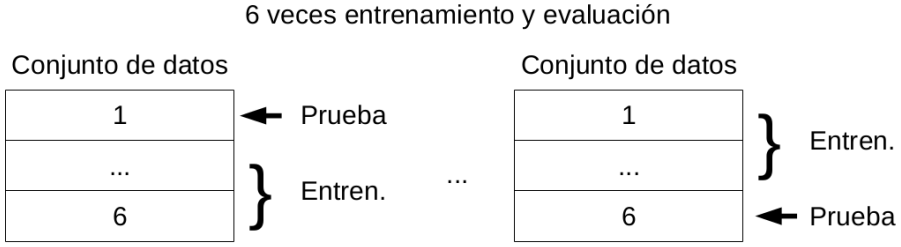


Figura 4.16 Validación cruzada.

4.6. Entrenamiento de red neuronal mapa auto-organizado

Habiendo implementado la red neuronal, generado siete conjuntos de datos y definido el método para evaluar la eficacia del algoritmo, se pudo entrenar y evaluar la red neuronal mapa auto-organizado dando como resultado lo mostrado en las tablas 4.9 y 4.10:

Tabla 4.9 Parámetros de la red neuronal SOM.

Parámetros	**Valores**
Número de clusters	169
Radio	0
Razón de aprendizaje	$\alpha(0) = 0.1$
Épocas	20
Eficiencia	**20.11 %**

En un primer entrenamiento, la Tabla 4.9 muestra los valores de los parámetros de la red neuronal SOM. Se fijó 169 clústers en relación a que se tiene 169 estados en el mundo discreto; se esperaba que cada clúster se formara para cada estado y hacer una perfecta correspondencia con los ejemplos de prueba, sin embargo, no fue así, como lo indica la baja eficiencia de 20.11 %.

Incrementando la razón de aprendizaje a 0.9 y duplicando las épocas de entrenamiento, la red neuronal alcanza un 64.83 % de eficiencia, Tabla 4.10.

Tabla 4.10 Características de la red neuronal SOM.

Parámetros	**Valores**
Número de clusters	169
Radio	0
Razón de aprendizaje	$\alpha(0) = 0.9$
Épocas	40
Eficiencia	**64.83 %**

Se hicieron varias pruebas hasta encontrar los parámetros adecuados para una eficiencia aceptable. La Tabla 4.11 muestra que se requirieron 2,500 clústers, utilizando un radio R igual a 3, y las épocas de entrenamiento se fija en 10 para lograr una eficiencia del 97.01 %

Tabla 4.11 Parámetros de la red neuronal SOM.

Parámetros	**Valores**
Número de clústers	2500
Radio	3
Razón de aprendizaje	$\alpha(0) = 0.1$
Épocas	10
Eficiencia	**97.01 %**

Siguiendo la misma configuración de parámetros de la red neuronal SOM de la Tabla 4.11, se presenta la siguiente tabla donde muestra la eficiencia de la red neuronal con los siente conjuntos de datos previamente construidos.

Tabla 4.12 Eficiencia de la red neuronal SOM de acuerdo al conjunto de datos con la que es entrenada.

Datos	Combinación de sensores	Eficiencia
1	Los tres son sensores ideales.	99.40 %
2	Los tres son sonares.	95.25 %
3	Los tres son radares.	98.44 %
4	Los tres son Sensores ópticos.	96.76 %
5	Los tres sensores son peores sensores.	94.36 %
6	Se tiene un sonar, un radar y un sensor óptico.	97 %
7	Se tiene un radar, un sensor óptico y un peor sensor.	97.01 %

La Fig. 4.17 muestra los clústers entrenados con el conjunto de datos 7, se puede observar que algunos estados apenas tienen algunos clústers en el borde esto puede causar fallas en la estimación de ese estado.

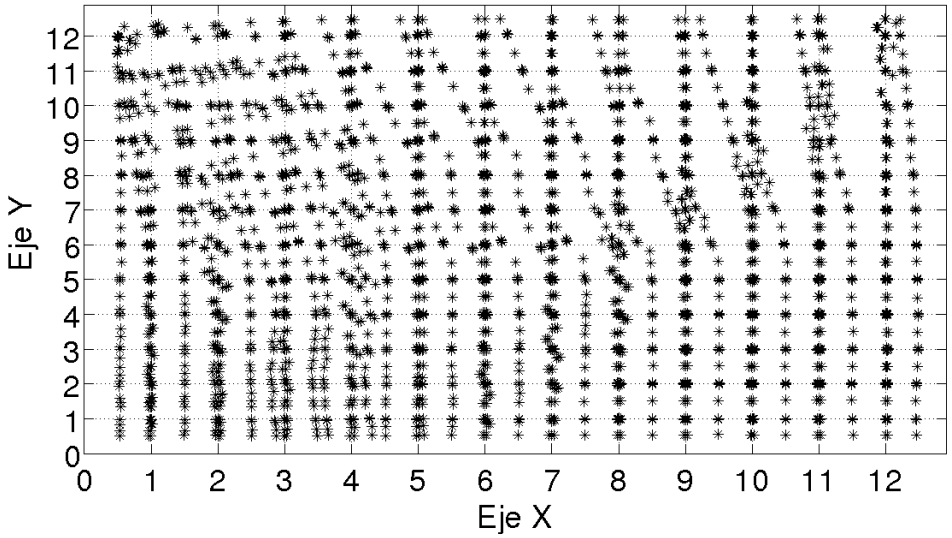


Figura 4.17 Clústers entrenados con la base de datos 7.

4.7. Integración de Intra-options Q-learning y la red neuronal mapa auto-organizado

El objetivo de este trabajo es presentar una arquitectura de agente inteligente que tenga la capacidad de aprender tareas a diferentes niveles de abstracción utilizando Intra-options

Q-learning de un sólo paso; mostrar la capacidad de integración multisensorial para una localización con precisión en un mundo continuo utilizando una red neuronal mapa auto-organizado. Para esto, se emplean tres sensores activos para simular sentidos de percepción.

En las subsecciones anteriores se ha implementado la red neuronal mapa auto-organizado y el método de aprendizaje por refuerzo Intra-options Q-learning de un sólo paso. Ahora lo interesensante es integrar estos algoritmos para que trabajen en cojunto sin que el uno afecte el buen funcionamiento del otro. Después de un análisis se pudo engranar estos dos algoritmos, de tal manera que, nuestro agente inteligente puede aprender a partir de su percepción de un mundo contínuo.

La Fig. 4.18 muestra la arquitectura del resultado de la intregración, y la llamamos *Intra-options + Som*. Según la arquitectura, primero los sensores procesan las señales de entrada v_t que vienen del ambiente. Cada sensor tiene como salida un punto (x,y). Estos puntos que vienen de los tres sensores va a la red neuronal ya entranada. La red neuronal procesa los puntos, y entrega como salida un estado. Dicho estado sirve al agente inteligente para elegir una option o_t, esta es ejecutada sobre el ambiente, el cual responde con una señal v_{t+1} y una recompensa r_{t+1}. Cabe resaltar que la tabla $Q(s,o)$ es actualizada con la recompensa que viene del ambiente o mundo real y de acuerdo con la regla de actualización Intra-options Q-learning de un sólo paso.

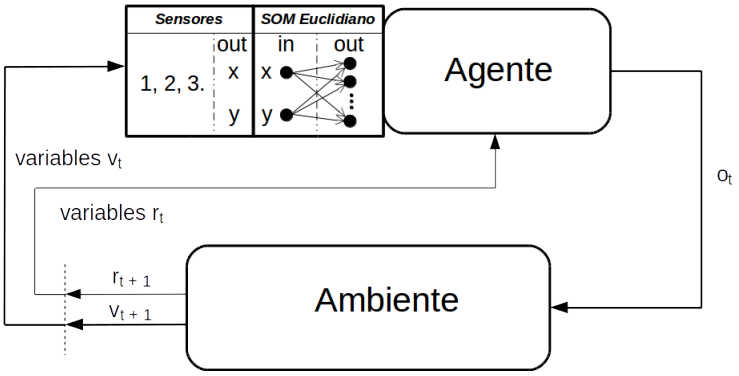


Figura 4.18 Arquitecura de agente inteligente con integración multisensorial y aprendizaje de abstracciones, Intra-options + Som.

El algoritmo del agente inteligente con integración multisensorial y aprendizaje de abstracciones Intra-option + Som queda de la siguiente manera (ver Algoritmo 2):

Algoritmo 2 Algoritmo Intra-option + Som.

Inicializar Intra-options Q-learning, Self-organizing Map y Sensores
Repetir para cada episodio:
SOM($datos_t$ *de entrada de los sensores)*
$s_t \leftarrow$ *SOM estima un* s_t *del ambiente*
Repetir en cada paso de episodio:
$o_t \leftarrow$ *Elegir una* $o \in O_{s_t}$ *de acuerdo* ε*-greedy*
$a_t \leftarrow$ *Elegir una a de acuerdo a la política* π *de* o_t
Ejecutar a_t
SOM($datos_{t+1}$ *de entrada de los sensores)*
$s_{t+1} \leftarrow$ *SOM estima un* s_{t+1} *del ambiente*
Observar la recompensa r_{t+1} *del ambiente*
Actualizar a toda $o \in O_{s_t}$ *de acuerdo a Intra-option Q-learning.*
$s_t \leftarrow s_{t+1}$
Hasta s_t *terminal*

Capítulo 5

Experimentos

5.1. Experimentos

5.1.1. Parámetros de entrenamiento Intra-option + Som

En la sección 4.7 se explica la integración de los algoritmos de aprendizaje por refuerzo y aprendizaje no supervizado, Intra-options Q-learning y SOM, respectivamente. En esta sección se muestran las curvas de aprendizaje que se generaron con el nuevo algoritmo Intra-options + SOM en un mundo continuo Fig. 5.1, con cada uno de los siete conjuntos de datos. Los parámetros utilizados en este nuevo algoritmo son las mismas que las descritas anteriormente para el Intra-options Q-learning y el algoritmo de red neuronal mapa auto-organizado, así como también los parámetros descritos previamente para el mundo continuo y los sensores.

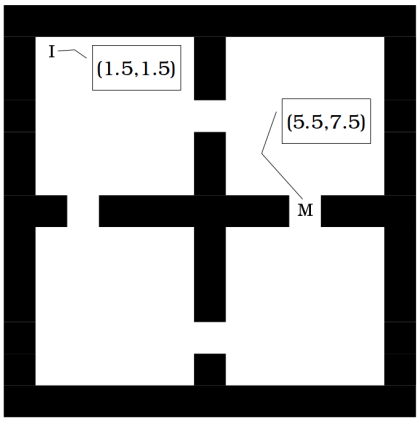


Figura 5.1 Mundo continuo.

Cada conjunto de datos fue generado de acuerdo a un arreglo de tipos de sensores. En la Tabla 4.8 muestra que el primer conjunto de datos fue generado sólo con sensores ideales, es decir, los tres sensores tienen el mismo error o desviación estándar, en este caso cero. La variación de la desviación estándar es lo que hace diferente entre un sensor y otro; por lo tanto, este conjunto de datos no es multisensorial, más bien es unisensorial porque fue generado a partir de un sólo tipo de sensores [Johannes Bauer]. Entonces los conjuntos de datos que fueron generados con el mismo tipo de sensores (conjunto de datos 1, 2, 3, 4 y 5) son unisensoriales, y los que fueron generados por arreglos de tipos de sensores (conjunto de datos 6 y 7) son multisensoriales (ver Tabla 5.1).

Tabla 5.1 Todos los conjuntos de datos caen en dos combinaciones de sensores, unisensorial y multisensorial.

Cojunto de datos	**Combinación sensorial**
1	
2	
3	Unisensorial
4	
5	
6	
7	Multisensorial

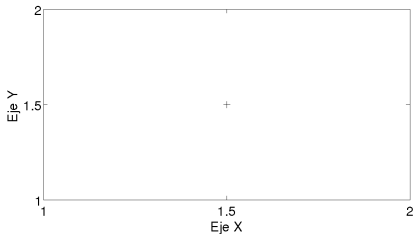
Este experimento se divide entonces en dos partes, experimento unisensorial y experimento multisensorial.

5.1.2. Experimento unisensorial

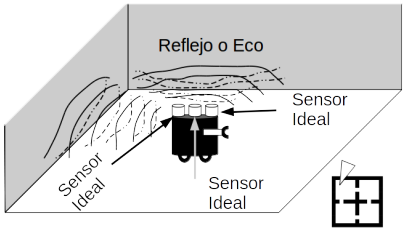
Los experimentos del 1 al 5 son unisensoriales, en los cuales el agente utiliza en sus tres sensores una sóla modalidad sensorial, se tiene cinco modalidades y para cada uno se tiene un experimento. Estos experimentos unisensoriales se hacen con el fin de observar el comportamiento del agente con estas cinco modalidades, dado que cada uno tiene error diferente, entonces se quiere observar el comportamiento del agente en función de estos errores.

Experimento 1

En este experimento el agente tiene sensores ideales, ver Fig. 5.2b. El error de cada uno se representa en la Fig. 5.2a y definido exactamente en la sección 4.3 por una desviación estándar. En todos los episodios el agente inicia en el punto $(1.5, 1.5)$ y con el algoritmo Intra-option + SOM se genera la curva de aprendizaje de la Fig. 5.3 que representa la cantidad de pasos necesarios para llegar en el punto objetivo, el cual es el punto $(5.5, 7.5)$.



(a) Error cero del sensor ideal.



(b) Agente y sus sensores ideales.

Figura 5.2 En a) se observa que el sensor ideal no presenta variación de las muestras extraídas en función de la ubicación real del agente, coordenada $(1.5, 1.5)$. En b) se presenta el agente con sensores de un sólo tipo, sensores ideales.

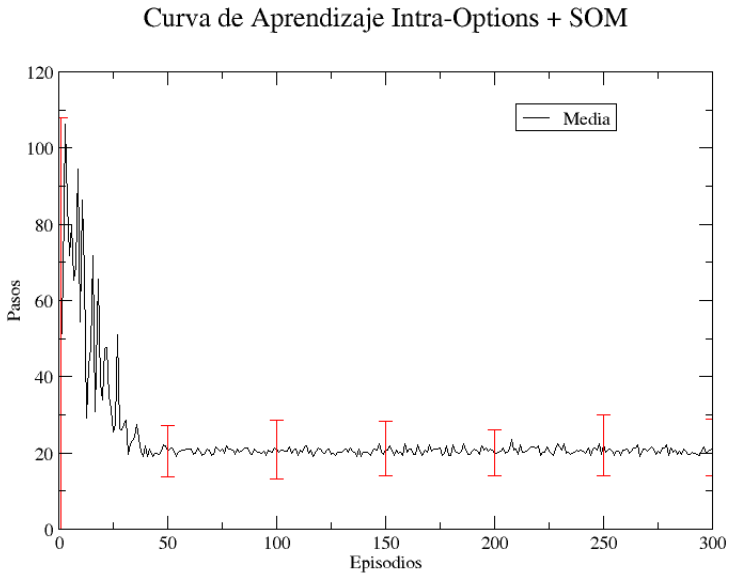


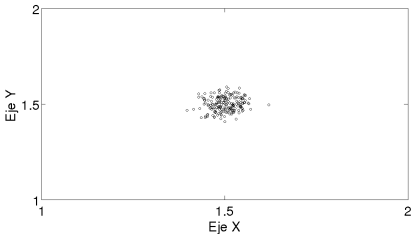
Figura 5.3 Curva de aprendizaje generada con el conjunto de datos 1. El agente utiliza sólo sensores ideales.

Se observa que la curva de aprendizaje tiene un comportamiento similar a la gráfica generada utilizando sólo Intra-options en la sección 4.2, utilizando este conjunto de datos.

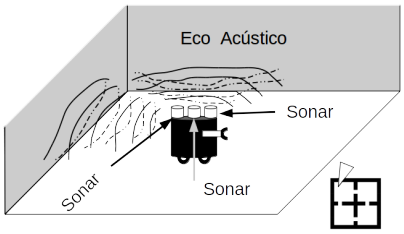
Es como si el agente estuviera navegando en el mundo discreto, ya que la ubicación es siempre exacta con el sensor ideal, entonces la similitud de las gráficas era de esperarse. En los primeros episodios el número máximo de pasos está cerca de los 110 para esta gráfica y para el Intra-options sólamente la mejor curva de aprendizaje está alrededor de los 70. Esta variación se debe a la elección de las options con una probablidad de exploración de 0.1 y a los valores iniciales que toma cada estado-options en Q.

Experimento 2

Con la misma mecánica que en el experimento 1, en el experimento 2 se usan para el agente sonares con el error representado en la Fig. 5.4a. Con el conjunto de datos 2 el agente aprende un comportamiento óptimo que se puede observar en la Fig. 5.5.



(a) Error pequeño del sonar.



(b) Agente con sus sonares.

Figura 5.4 En a) se observa que el sonar presenta una pequeña variación de las muestras extraídas en función de la ubicación real del agente, coordenada $(1.5, 1.5)$. En b) se presenta el agente con sensores de un sólo tipo, sonares.

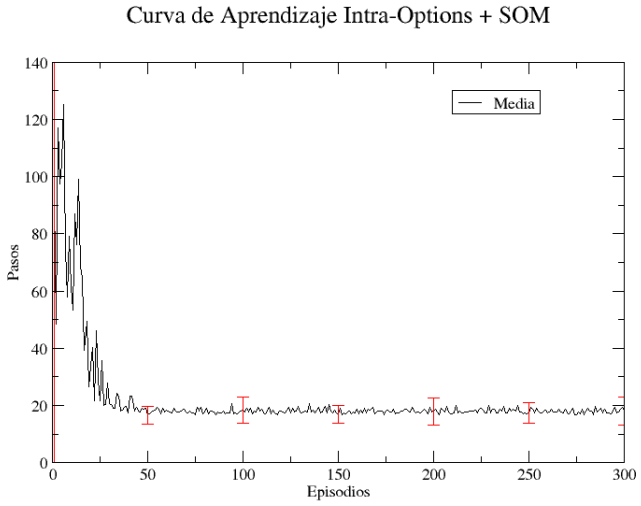
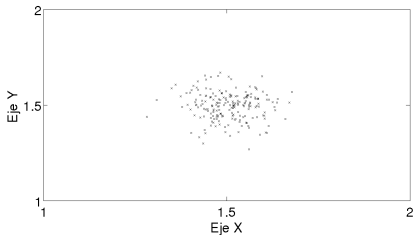


Figura 5.5 Curva de aprendizaje generada con el conjunto de datos 2. El agente utiliza sólo sonares.

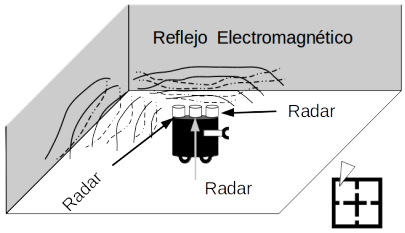
Para este experimento, los primeros episodios tienen un máximo en número de pasos en los 120, y la tendencia estable lo alcanza alrededor del episodio 50, la similitud con la gráfica del Intra-options permanece, el error pequeño de los sonares no afecta el rendimiento del algoritmo Intra-option + SOM.

Experimento 3

La curva de aprendizaje de la Fig. 5.7 es generada cuando el agente se viste de sólo radares, y el error es representado en la Fig. 5.6a.



(a) Error mediano del radar.



(b) Agente con sus radares.

Figura 5.6 En a) se observa que el radar presenta una mediana variación de las muestras extraídas en función de la ubicación real del agente, coordenada $(1.5, 1.5)$. En b) se presenta el agente con sensores de un sólo tipo, radares.

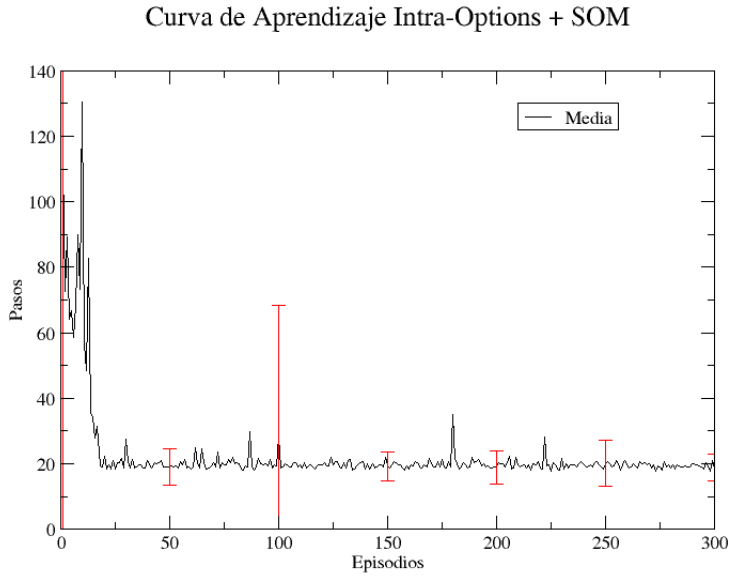
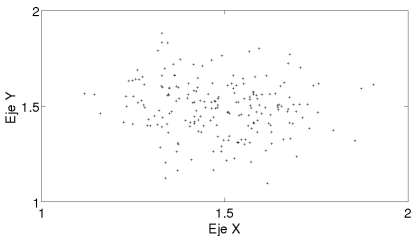


Figura 5.7 Curva de aprendizaje generada con el conjunto de datos 3. El agente utiliza sólo radares.

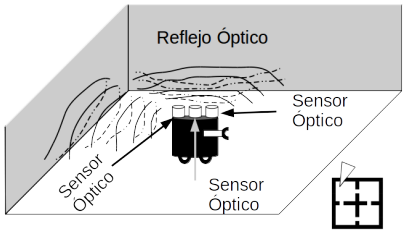
En esta gráfica se puede observar que se dispara hasta en los 130 pasos en los primeros episodios. Este comportamiento es propio del Intra-option, no es debido a la fusión con el algoritmo de redes neuronales SOM, porque en la peor curva en la Fig. 4.5 se puede observar un máximo de pasos de hasta 150 en los primeros episodios. Este comportamiento sucede por la elección y actualización, por una probabilidad de exploración de las options en cada episodio. Y los pequeños picos que se puede ver a lo largo de la curva es porque hay options que no han alcanzado el valor estado-option óptimo en Q.

Experimento 4

Este es un experimento unisensorial, en el cual el agente utiliza sólo sensores ópticos, el error grande se puede observar en la Fig. 5.8a y la curva de aprendizaje es la que se representa en la Fig. 5.9.



(a) Error grande del sensor óptico.



(b) Agente con sus sensores ópticos.

Figura 5.8 En a) se observa que el sensor óptico presenta una variación grande de las muestras extraídas en función de la ubicación real del agente, coordenada $(1.5, 1.5)$. En b) se presenta el agente con sensores de un sólo tipo, sensor óptico.

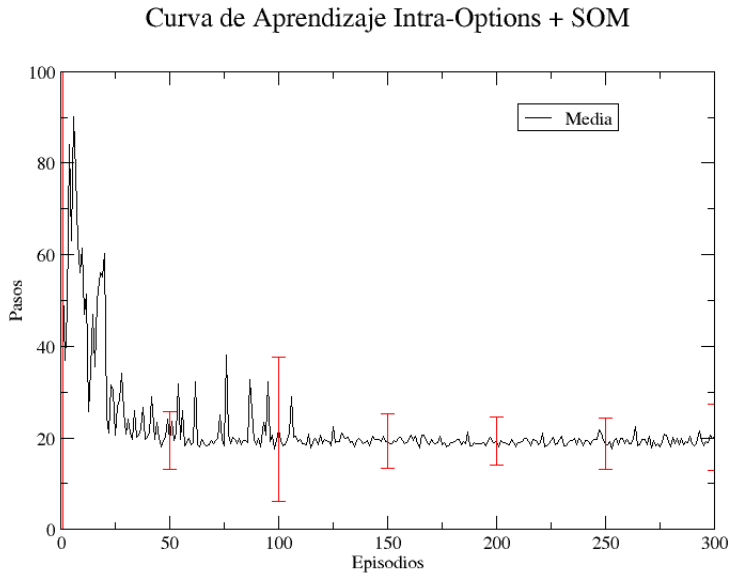
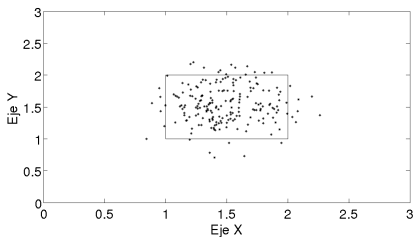


Figura 5.9 Curva de aprendizaje generada con el conjunto de datos 4. El agente utiliza sólo sensores ópticos.

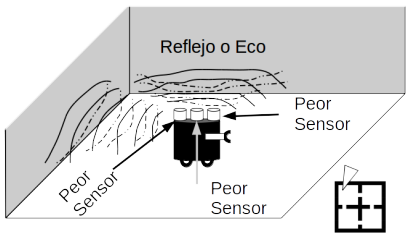
En esta gráfica la curva de aprendizaje tiene en los primeros episodios 90 pasos y tarda hasta alrededor del episodio 100 para alcanzar el valor estado-acción óptimo en Q, luego sigue una tendencia estable.

Experimento 5

En este experimento el agente utiliza el peor sensor. En la Fig. 5.10a se representa el error. En cada uno de sus tres sensores para aprender un comportamiento óptimo en un mundo contínuo, la curva de aprendizaje se muestra en la Fig. 5.11.



(a) Error peor del peor sensor.



(b) Agente con sus peores sensores.

Figura 5.10 En a) se observa que el peor sensor presenta la peor variación de las muestras extraídas en función de la ubicación real del agente, coordenada $(1.5, 1.5)$. En b) se presenta el agente con sensores de un sólo tipo, peores sensores.

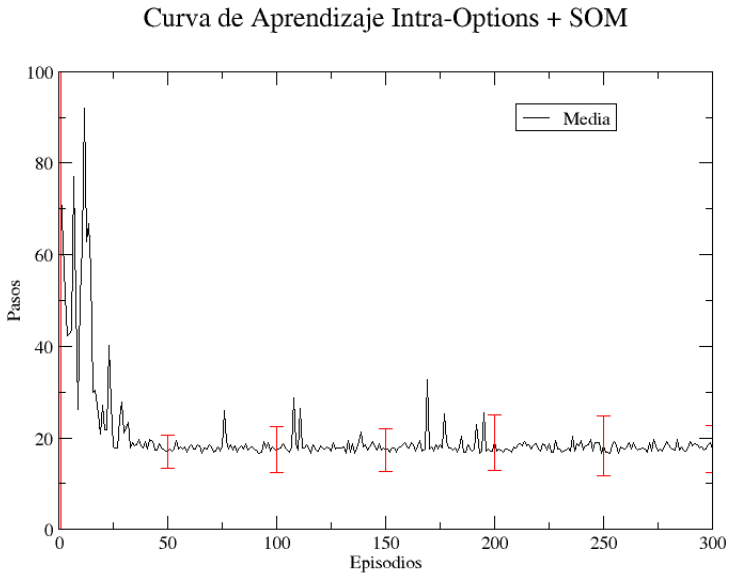


Figura 5.11 Curva de aprendizaje generada con el conjunto de datos 5. El agente utiliza sólo peores sensores.

Para generar esta gráfica, se utilizó el sensor que estima la peor ubicación del agente, pero se puede observar que el comportamiento es el mismo que las gráficas que están en la Fig. 4.5 que son de sólo Intra-options. Hasta este punto se mantiene el rendimiento de Intra-option + SOM como lo hace sólo el algoritmo Intra-option. La tendencia estable tarda hasta el episodio 200.

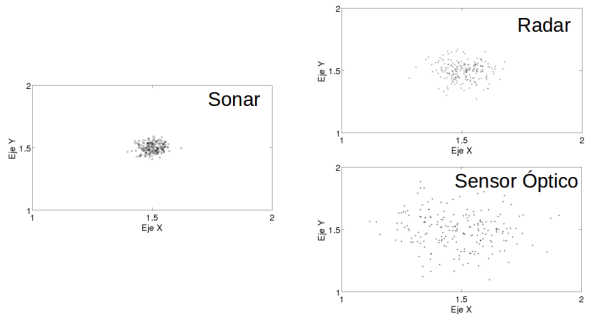
5.1.3. Experimento multisensorial

Los experimentos 6 y 7 son los que más nos interesan porque en este trabajo queremos observar el comportamiento del agente usando la información de varios tipos de sensores e integrarlo para que aprenda un comportamiento óptimo en un mundo contínuo. En estos experimentos se usan tres tipos de sensores, cada uno con diferente error de estimación de la ubicación real del agente. El experimento 6 usa el sonar, el radar y el sensor óptico, el experimento 7 usa el radar, sensor óptico y el peor sensor. Es en estos casos donde se puede observar la integración multisensorial y aprendizaje de abstracciones de nuestro algoritmo generado.

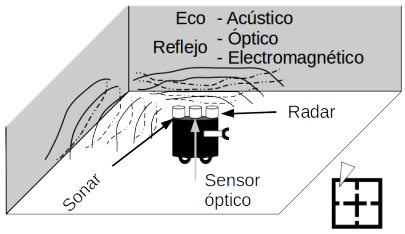
Experimento 6

En la Fig. 5.12a se observan los errores de los tipos de sensores que usa el agente para obtener información del ambiente, el sonar es el que proporciona una mejor ubicación y el

sensor óptico proporciona la peor ubicación en este experimento. Se observa una curva de aprendizaje similar a los experimentos anteriores, Fig. 5.13.



(a) Error del sonar, radar y sensor óptico.



(b) Agente con sonar, radar y sensor óptico.

Figura 5.12 En a) se observa el error de los sensores que usa el agente en función de su ubicación real en la coordenada $(1.5, 1.5)$. En b) se presenta el agente con los sensores que utiliza, los cuales son un sonar, un radar y un sensor óptico.

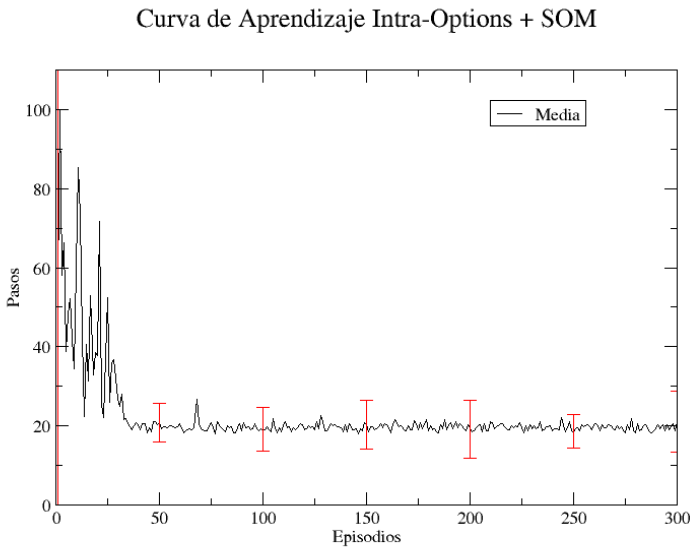
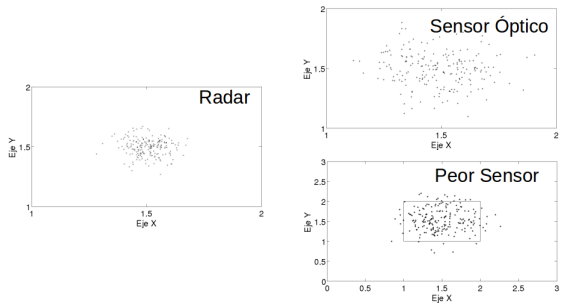


Figura 5.13 Curva de aprendizaje generada con el conjunto de datos 6. El agente utiliza un sonar, un radar y un sensor óptico.

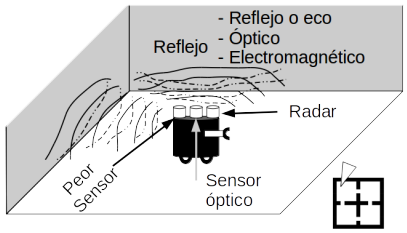
En esta gráfica del experimento 6 se puede notar que lo más importante es el rendimeinto que mantiene el algoritmo Intra-option + SOM porque, aunque se manejan sensores con diferentes errores, logra el valor óptimo en Q en menos de 50 episodios. Se tiene un pequeño pico que significa que alguna option no ha alcanzado el valor óptimo; sin embargo, después de ese mantiene la tendencia.

Experimento 7

En el último experimento (experimento 7), el agente usa los tres sensores que proporcionan una peor ubicación, entre ellos el sensor que llamamos el peor sensor. Aún así se puede observar una curva de aprendizaje sin mayores diferencias que los experiemntos anteriores.



(a) Error del radar, sensor óptico y peor sensor.



(b) Agente con radar, sensor óptico y peor sensor.

Figura 5.14 En a) se observa los errores de los sensores que usa el agente en función de su ubicación real en la coordenada $(1.5, 1.5)$. En b) se presenta el agente con los sensores que utiliza, los cuales son radar, sensor óptico y peor sensor.

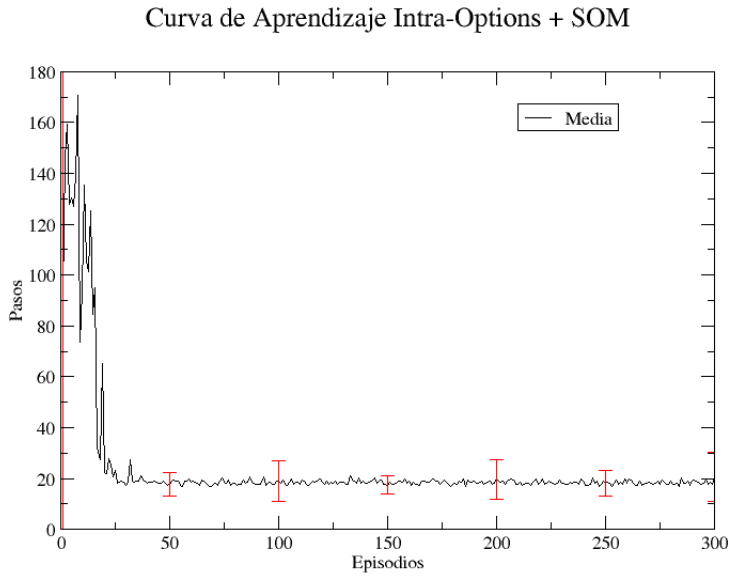


Figura 5.15 Curva de aprendizaje generada con el conjunto de datos 7. El agente utiliza un radar, un sensor óptico y un peor sensor.

Si en algo resalta esta gráfica es que en los primeros episodios tiene el mayor número de pasos los cuales estan alrededor de los 170, alcanza la tendencia estable en menos de los 50 episodios. Una vez más el rendimiento es similar al Intra-options aunque se utilice sensores con la peor estimación de la ubicación del agente.

Capítulo 6

Discusión de los Resultados

En la sección 4.2 se expuso el funcionamiento del algoritmo Intra-options de un sólo paso y se generaron dos gráficas que representan un mejor y peor resultado. Estos dos extremos son causados por dos factores, primero por los valores iniciales aleatorios estado-options, segundo por la elección de las options a través del ε-greedy. El mejor resultado está en que en los primeros episodios el agente llega a la meta en pocos pasos y alcanza rápidamente una tendencia estable y el peor resultado está en que en los primeros episodios tarda muchos pasos para llegar a la meta y también tarda muchos episodios para alcanzar una tendencia estable.

El algoritmo Intra-option + SOM es experimentado en un mundo continuo con siete conjuntos de datos, estos conjuntos de datos se dividen en dos grupos debido a que fueron generados por dos tipos de combinaciones sensoriales, los cuales son unisensoriales y multisensoriales.

Haciendo una comparación en las curvas de aprendizaje generados en los experimentos unisensoriales con las curvas generadas con sólo Intra-option se puede observar que Intra-options + SOM mantiene el rendimiento, como lo hace sólo Intra-options, usando desde sólo sensores ideales hasta sólo peores sensores.

Por otro lado, las curvas de aprendizaje generadas por Intra-option + SOM en los experimentos multisensoriales tambien mantienen el rendimiento como lo hace sólo Intra-option, usando sensores cada uno con diferente error.

Con estos dos grupos de experimentos observamos que el algoritmo Intra-option + SOM mantiene un rendimiento como lo hace el algoritmo Intra-option. Entonces podemos decir que **nuestro agente cuenta con la capacidad de integración multisensorial como con la habilidad de aprender tareas a diferentes niveles de abstracción.**

Capítulo 7

Conclusiones

Intra-option + SOM es el algoritmo desarrollado en este trabajo, su principal característica es dotar a un agente inteligente la capacidad de percepción de su mundo real y ejecutar en él acciones aprendidas de diferentes niveles de abstracción para lograr un objetivo.

El algoritmo desarrollado logra hacer una conexión entre dos áreas del aprendizaje automático, estas son el aprendizaje por refuerzo y el aprendizaje no-supervisado. El aprendizaje por refuerzo nos ofrece un método de aprendizaje muy eficiente llamado Intra-options Q-learning de un sólo paso que tiene la propiedad de aprendizaje de abstracciones. El aprendizaje no-supervisado ofrece un algoritmo de redes neuronales que tiene la propiedad de integración multisensorial. La integración de estos dos algoritmos habilita a nuestro sistema inteligente la capacidad de percibir su mundo real, aprender de ella interactuando, y con el tiempo, por experiencia poder conocer cuáles son las mejores acciones, con diferentes niveles de abstracción, a elegir para lograr de manera óptima su objetivo. Lo novedoso de este sistema basado en aprendizaje por refuerzo es que cuenta con un algoritmo de red neuronal artificial que modela la integración multisensorial, lo cual mejora el sistema para percibir suficiente información para la toma de desiciones o la elección de la acción a ejecutar. Se hicieron los experimentos necesarios, los cuales consistieron en la generación de unas curvas de aprendizaje para el algoritmo Intra-option Q-learning de un sólo paso y generación de curvas de aprendizaje con el nuevo algoritmo Intra-option + SOM. Entonces haciendo una comparación entre las curvas de aprendizaje del primer algoritmo y las curvas de aprendizaje del segundo se puede observar que el rendimiento es el mismo, con la ventaja de que el segundo algoritmo tiene la capacidad de integración multisensorial.

7.1. Trabajo futuro

En este trabajo se desarrolló un algoritmo capaz de percibir el mundo continuo por medio de tres modalidades sensoriales logrando de esta manera que el agente reciba de varias fuentes sensoriales suficiente información para la toma de decisiones sobre las acciones a ejecutar. El ambiente donde el agente interactuó fue estático, es decir, los obstáculos y el objetivo en ningún momento cambiaron de ubicación, entonces en un trabajo futuro sería interesante hacer que este ambiente sea dinámico. Dado el potencial que este algoritmo ofrece de no limitarse a recibir información de sólo tres fuentes sensoriales, pueden ser más, para la toma de decisiones. Los ambientes dinámicos generalmente son muy complejos y tienen un numero infinito de estados. Un detalle de la representación de estados en nuestro agente es que lo hace de manera discreta, incapaz de manejar un número infinito de estados, entonces al interactuar en un ambiente dinámico es necesario hacer que esta representación interna del ambiente sea continuo pudiendo utilizarse algún aproximador de funciones, el cual utiliza recursos computacionales finitos para representar el valor continuo de los pares estado-acción. Este trabajo futuro es importante dado que los robots en el mundo real generalmente trabajan en ambientes dinámicos donde requieren navegar entre obstáculos dinámicos, mover objetos con sus actuadores, agarrar herramientas u objetos para completar sus tareas.

Bibliografía

[1] Bajracharya, M. and Backes, P. (2008). Enabling Reconnaissance and Return for Mars Rovers. In *Proc. 9th Int'l Symp. Artificial Intelligence, Robotics, and Automation in Space*.

[2] Bajracharya, M., Maimone, M. W., and Helmick, D. (2008). Autonomy for Mars Rovers: Past, Present, and Future. *Computer*, 41(12):44–50.

[3] Bauer, J., Weber, C., and Wermter, S. (2012). A Som-Based Model for Multi-Sensory Integration in the Superior Colliculus. In *IJCNN*, pages 1–8. IEEE.

[4] Baujard, O., Baujard, V., Aurel, S., Boyer, C., and Appel, R. D. (1997). A Multi-Agent Softbot to Retrieve Medical Information on Internet. *Studies in Health Technology and Informatics*, 52:150–154.

[5] Bekey, G., Ambrose, R., Kumar, V., Lavery, D., Sanderson, A., Wilcox, B., Yuh, J., and Zheng, Y. (2008). *Robotics: State of the Art and Future Challenges*. World Scientific.

[6] Bekey, G. A. (2005). *Autonomous Robots: from Biological Inspiration to Implementation and Control*. MIT press.

[7] Bennu, D. A. N. (2001). Catching Dinner on the Fly; the Night is Alive with the Sound of Echoes. *Autumn Quarter Biology Department Newsletter*.

[8] Chestnutt, J., Lau, M., Cheung, G., Kuffner, J., Hodgins, J., and Kanade, T. (2005). Footstep Planning for the Honda Asimo Humanoid. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 629–634. IEEE.

[9] Dale, R. (2015). The Limits of Intelligent Personal Assistants. *In Natural Language Engineering*, 21(02):325–329.

[10] Dautenhahn, K. (2000). *Human Cognition and Social Agent Technology*. Advances in Consciousness Research 19. John Benjamins.

[11] Davies, N., Weeks, R., and Revett, M. (1995). Jasper: Communicating Information Agents. In *Proceedings of the 4th International Conference on the World Wide Web*.

[12] Fausett, L. V. (1994). *Fundamentals of Neural Networks: Architectures, Algorithms and Applications.* Prentice-Hall.

[13] Ferguson, D. and Stentz, A. (2005). The Field D* Algorithm for Improved Path Planning and Replanning in Uniform and Non-Uniform Cost Environments. *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-19*.

[14] Fernandez-Redondo, M. and Hernandez-Espinosa, C. (2001). Weight Initialization Methods for Multilayer Feedforward. In *ESANN*, pages 119–124.

[15] Foner, L. N. (1997). Entertaining Agents: A Sociological Case Study. In *Proceedings of the First International Conference on Autonomous Agents*, pages 122–129. ACM.

[16] Franklin, S. and Graesser, A. (1996). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In *International Workshop on Agent Theories, Architectures, and Languages*, pages 21–35. Springer.

[17] George A. Bekey (auth.), G. A. B. e. (1998). *Autonomous Agents*. Springer US, 1 edition.

[18] Ghahramani, Z. (1995). *Computation and Psychophysics of Sensorimotor Integration*. PhD thesis, Citeseer.

[19] Ghahramani, Z. (2004). Unsupervised Learning. In *Advanced Lectures on Machine Learning*, pages 72–112. Springer.

[20] Goldberg, S. B., Maimone, M. W., and Matthies, L. (2002). Stereo Vision and Rover Navigation Software for Planetary Exploration. In *Aerospace Conference Proceedings, 2002. IEEE*, volume 5, pages 5–5. IEEE.

[21] Gueaieb, W. and Miah, M. S. (2008). An Intelligent Mobile Robot Navigation Technique Using RFID Technology. *IEEE Transactions on Instrumentation and Measurement*, 57(9):1908–1917.

[22] Jaradat, M. A. K., Al-Rousan, M., and Quadan, L. (2011). Reinforcement Based Mobile Robot Navigation in Dynamic Environment. *Robotics and Computer-Integrated Manufacturing*, 27(1):135–149.

[Johannes Bauer] Johannes Bauer, S. W. Self-Organized Neural Learning of Statistical Inference from High-Dimensional Data. *In Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*.

[24] Keil, F. C. (1992). *Concepts, Kinds, and Cognitive Development*. MIT Press.

[25] Kohonen, P. T. (1989). *Self-Organization and Associative Memory*. Springer Series in Information Sciences 8. Springer-Verlag Berlin Heidelberg, 3 edition.

[26] Kuffner, J., Nishiwaki, K., Kagami, S., Inaba, M., and Inoue, H. (2005). Motion Planning for Humanoid Robots. In *Robotics Research. The Eleventh International Symposium*, pages 365–374. Springer.

[27] Landzettel, K., Steinmetz, B.-M., Brunner, B., Arbter, K., Pollefeys, M., Vergauwen, M., Moreas, R., Xu, F., Steinicke, L., and Fontaine, B. (2004). A Micro-Rover Navigation and Control System for Autonomous Planetary Exploration. *Advanced Robotics*, 18(3):285–314.

[28] Lieberman, H. (1997). Autonomous Interface Agents. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 67–74. ACM.

[29] Maimone, M. W., Leger, P. C., and Biesiadecki, J. J. (2007). Overview of the Mars Exploration Rovers Autonomous Mobility and Vision Capabilities. In *IEEE International Conference on Robotics and Automation (ICRA) Space Robotics Workshop*.

[30] Marsland, S. (2009). *Machine Learning: An Algorithmic Perspective*. CRC Press, 1 edition.

[31] Merran Evans, N. A. J. Hastings, B. P. (1993). *Statistical Distributions*. Wiley-Interscience, 2 sub edition.

[32] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Series in Computer Science. McGraw-Hill, 1 edition.

[33] Müller, B., Glösmann, M., Peichl, L., Knop, G. C., Hagemann, C., and Ammermüller, J. (2009). Bat Eyes Have Ultraviolet-Sensitive Cone Photoreceptors. *PLoS ONE*, 4(7):1–7.

[34] Nwana, H. S. (1996). Software Agents: An Overview. *The Knowledge Engineering Review*, 11(03):205–244.

[35] Oh, J. S., Choi, Y. H., Park, J. B., and Zheng, Y. F. (2004). Complete Coverage Navigation of Cleaning Robots Using Triangular-Cell-Based Map. *IEEE Transactions on Industrial Electronics*, 51(3):718–726.

[36] Payne, T. R., Edwards, P., and Green, C. L. (1997). Experience with Rule Induction and K-Nearest Neighbor Methods for Interface Agents that Learn. *IEEE Transactions on Knowledge and Data Engineering*, 9(2):329–335.

[37] Riccardi, G. (2014). Towards Healthcare Personal Agents. In *Proceedings of the 2014 Workshop on Roadmapping the Future of Multimodal Interaction Research including Business Opportunities and Challenges*, pages 53–56. ACM.

[38] Richard S. Sutton, D. P. and Singh, S. (1999). Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence*, 112:181–211.

[39] Samuel Kotz, N. Balakrishnan, N. L. J. (2000). *Continuous Multivariate Distributions*. Volumen 1 of Wiley Series in Probability and Statistics. Wiley, 2nd edition.

[40] Schneider, J. and Moore, A. W. (1997). *A Locally Weighted Learning Tutorial Using Vizier 1.0*.

[41] Schwab, I. and Pettigrew, J. (2005). A Choroidal Sleight of Hand. *British Journal of Ophthalmology*, 89(11):1398–1398.

[42] Seydim, A. Y. (1999). Intelligent Agents: A Data Mining Perspective. *Southern Methodist University, Dallas*.

[43] Song, H., Franklin, S., and Negatu, A. (1996). Sumpy: A Fuzzy Software Agent. In *Proceedings of the ISCA Conference on Intelligent Systems*, pages 124–129.

[44] Stuart Russell, P. N. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, 3rd edition.

[45] Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition.

[46] Volpe, R., Estlin, T., Laubach, S., Olson, C., and Balaram, J. (2000). Enhanced Mars Rover Navigation Techniques. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 926–931. IEEE.

[47] Wicklin, R. (2013). *Simulating Data with SAS*. SAS Institute.