

e_Puck Repartidor

Yamil Mateo Rodríguez y José Antonio Antona Díaz
(GR09)

Mayo 2020



Figura 1: E_puck repartidor

Índice

1. Enunciado y Objetivo del Trabajo	3
2. Introducción a la idea	3
3. Mapa	4
4. Redes neuronales artificiales	5
5. Redes neuronales artificiales estáticas	6
5.1. Función <i>fitness</i>	7
5.2. Simulación del robot	9
5.2.1. Gráficas complementarias	10
6. Redes neuronales artificiales dinámicas	12
6.1. Simulación del robot	13
6.1.1. Gráficas complementarias	14
7. Comparación de arquitecturas y conclusiones	16
8. Problemas encontrados	16
9. Posibles implementaciones futuras	16

1. Enunciado y Objetivo del Trabajo

Enunciado: Implementar en el simulador IRSIM un comportamiento mediante una red neuronal artificial estática sintonizando los parámetros con algoritmos genéticos para la resolución de una tarea a decidir libremente por el grupo. Extender dicha arquitectura a una red neuronal artificial dinámica y comparar los resultados con la red neuronal artificial estática.

Objetivo: La finalidad del trabajo será adquirir conocimientos básicos de redes neuronales artificiales, tanto estáticas como dinámicas, y conocer la diferencia entre ambas redes neuronales. Para ello, realizaremos experimentos en el simulador IRSIM. Se analizarán y presentarán los resultados obtenidos, y se realizará una comparación entre la conducta obtenida y la conducta deseada con ambas redes neuronales. Se pretenden conseguir comportamientos relacionados con evitar obstáculos y realizar trayectorias hacia diversas luces, con el fin de cumplir objetivos que se van a detallar a continuación, como recargar una batería o recoger objetos.

2. Introducción a la idea

Actualmente se han incrementado las ventas “*online*” de manera significativa con respecto a años anteriores. Dicho incremento ha tenido como consecuencia la creación de nuevos puestos de trabajo, entre ellos, de repartidor. Nuestro e_Puck va a aprovechar la oportunidad que se le ha puesto delante y se va convertir en un repartidor de paquetes en una empresa de ventas “*online*”. Por lo que nuestro e_Puck, en primer lugar, tendrá como objetivo ir a hasta el lugar donde se sitúa la empresa, señalado con suelo negro y luz amarilla, a partir de ese momento irá a la casa del cliente correspondiente, marcado con suelo gris y luz azul, donde entregará su paquete y volverá de nuevo a la empresa. Este ciclo se repite continuamente, hasta que se acaba su jornada laboral y volverá a su casa, marcado con una luz roja, y descansará, es decir, recargará la batería, hasta que vuelva a empezar su jornada de trabajo y repita el ciclo indicado anteriormente.

3. Mapa

A continuación, se muestra una figura del concepto de mundo descrito en la introducción.

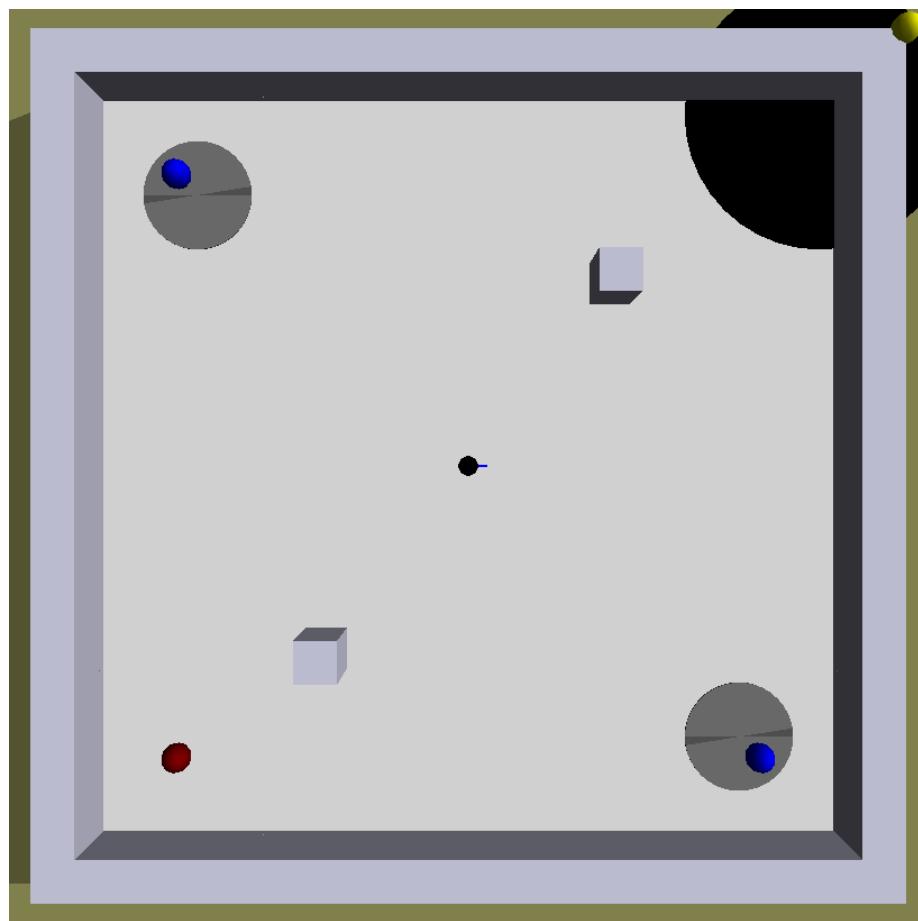


Figura 2: Mapa

- *Suelo negro y luz amarilla*: hacen referencia a la empresa de venta “online”.
- *Luz roja*: hace referencia a la casa del e_Puck repartidor.
- *Suelo gris y luz azul*: hacen referencia a las casas donde se entregan los paquetes.

4. Redes neuronales artificiales

Las redes neuronales artificiales son mecanismos de procesado de información que se inspiran en redes neuronales biológicas. Son sistemas de neuronas conectadas entre sí cuyo principal objetivo es desarrollar operaciones de síntesis y procesamiento de información, relacionadas con los sistemas biológicos. Las neuronas reciben entradas procedentes de otras neuronas o de una fuente externa de datos. Cada entrada tiene un peso asociado w , que permitirá asociar más o menos importancia a las distintas entradas. Además, cada neurona aplica una función de activación al resultado de sumar las entradas ponderadas, mediante sus pesos.

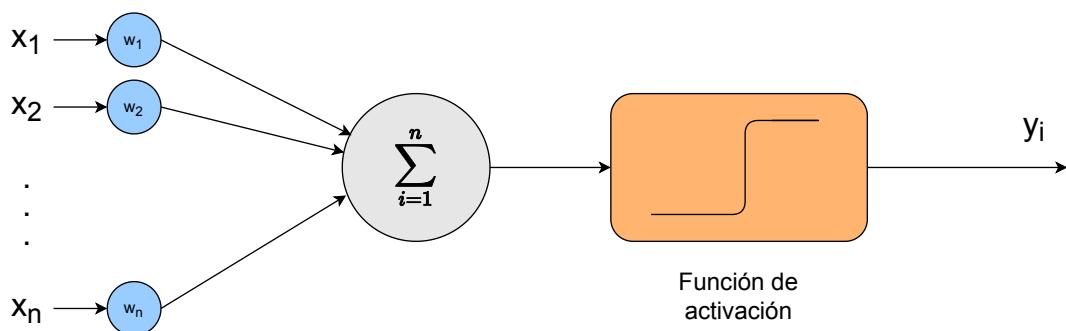


Figura 3: Neurona

No pretenden incorporar una solución diseñada desde el primer instante, sino que describen las primitivas del robot en forma de cromosomas artificiales, dichos cromosomas estarán diseñados con una disposición aleatoria de los genes. Los algoritmos empleados utilizarán poblaciones con individuos, que serán representados a partir de cromosomas, y estos cromosomas estarán representados a su vez por cadenas de dígitos binarios. El sistema probará los distintos cromosomas, y se seleccionarán y reproducirán, aquellos que obtengan una mejor “fitness”, es decir, aquellos que obtengan las aptitudes requeridas. De esta manera se establece un criterio de supervivencia para el robot y solo los cromosomas que permitan al robot desempeñar la conducta deseada serán seleccionados. Los nuevos cromosomas se generarán a través de copias, mutaciones o combinaciones de los cromosomas de los individuos más aptos.

Este proceso de testeo o entrenamiento de los cromosomas se realizará repetidamente, hasta que se obtenga un rendimiento medio de la población lo suficientemente bueno para llevar a cabo el comportamiento para el que ha sido diseñado.

Las principales características de las redes neuronales artificiales son:

- *Algoritmos de aprendizaje adaptativo.*
- *Procesamiento de datos no lineal.*
- *Procesamiento de datos de manera paralela entre las distintas neuronas.*

5. Redes neuronales artificiales estáticas

En las redes neuronales artificiales estáticas la información se propaga en una sola dirección, de las entradas a las salidas, es decir, ante unas entradas proporcionan una salida, por lo que en la ejecución convergen y garantizan una estabilidad del sistema.

Nuestra red neuronal artificial estática tendrá la siguiente arquitectura:

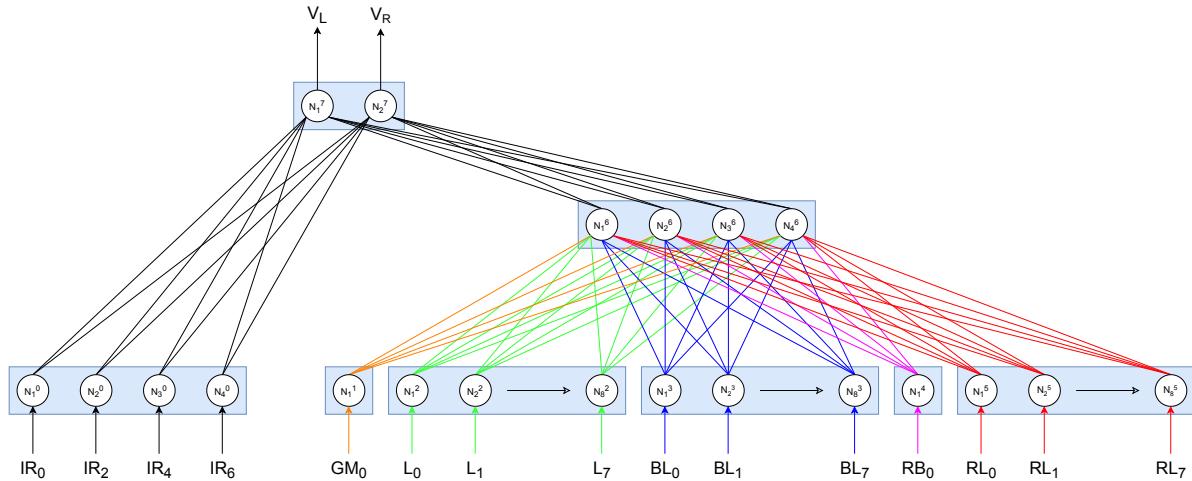


Figura 4: Arquitectura ANN

Por un lado, se dispondrá de seis capas sensoriales, que como su nombre indica, recibirán las entradas de los sensores correspondientes. Los sensores que se van a utilizar para llevar a cabo la conducta deseada del robot serán los sensores de proximidad, suelo con memoria, luz amarilla, luz azul, luz roja y batería roja. Cada una de las capas sensoriales recibirán como entradas los siguientes sensores:

- **Capa 0:** sus neuronas recibirán información de cuatro de los ocho posibles sensores de proximidad.
- **Capa 1:** su neurona recibirá información del único sensor de suelo con memoria.
- **Capa 2:** sus neuronas recibirán información de los ocho sensores de luz amarilla.
- **Capa 3:** sus neuronas recibirán información de los ocho sensores de luz azul.
- **Capa 4:** su neurona recibirá información del único sensor de batería roja.
- **Capa 5:** sus neuronas recibirán información de los ocho sensores de luz roja.

Por otro lado, dispondremos de una capa oculta con cuatro neuronas, que recibirán la información de las capas sensoriales desde la número 1 hasta la número 5.

Por último, se utilizará una capa motora cuya salida conecta con los actuadores del robot, en nuestro caso con las ruedas, y recibirá como entradas las salidas de la capa oculta y la capa número 0.

5.1. Función fitness

Como se ha comentado anteriormente, el sistema probará los distintos cromosomas de cada uno de los individuos, para ello se utilizará una función denominada *fitness*, de la cual se obtendrá un resultado en función del comportamiento del robot. Dicho resultado tendrá un valor entre 0 y 1, obteniendo un 0 cuando el robot no cumpla con el comportamiento deseado y se irá incrementando dicho valor hasta 1, según se cumpla con los distintos requisitos, relacionados con la conducta deseada, que se determinarán en la función *fitness*. Antes de explicar la función *fitness*, se explicará la conducta deseada del robot. A continuación, se muestran los subcomportamientos que el robot debe cumplir, en orden de mayor a menor gravedad sobre la penalización que supondría la desobediencia de dicho subcomportamiento en el valor de la *fitness*:

- Debe evitar chocar con los obstáculos del mapa.
- Debe coger y dejar al menos 5 paquetes o será penalizado.
- Si detecta el nivel de batería roja por debajo del umbral establecido, debe ir en dirección a la luz roja para cargar dicha batería.
- Si no se ha repartido ningún paquete, es decir no ha entrado en zona gris, debe ir en dirección a la luz azul que detecte con mayor intensidad.
- Si se ha repartido ya el paquete, es decir ha entrado en zona gris, debe ir en dirección a la luz amarilla y entrar en la zona negra.

La función *fitness* nos permite establecer penalizaciones cuanto más se aleje del subcomportamiento deseado, de manera que si no se cumple con el objetivo se “castigará” al robot según como se desvíe de ese subcomportamiento, generalmente esto se lleva a cabo multiplicando el resultado por un factor pequeño. Nuestra función *fitness* será la siguiente:

$$F = \frac{\sum_{i=0}^{N_{Steps}} BattN * A * V(1 - \sqrt{\Delta v})(1 - \max\{IR_i\}) * Sensores}{N_{Steps}} \left(1 - \min\left\{\frac{N_{coll}, 30}{30}\right\}\right) \left(\frac{F_{Obj}}{5}\right) \quad (1)$$

$$BattN = \min\left\{\frac{RBatt, 0,2}{0,2}\right\} \quad (2)$$

$$A = |M_{right} * M_{left}| \quad (3)$$

$$V = |M_{right} - 0,5| + |M_{left} - 0,5| \quad (4)$$

$$F_{Obj} = \min\{GreyCounter, 5\} \quad (5)$$

$$Sensores = \begin{cases} RBatt > 0,35 \rightarrow & \begin{cases} GM = 0 & (BL_0 + BL_2 + BL_5 + BL_7) \\ GM = 1 & \frac{(L_0 + L_2 + L_5 + L_7)}{1,5} \end{cases} \\ RBatt < 0,35 \rightarrow & 2 * (RL_0 + RL_2 + RL_5 + RL_7) \end{cases} \quad (6)$$

Primero se van a explicar los factores que penalizan de una manera más significativa al robot. Por un lado, se penalizará al robot cuando colisione con las paredes. N_{coll} recogerá el número de veces que el robot se choque con las paredes, y se establece un umbral en treinta choques, de manera que cuando el robot supere o iguale dicho umbral, la función *fitness* será 0 independientemente del resto de parámetros. Por otro lado, el robot deberá realizar la acción de coger un paquete en alguna de las zonas grises y dejarlo en la zona negra al menos 5 veces, siendo $F_{Obj}/5$ igual a uno y no penalizando al robot, de otra manera se le penalizará cuanto más diste de 5 y anulándose la *fitness* cuando no realice esa acción ninguna vez. El factor V se maximiza cuando la rotación de las ruedas, independientemente de su sentido, es máximo, con el objetivo de favorecer el movimiento del robot. La componente $1 - \sqrt{\Delta v}$ se encarga de que ambas ruedas giren en el mismo sentido, tiendiendo a 1 este factor en caso de que así sea. En cuanto a la fórmula 3, lo que se pretende es maximizar que el robot vaya hacia adelante.

Los siguientes tres subcomportamientos tendrán un impacto menos radical en la *fitness*. En primer lugar, se tendrá en cuenta el valor de la batería roja, RBatt. Si su valor es mayor que el valor del umbral, 0,35, significa que tiene un nivel de batería suficiente, de manera que se revisará el valor del sensor de suelo con memoria, GM, si su valor es 0 significa que no lleva ningún objeto, de manera que se dirigirá a la luz azul posicionada sobre un espacio gris para recoger un objeto, si el valor de GM fuera 1 significaría que ha recogido un objeto y se dirigiría a la luz amarilla posicionada sobre una zona negra para dejarlo, en este caso se divide un factor 1,5 con el objetivo de que no caiga en el error de acercarse a la zona negra, dar vueltas e ir a cargarse, con este factor se logra que el robot entre de lleno en la zona negra debido a que la luz amarilla se posiciona en el borde superior derecho del mapa elegido. En caso de que el valor de RBatt fuera 0, necesitaría cargarse, y el robot se dirigiría hacia la luz roja, para este caso se le multiplica por un factor 2 ya que es de gran interés que los mejores robot sean aquellos que se carguen cuando tengan que hacerlo al menos, de esta manera se evitan comportamientos en los que al robot no le importa estar penalizado por no cargarse y se queda dando vueltas en una posición que minimiza esa penalización debido a las recompensas por cumplir con los otros subcomportamientos.

A continuación se mostrará la evolución del valor de *fitness* obtenido para tres casos principales:

- El individuo con mejor valor de *fitness* de cada generación (azul).
- El valor medio de *fitness* de los individuos de cada generación (rojo).
- El individuo con peor valor de *fitness* de cada generación (verde).

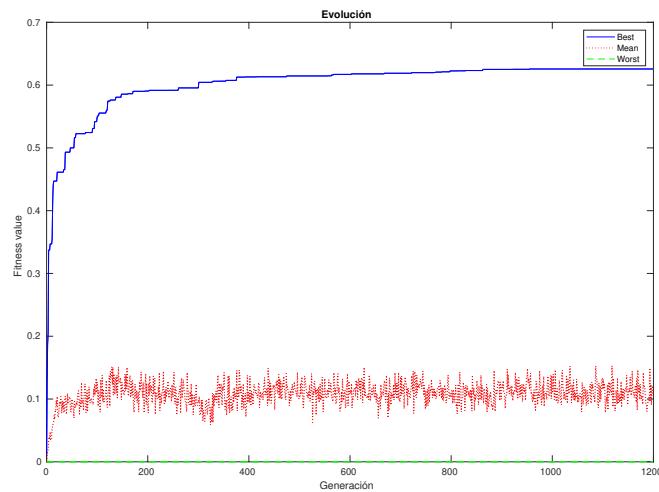


Figura 5: Evolución de la función fitness

5.2. Simulación del robot

En este apartado se analizará la conducta del robot utilizando el simulador IRSIM.

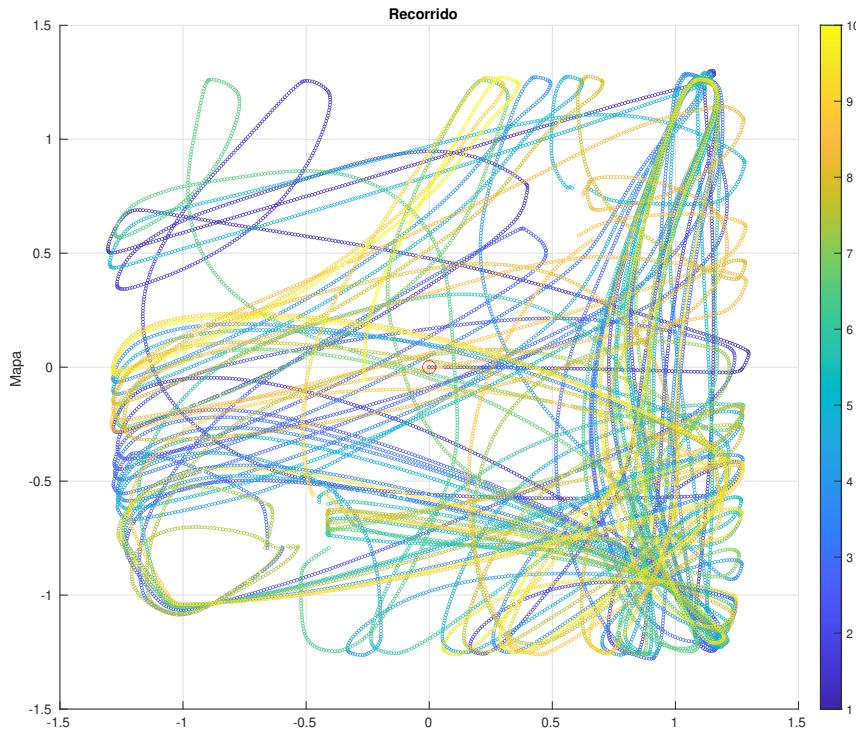


Figura 6: Recorrido del e_Puck

El círculo rojo situado en las coordenadas (0,0) indica el lugar de inicio del e_Puck en el mapa, y su recorrido se puede observar a partir del color de la línea que marca su trayectoria. La zona de color azul oscuro indica los primeros movimientos del robot, mientras que la zona de color amarillo claro indica los últimos movimientos del robot en la simulación. Caben destacar varias situaciones:

- No se aprecia movimiento del robot en los alrededores de las coordenadas $(-0,75, -0,75)$ y $(0,75, 0,75)$. Esto se debe a los obstáculos que se han situado, tal y como se ha podido apreciar en la figura 2.
- Se ha producido un movimiento de llegada y salida en los alrededores de las coordenadas $(-1, -1)$ debido a que se sitúa en sus proximidades la luz roja que va a permitir al robot cargar la batería. Se ha producido una llegada y salida en instantes espaciados de tiempo, ya que se observan líneas de color amarillo, verde, azul y naranja. Los instantes espaciados se deben a que el robot se acercará a la luz roja cuando se haya descargado, y el tiempo de descarga es el mismo en todo momento.
- Se observa un continuo movimiento del robot entre la esquina superior derecha y la esquina inferior derecha, esto se debe a que el robot se dedica la mayor parte del tiempo a ir hacia la zona gris, a entregar un paquete, y cuando lo haya entregado vuelve a la zona negra a recoger otro paquete para entregarlo de nuevo. También se observa un menor movimiento a la esquina superior izquierda, ya que se sitúa otra zona gris, en la simulación se ha tenido una clara preferencia por la primera zona gris.

5.2.1. Gráficas complementarias

Con la finalidad de ampliar los conocimientos acerca del comportamiento exacto del robot en cada momento, se muestran las entradas a las capas sensoriales, que mostrarán la información recibida por los sensores de proximidad (capa 0), sensor de suelo con memoria (capa 1), sensores de luz amarilla (capa 2) y azul (capa 3), sensor de batería roja (capa 4) y sensores de luz roja (capa 5).

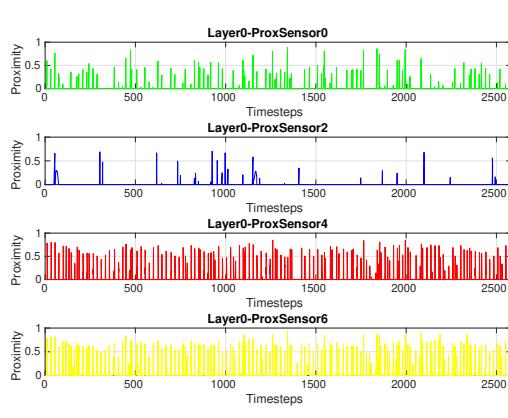


Figura 7: Sensores de proximidad

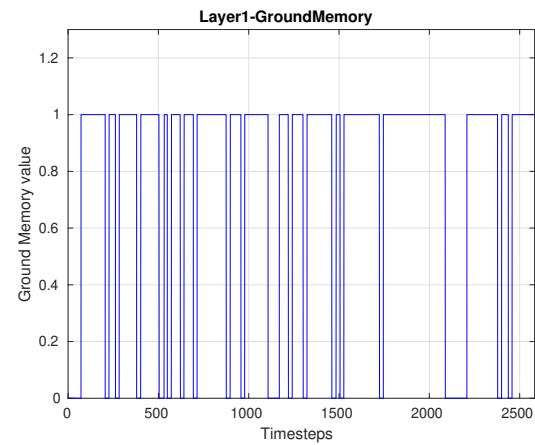


Figura 8: Sensor de suelo con memoria

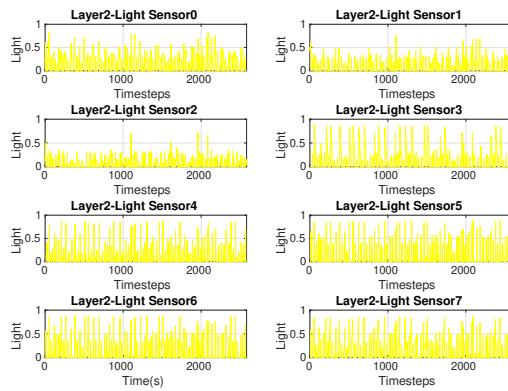


Figura 9: Sensores de luz amarilla

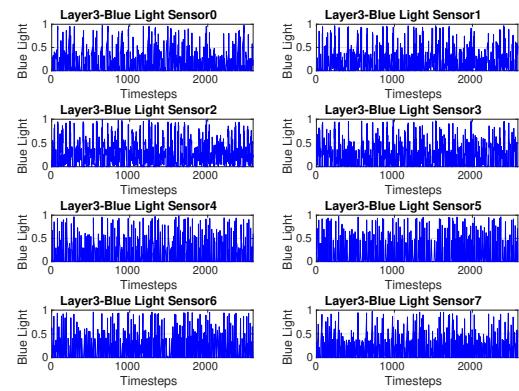
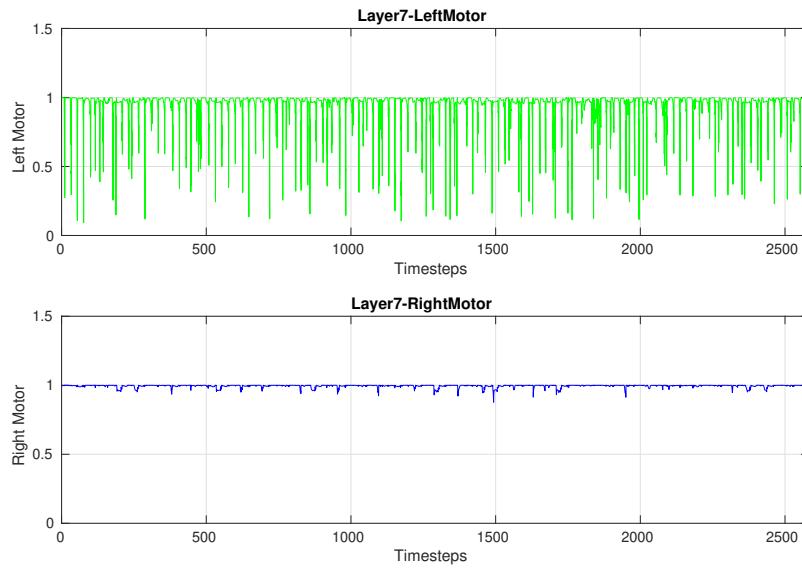
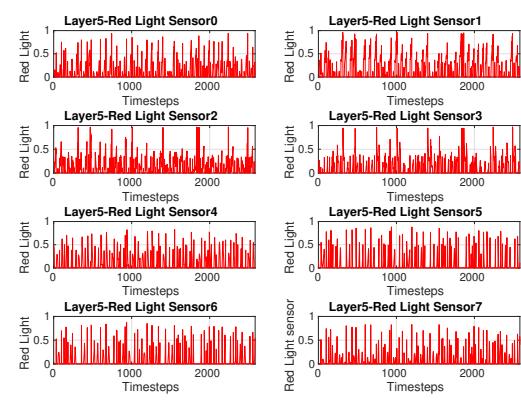
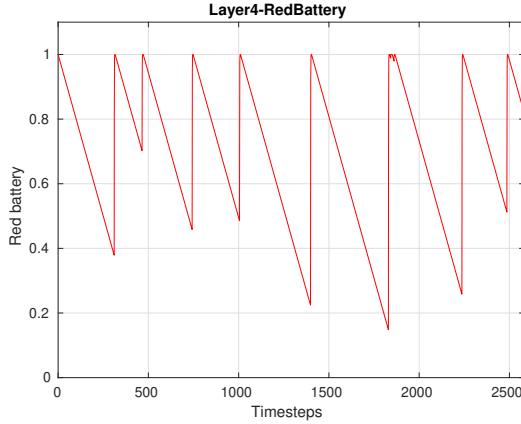


Figura 10: Sensores de luz azul



Respecto a las situaciones destacadas anteriormente, por un lado, se comprueba en la figura 8 un continuo cambio en el sensor de suelo con memoria, que explica el movimiento del robot entre la zona gris y la zona negra explicada anteriormente. Por otro lado, en la figura 11 se comprueban unos intervalos con duraciones similares en lo que se refiere al tiempo entre la descarga y la carga del robot, que explican los intervalos equiespaciados en los que el robot se dirigía hacia la luz roja, para cargar su batería. Por último, se aprecian picos más continuados en las figuras 9 y 10 respecto a la figura 12, esto se debe a que el robot debe recargar su batería menos frecuentemente y verifica que el robot se dedica la mayor parte de la simulación a entregar y recoger paquetes, es decir, a moverse entre las zonas negras y grises.

6. Redes neuronales artificiales dinámicas

A diferencia de la anterior, la red neuronal artificial dinámica tendrá la siguiente arquitectura:

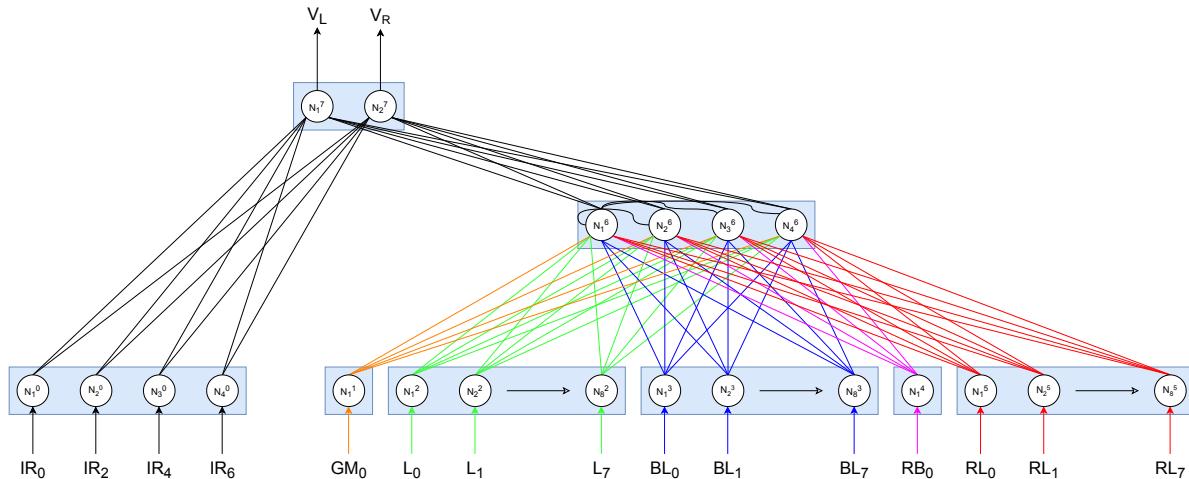


Figura 14: Arquitectura CTRNN

La principal diferencia con la arquitectura anterior es que las neuronas de la capa oculta están conectadas entre sí. Esto se traduce en que las redes neuronales estáticas no poseen memoria, solo transforman sus entradas en salidas, de manera que cada neurona depende únicamente de sus entradas y de los parámetros establecidos previamente. Las redes neuronales dinámicas permiten establecer una relación entre las salidas y entradas y/o las salidas y entradas previas.

En cuanto a la función *fitness*, ambas redes neuronales dispondrán de la misma función, con la finalidad de realizar una comparación lo más “justa” posible. La función *fitness* y su comportamiento deseado, que se han explicado anteriormente, no variarán, por lo que no se explicará de nuevo, pero sí se mostrará la evolución del valor de *fitness* obtenido para los tres casos principales, de la misma manera que en la arquitectura estática:

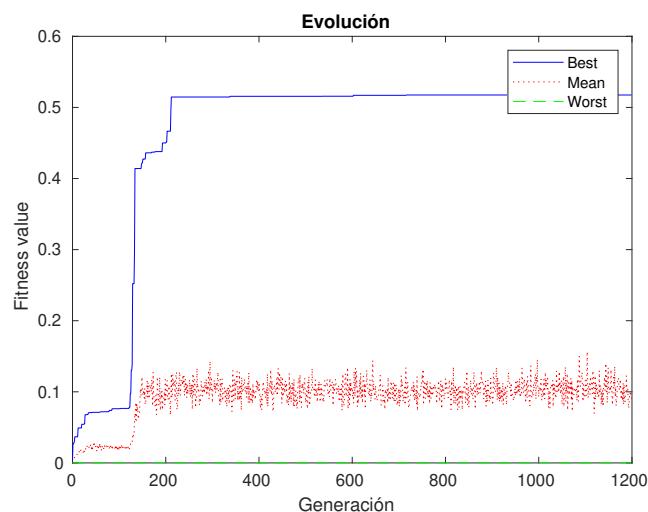


Figura 15: Evolución de la función fitness

Se destacan tres características respecto a la evolución de la *fitness* en la red estática:

- El individuo con mejor valor de *fitness* de cada generación (azul) presenta un valor menor de fitness, ya que se estabiliza alrededor de 0,5, a partir de la generación número 300 aproximadamente. Además presenta una evolución más lenta
- El valor medio de *fitness* de los individuos de cada generación (rojo) se mantiene en torno al valor 0,1.
- El individuo con peor valor de *fitness* de cada generación (verde) sigue en torno a 0 .

6.1. Simulación del robot

En este apartado se analizará la conducta del robot, utilizando la arquitectura dinámica, en el simulador IRSIM.

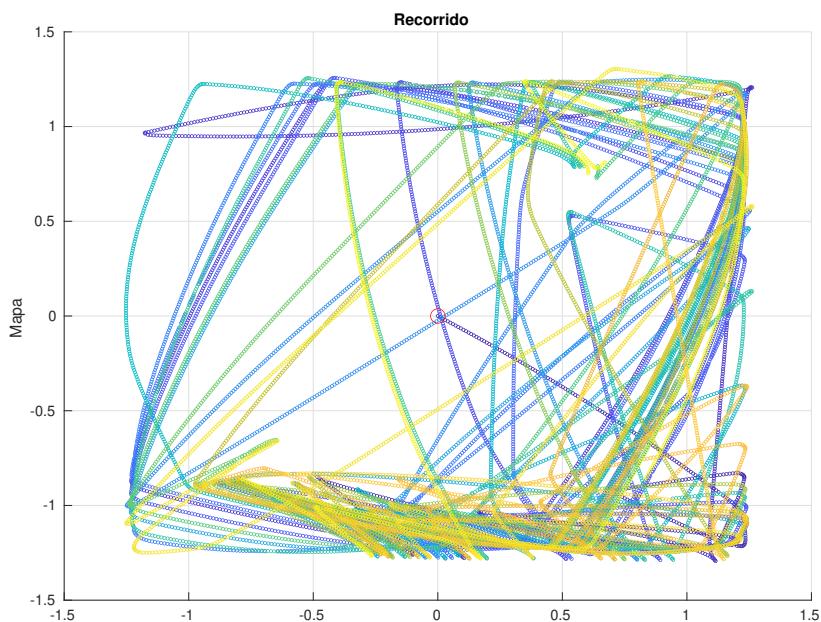


Figura 16: Recorrido del e_Puck

El recorrido del robot inicia en el círculo rojo y su recorrido a lo largo del tiempo lo marca el color de la línea, siendo el color azul oscuro los primeros movimientos del robot, y el amarillo claro los últimos movimientos a lo largo del mapa. Se van a destacar los siguientes sucesos:

- Cuando se detecta una pared u obstáculo, se evita cambiando de dirección de una manera brusca. Además, la mayoría de los desplazamientos de media y larga distancia son prácticamente rectos.
- Se observan dos áreas alrededor de las coordenadas $(-0,75, -0,75)$ y $(0,75, 0,75)$ por las que el robot no pasa, esto se debe a los obstáculos que se han puesto en el mapa.
- Se producen desplazamientos hacia la luz roja, situada en la esquina inferior izquierda, en intervalos de tiempo similares entre ellos, esto se debe a que el robot, cada cierto intervalo de tiempo necesitará cargar su batería ya que su nivel estará por debajo del umbral establecido.
- Se realizan movimientos entre la zona negra (esquina superior derecha) y las zonas grises (esquina superior izquierda y esquina inferior derecha) más equitativos que en la red estática, aunque se aprecia una clara preferencia a la zona gris de la esquina inferior derecha, como sucedía en la red neuronal estática.

6.1.1. Gráficas complementarias

De la misma manera que en la red estática, para conocer el comportamiento exacto del robot en cada momento, se enseñarán las entradas a las capas sensoriales, que mostrarán la información recibida por los sensores de proximidad (capa 0), sensor de suelo con memoria (capa 1), sensores de luz amarilla (capa 2) y azul (capa 3), sensor de batería roja (capa 4) y sensores de luz roja (capa 5).

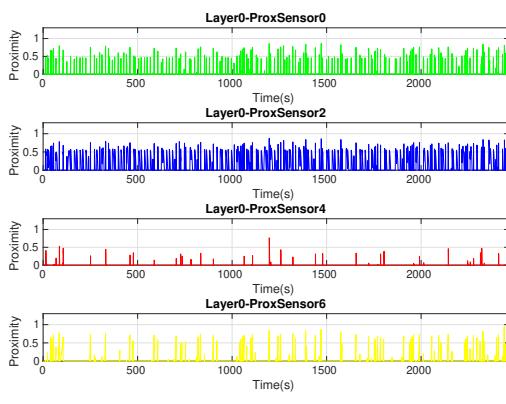


Figura 17: Sensores de proximidad

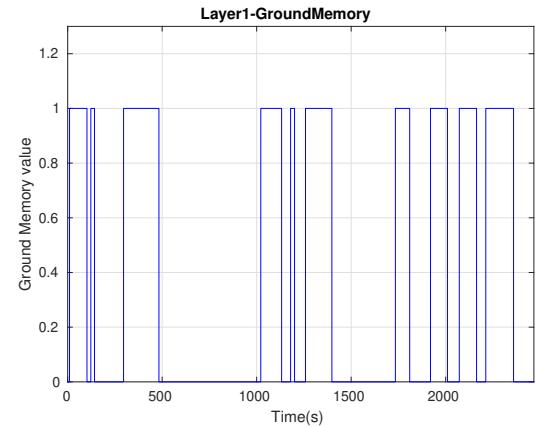


Figura 18: Sensor de suelo con memoria

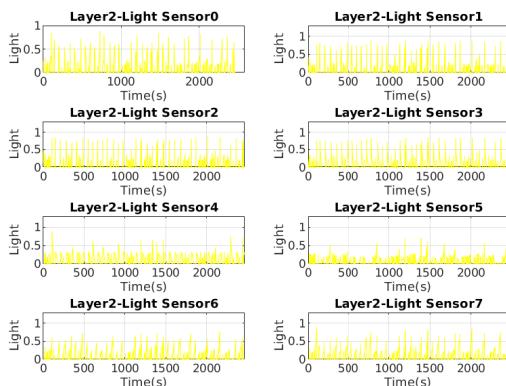


Figura 19: Sensores de luz amarilla

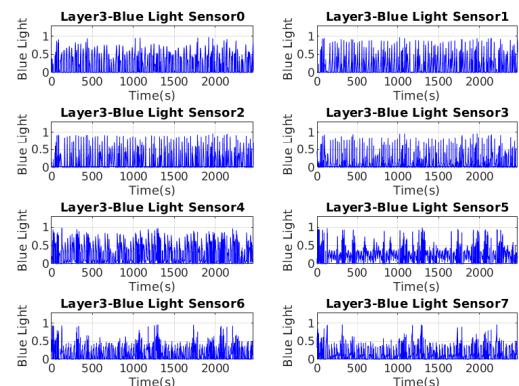


Figura 20: Sensores de luz azul

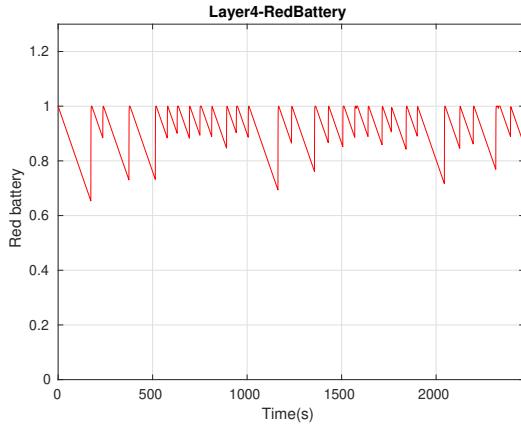


Figura 21: Sensor de batería roja

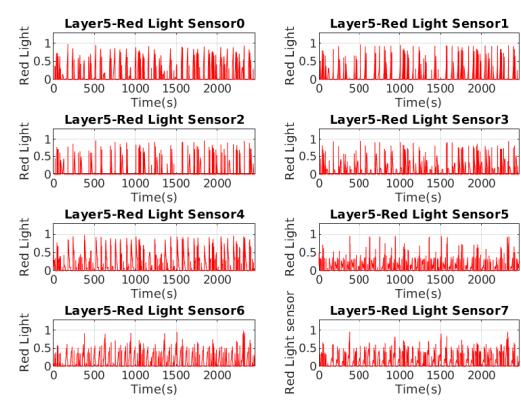


Figura 22: Sensores de luz roja

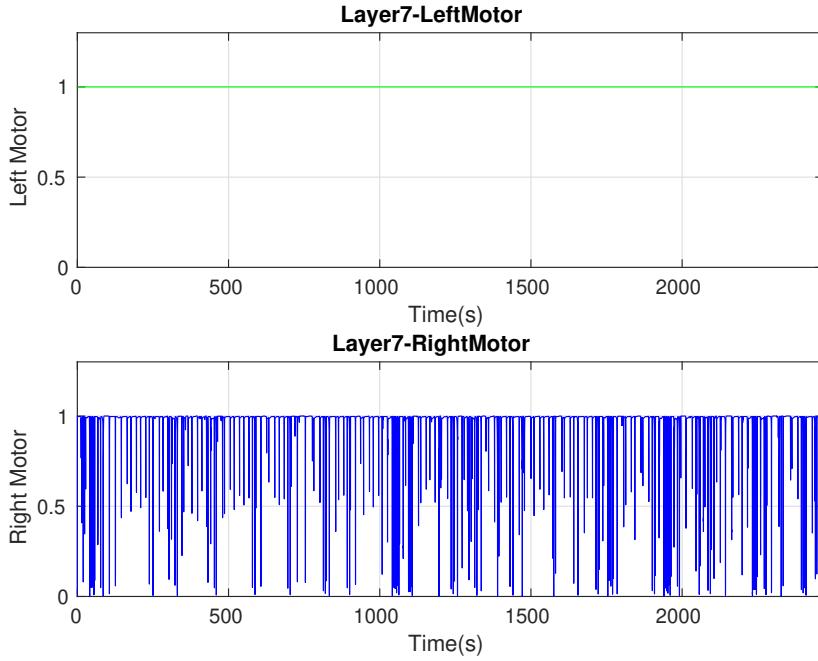


Figura 23: Capa motora

A partir de las gráficas se pueden comprobar los sucesos nombrados anteriormente. Por un lado, se observa el sensor de suelo con memoria, en la figura 18, que varía entre 0 y 1. Se debe a que el robot entra las distintas zonas (negra y gris) para dejar el paquete o recogerlo. Cabe destacar varios intervalos en los que se mantiene en 0 durante un largo periodo de tiempo, esto se produce ya que el robot debe cargar la batería en esos momentos y se mantiene cerca de la luz roja, tal y como se puede observar en la figura 21 (desde 500 hasta 1000 steps y desde 1400 a 1700 steps aproximadamente). Por otro lado, se aprecian picos más espaciados en la luz roja (figura 22) que en las luces amarillas (figura 19) y azules (figura 20), ya que el robot deberá ir hacia la luz roja con menos frecuencia que hacia la luz amarilla y azul, ya que la carga de la batería se produce cada ciertos intervalos de tiempo, como se puede ver en la figura 21.

7. Comparación de arquitecturas y conclusiones

Se pueden destacar varios aspectos que son claramente diferenciables entre ambas arquitecturas:

- En cuanto a la evolución de la *fitness* para la arquitectura ANN, figura 5, y para la arquitectura CTRNN, figura 15, se puede apreciar que para CTRNN el valor de la fitness del mejor individuo es menor que el valor del mejor individuo de la ANN en todas las generaciones, además, se puede ver que ambas estabilizan su valor de la fitness cerca de la generación 200. Por último, se aprecia como los cambios en los valores de la fitness del mejor individuo son más abruptos en CTRNN que en ANN, sirviendo como ejemplo la generación 160.
- En cuanto al recorrido para ambas arquitecturas, figura 6 y figura 16, respectivamente, se pueden apreciar en CTRNN trayectorias significativamente más rectas, y el robot da menos vueltas en la esquina inferior derecha que en ANN.
- Resulta interesante comparar también los ciclos de carga y descarga de la batería para ambas arquitecturas, figura 11 para ANN y figura 21 para CTRNN. Se puede apreciar que el valor de la batería para el caso de CTRNN oscila menos estando siempre entre el 65 % y el 100 %, mientras que para ANN entre el 18 % y el 100 %.
- Por último, se van a comparar la figura 13, que hace referencia a la capa motora de ANN y la figura 23 que hace referencia a la capa motora de la arquitectura CTRNN. Respecto al motor izquierdo se puede ver que se consigue un maximización constante de la velocidad de este para CTRNN mientras que en ANN hay ciertos instantes en los que esa velocidad baja. Además, respecto al motor derecho se puede ver que en la arquitectura ANN la maximización no es tan perfecta como para la rueda izquierda de CTRNN pero se mantiene cercado al valor máximo durante la mayoría del tiempo, sin embargo, con la rueda derecha en CTRNN se pueden apreciar esas variaciones abruptas de la velocidad. A la vista de este análisis se puede concluir que el motor izquierdo se encarga del giro en ANN con minimas ayudas del motor derecho y en CTRNN el motor derecho es el que es totalmente responsable del giro del robot.

A la vista de las comparaciones expuestas, se concluye que el robot con la arquitectura CTRNN cumple de mejor manera con la conducta deseada del robot, sin embargo, el valor de fitness del mejor individuo, es mayor en la arquitectura ANN que en la arquitectura CTRNN.

8. Problemas encontrados

- El objetivo era conseguir un reparto más equitativo en lo que se refiere a movimiento entre una zona gris y negra, pero en la simulación se ha tenido una clara preferencia por la zona gris de la esquina inferior derecha.
- Se han encontrado diversos problemas que han determinado el uso de los factores usados tanto en los sensores de luz amarilla, como en los de luz roja. Con estos factores se ha pretendido evitar que el robot realizara subcomportamientos sin ningún sentido, por ejemplo dar vueltas sobre la esquina que rodeaba la batería o dar vueltas en el centro del mapa.
- Conseguir un valor de fitness mayor cumpliendo con la conducta deseada.

9. Posibles implementaciones futuras

Aunque las opciones con el simulador son muy limitadas, es cierto que se podrían usar más sensores que aportaran información relevante para alguna tarea específica, o hacer que el robot fuera capaz de realizar la tarea de la manera deseada en un mapa con más obstáculos. Con más complejidad aún y de acuerdo con los contenidos teóricos explicados en la asignatura *Teoría de la información*, se podría implementar un aprendizaje por refuerzo en el que el simulador o el medio en el que se mueve el robot, modelado como un proceso de decisión de Markov, aportase una observación y un coste instantáneo sobre cada acción que el robot realizará, esto se implementaría en la parte de las redes dinámicas.

Referencias

- [1] IRIN. *Manual del simulador irsim*. Febrero, 2020
- [2] Adolfo Cursillo Ngua. Diseño de sensores software no lineales para la estimación de variables de calidad de los procesos. Febrero, 2019
- [3] Advanced Tech Computing Group UTPL. Clases de redes neuronales artificiales. Agosto 2007
- [4] Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot (D. Floreano and F. Mondada 1994)