Universidad Zaragoza

Trabajo Fin de Máster

Aprendizaje adaptativo basado en el mantenimiento de invarianzas organizacionales neuronales

Adaptive learning based on the maintenance of neural organizational invariances

Autor:

Víctor García Otin

Directores:

Miguel Aguilera y Manuel Bedia

ESCUELA DE INGENIERÍA Y ARQUITECTURA DE ZARAGOZA

2020

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe entregarse en la Secretaría de la EINA, dentro del plazo de depósito del TFG/TFM para su evaluación).

D./Dª. Victor Garcia Otin ,en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza, Declaro que el presente Trabajo de Fin de (Grado/Máster)

(Título del Trabajo)

Aprendizaje adaptativo basado en el mantenimiento de invarianzas organizacionales neuronales

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 16 de Noviembre

Fdo: Victor

RESUMEN

Durante las últimas décadas se están utilizando herramientas que vienen de la física para estudiar sistemas biológicos. En estos estudios se ha observado que los sistemas biológicos no se posicionan en una fase u otra, sino que suelen posicionarse en la transición de ambas para mejorar su capacidad de adaptación.

Un artículo de la revista *Scientific Reports* [1] propone un mecanismo de aprendizaje que genera comportamientos adaptativos, en entornos sencillos, buscando los puntos críticos, que son las transiciones de fase comentadas anteriormente.

La primera parte del trabajo consiste en acercar el funcionamiento del algoritmo al de los sistemas biológicos reales. Esto se consigue sustituyendo la red neuronal finita del mismo por una red infinita. En la segunda parte se replica el algoritmo con la nueva arquitectura y se comprueban sus resultados no solo en los mismos entornos, sino también en uno más complejo. Estos entornos son: Un robot móvil, un péndulo con una barra articulada y una serpiente que aprender a avanzar.

En todos los entornos se observa que, simplemente buscando estos puntos críticos, los agentes generan comportamientos interesantes sin haber sido programados para ello. Además, la complejidad de su comportamiento aumenta a medida que se aumenta el tamaño de las redes neuronales.

Índice general

Capítulo 1

INTRODUCCIÓN

1.1. Motivación del proyecto

Uno de los grandes desafíos científicos de nuestro tiempo es entender cómo el cerebro procesa la información. Los trabajos actuales sugieren que las neuronas individuales son relativamente modestas en su potencia de cálculo, pero que las redes neuronales pueden realizar colectivamente operaciones extremadamente complejas. Por lo tanto, cualquier sistema que pueda ayudar a clarificar la explicación de las propiedades emergentes [1] puede potencialmente avanzar en nuestra comprensión de la función neuronal.

Afortunadamente, las propiedades emergentes han recibido mucha atención en una amplia variedad de sistemas. Las simulaciones de "agentes en evolución"llevaron a *Packard* [17] a sugerir que la adaptabilidad se optimizaba cuando los agentes se acercaban al "borde del caos"[14]. *Langton* [11] estudió los autómatas celulares que podían corresponderse con regímenes ordenados, críticos o caóticos, y concluyó que los cálculos se realizaban mejor en sistemas cercanos al punto crítico entre el orden y el caos. *Kauffman* y su equipo [10] [9] modelaron las interacciones de las proteínas celulares con redes booleanas aleatorias y demostraron que, bajo la presión de la selección, estas redes se auto-organizarían hasta un estado casi crítico en el que las perturbaciones se propagarían en forma de avalanchas. Los modelos neuronales también han recogido esta idea, y varios autores han sugerido que las redes neuronales deberían funcionar cerca del régimen caótico [2] [5] [13]. En general, estas simulaciones predicen que los sistemas vivos se auto-organizarán para operar cerca de un punto crítico. Esta hipótesis es conocida cómo la "hipótesis de la criticidad".

[1] Una propiedad emergente es una propiedad que tiene una colección o un sistema complejo, pero que los miembros individuales no tienen.

A pesar de la abundancia de simulaciones, las pruebas experimentales que apoyan la hipótesis de la criticidad han sido relativamente escasas. Sin embargo, experimentos recientes han comenzado a apoyar esta idea en el área de las redes neuronales [4] [7].

Pese a estos recientes avances aun no se tiene la certeza de cómo surge el posicionamiento en el punto crítico. Para intentar dar respuesta a esta pregunta, en el artículo tomado cómo referencia [1] se diseña una regla de aprendizaje para tratar de posicionar varios sistemas en una transición de fase. El resultado que se espera obtener es que, tras lograr posicionar al agente en un punto crítico, este sea capaz de adaptarse a su entorno, generando comportamientos espontáneos y complejos, sin habérselos definido de forma explícita.

1.2. Fases del proyecto

Este proyecto se basa en un estudio realizado y publicado en la revista *Scientific Reports* [1]. En este proyecto se desarrolla un algoritmo cuya finalidad es adaptar las correlaciones entre neuronas para tratar de llevar a un agente a su punto crítico de funcionamiento. Esto se hace en dos pasos: Primero, se calcula, con una red neuronal del modelo de *Ising*, el nivel de correlaciones que se dan en la criticalidad y posteriormente, conociendo el valor de esta distribución de correlaciones se analiza el agente con el algoritmo de criticalidad. Este algoritmo asegura que el agente a evaluar escoge las acciones respetando siempre este tipo de distribución de correlaciones entre sus neuronas. Tras el entrenamiento de esta segunda red, se analiza el funcionamiento del agente y se observa si la red ha sido capaz de hacer que el agente esté en un punto crítico, y si este ha sido capaz de adaptarse a su entorno generando comportamientos espontáneos.

En el proyecto tomado cómo modelo, se han realizado pruebas de la red neuronal finita de 20 x 20 controlando varios agentes en sus respectivos entornos entornos. Sin embargo, en nuestro proyecto se ha sustituido esta red finita por una red infinita. Para ser exactos, se ha sustituido por una red infinita y posteriormente se ha cogido una cuadricula de 20 x 20 de la misma. Con esto se consigue evitar los "efectos de borde.º dicho de otra manera, se consigue que las neuronas de la ultima capa estén calculadas teniendo en cuenta que alrededor suya también hay neuronas.

Más específicamente las razones de este cambio son principalmente dos: El primero es que si

se lo logra demostrar que el funcionamiento con una red infinita es similar al de la red finita propuesta en el articulo, ya no haría falta calcular la distribución de correlaciones de la red neuronal del modelo de *Ising* en 2D, sino que se podría hacer todo con la formula teórica que se presentará más adelante. Esto supondría un salto cualitativo para la realización de otros experimentos ya que el cálculo de esta red tiene un coste computacional muy alto. La segunda razón es que la red infinita es una aproximación mucho más realista al funcionamiento real de los sistemas biológicos.

Los entornos y los experimentos se han llevado a cabo utilizando la libreria y web de codigo abierto *OpenAI* [6]. Este simulador tiene cómo objetivo facilitar la investigación y el desarrollo de la robótica, la biomecánica, los gráficos y la animación, y otras áreas en las que se necesita una simulación rápida y precisa. Ofrece una combinación única de velocidad, precisión y potencia de modelado.

Los primeros entornos en ser probados son el *Mountain Car* que es un robot móvil y el *Acrobot* que es una barra biarticulada. Al ser dos entornos que también se han calculado en el artículo de referencia nos permitirá comparar claramente el efecto del cambio de red neuronal. Una vez analizado el comportamiento de estos entornos se pasa a probar un entorno más complejo: El *Swimmer*. Este entorno pertenece a una sub-librería de la plataforma *OpenAI* y se llama *MuJoCo* [19]. Esta libreria contiene entornos más complejos, cómo es el caso del *Swimmer*.

Sin embargo, esta plataforma tiene la gran desventaja de que en el sistema operativo *Windows* no funciona, o lo hace con muchas restricciones. Es por eso que para la realización de esta parte fue necesario la instalación y el aprendizaje del sistema operativo *Ubuntu*. *Ubuntu* es una distribución *Linux* basada en *Debian* y compuesta en su mayoría por software libre y de código abierto.

Una vez nos habíamos familiarizado con su funcionamiento, empezamos a hacer las modificaciones oportunas en el algoritmo para adaptarlo al nuevo entorno. Tras estas modificaciones en el código, evaluamos si el algoritmo de criticalidad es capaz de llevar al agente al punto crítico.

Esto se puede comprobar de diferentes formas cómo por ejemplo comparando sus cambios de

posición y movimientos ó calculando la capacidad calorífica del sistema [2]. Una divergencia de la capacidad calorífica del sistema es condición suficiente que indica que el sistema se encuentra en un punto crítico. La capacidad calorífica se calcula a través de la entropía, lo cual tiene un coste computacional muy elevado.

En el proyecto se propondrá un método alternativo para el cálculo de la capacidad calorífica. En lugar de calcularla a partir de la entropía, se hará a partir de la energía del sistema. Se esperan obtener los mismos resultados que con el cálculo a partir de la entropía, pero con la ventaja de que tiene un coste computacional menor.

[2]La capacidad calorífica es la cantidad de calor que hay que suministrar a la unidad de masa de una sustancia o sistema termodinámico para elevar su temperatura en una unidad

Capítulo 2

CONCEPTOS

2.1. Machine Learning

Los métodos de aprendizaje automático (*Machine learning*) son generalmente métodos flexibles y no paramétricos para hacer predicciones o clasificaciones a partir de los datos. Estos métodos se describen típicamente por el algoritmo que detalla cómo se hacen las predicciones utilizando los datos en bruto y puede permitir un mayor número de predictores, lo que se denomina datos de alta dimensión. A menudo estos métodos pueden detectar automáticamente las no linealidades en las relaciones entre las variables independientes y dependientes y pueden identificar las interacciones automáticamente. Estos métodos pueden aplicarse para predecir resultados continuos, lo que se conoce generalmente cómo problemas de tipo regresivo, o para predecir los niveles de una variable categórica, lo que se conoce generalmente cómo problemas de clasificación. Los métodos de aprendizaje automático también pueden utilizarse para agrupar casos sobre la base de un conjunto de variables conocidas para todos los casos.

El aprendizaje supervisado (*Supervised learning*) utiliza datos de entrenamiento etiquetados para aprender la función de mapeo que convierte las variables de entrada (X) en la variable de salida (Y). Esto nos permite generar con precisión las salidas cuando se dan nuevas entradas. Principalmente podemos hablar de dos tipos de aprendizaje supervisado: clasificación y regresión.

Por otro lado, los modelos de aprendizaje no supervisado (*Unsupervised learning*) se utilizan cuando sólo tenemos las variables de entrada (X) y ninguna variable de salida correspondiente. Utilizan datos de aprendizaje no supervisado para modelar la estructura subyacente

de los datos.

Por ultimo, los algoritmos de aprendizaje por refuerzo (*Reinforcement learning*) permiten a un agente decidir la mejor acción siguiente en función de su estado actual mediante el aprendizaje de comportamientos que maximizarán una función de recompensa.

Los algoritmos de refuerzo normalmente aprenden las acciones óptimas mediante prueba y error. Un ejemplo de esto sería un videojuego en el que el jugador necesita moverse a ciertos lugares en ciertos momentos para ganar puntos. Un algoritmo de refuerzo que jugara ese juego comenzaría moviéndose al azar pero, con el tiempo, a través de un mecanismo de prueba y error, aprendería dónde y cuándo necesita mover al personaje del juego para maximizar su total de puntos.

A diferencia de los ejemplos anteriores, en este trabajo utilizaremos un algoritmo que no tiene cómo objetivo conseguir el mayor numero de puntos, sino alcanzar una distribución especifica de correlaciones. Esta distribución de correlaciones es precisamente la que se da en el punto crítico, que es donde se maximizan las fluctuaciones de energía. Por tanto, lo que tenemos en realidad es implícitamente una red neuronal basada en energía.

2.2. Redes neuronales basadas en energía

El principal objetivo de la modelización estadística y el aprendizaje automático es codificar las dependencias entre las variables. Al capturar esas dependencias, un modelo puede utilizarse para responder a las preguntas sobre los valores de las variables desconocidas dados los valores de las variables conocidas.

Los Modelos Basados en la Energía [12] captan las dependencias asociando una energía escalar a cada configuración de las variables. La inferencia, es decir, la realización de una predicción o decisión, consiste en fijar el valor de las variables observadas y encontrar valores de las restantes variables que minimicen la energía. El aprendizaje consiste en encontrar una función de energía que asocia energías bajas a valores correctos de las variables restantes, y energías más altas a valores incorrectos. Una función de pérdida, minimizada durante el aprendizaje, se utiliza para medir la calidad de las funciones de energía disponibles. Dentro de este marco común de inferencia/aprendizaje, la amplia elección de funciones energéticas y de pérdida funcional permite el diseño de muchos tipos de modelos estadísticos, tanto

probabilísticos cómo no probabilísticos.

El aprendizaje basado en energía proporciona un marco unificado para muchos enfoques probabilísticos y no probabilísticos del aprendizaje, en particular para la formación no probabilística de modelos gráficos y otros modelos estructurados. El aprendizaje basado en energía puede considerarse una alternativa a la estimación probabilística para las tareas de predicción, clasificación o toma de decisiones. Dado que no se requiere una normalización adecuada, los enfoques basados en energía evitan los problemas relacionados con la estimación de la constante de normalización en los modelos probabilísticos. Además, la ausencia de la condición de normalización permite mucha más flexibilidad en el diseño de las máquinas de aprendizaje. La mayoría de los modelos probabilísticos pueden considerarse cómo tipos especiales de modelos basados en energía en los que la función de energía satisface ciertas condiciones de normalización, y en los que la función de pérdida, optimizada por el aprendizaje, tiene una forma particular.

2.3. Modelo de Ising ó Máquina de Boltzmann

El modelo de *Ising* [18] en física ó la máquina de *Boltzman* en informática es un sistema matemático utilizado principalmente para el estudio de las transiciones de fase. Está compuesto por elementos que pueden tomar el valor de -1 o +1. Para explicar el modelo se toma el ejemplo de un sistema magnético. El -1 indica que cierto elemento está apuntando hacia abajo y el +1 indica que dicho elemento está apuntando hacia arriba. En este modelo, también se contempla la existencia de un campo externo, descrito por la variable h. La energía del sistema se describe a través de la siguiente ecuación:

$$E = \sum_{i<j} J s_i s_j + \sum_i h s_i \tag{2.1}$$

Dónde:

J es el factor de escala entre interacción entre espines y energía.
s_i es el valor del espin del elemento *i-ésimo*.

Atendiendo a lo anterior, y suponiendo que los elementos sólo afectan a los elementos "vecinos", se obtiene que, cuando los espines de dos elementos apuntan en la misma dirección,

la energía es $-J$ (sin tener en cuenta el campo externo). Por lo tanto, en el momento que los espines apuntan en la misma dirección, la energía del sistema disminuye. De acuerdo con lo anterior, podría esperarse que la tendencia fuera que todos los espines apuntaran en la misma dirección ya que esto minimizaría la energía del sistema. A pesar de lo anterior, hay que tener en cuenta que, según la termodinámica, en estas condiciones, el equilibrio no vendrá dado por el estado de mínima energía, sino por el estado de mínima energía libre de *Helmoltz*. La energía libre de *Helmoltz* se define cómo:

$$F = U - TS \tag{2.2}$$

Dónde:

U es la energía interna.
T es la temperatura.
S es la entropía

Debido a la ecuación anterior, el mínimo de energía libre depende de una relación entre el producto TS y la energía interna. La energía interna del sistema es menor en el momento que todos los espines apuntan en la misma dirección. En esta situación, la entropía alcanza su mínimo, ya que es el estado más "ordenado" posible. Si la temperatura es baja, el producto TS será bajo, y por lo tanto el término que tendrá mayor influencia en el sistema será la energía interna, así que el sistema alcanzará el equilibrio cuando todos los espines apunten en la misma dirección.

En caso de que la temperatura aumente, el término que tendrá más influencia en la ecuación será el producto TS. Por lo tanto, el equilibrio se alcanzará cuando la entropía aumente, haciendo que la energía libre de *Helmoltz* se minimice. El modelo de *Ising* tiene dos fases bien diferenciadas, una gobernada por la entropía, y la otra gobernada por la energía interna. La magnitud que define el régimen en el que se encuentra el sistema es la temperatura. En este punto aparece una transición de fase, o punto crítico, a la temperatura crítica. La temperatura crítica (T_c) es tal que si $T > T_c$, el sistema se encontrará en la fase dominada por la entropía, y si $T < T_c$, el sistema se encontrará en la fase dominada por la energía interna.

En el artículo de referencia para este proyecto [1], se selecciona un modelo de *Ising 2D* sin acción de campos externos (i.e. h = 0) que se encuentra en el punto crítico. El punto

crítico se alcanza, para estas condiciones, cuando el valor de J es igual a $log(1+\sqrt{2})/2\beta$.

Dónde:
$\beta = 1/(TK_b)$
T, es la temperatura del sistema
K_b = Constante de *Boltzmann*.

Se toman estas condiciones para el modelo debido a que es una de las pocas situaciones en la que se puede obtener una solución analítica. Para simplificar, el término β se toma cómo 1. Esto es posible porque el parámetro β se puede definir de forma arbitraria, ya que será el punto de referencia. Es decir, en este modelo, el valor de temperatura que haga que $\beta = 1$ será la temperatura crítica. A temperaturas bajas todos los espines apuntan en la misma dirección mientras que a temperaturas altas el sistema se encuentra lo más "desordenado" posible.

2.4. Dinámica de Glauber

La dinámica de *Glauber* permite describir los procesos físicos en los que estamos interesados. Esta dinámica es suficiente para describir la dinámica de los espines. Hemos realizado la aproximación de considerar que el orden en la sub-red y el parámetro de orden magnetización-ocupación de la sub-red es una variable no conservable.

En el algoritmo o dinámica de *Glauber* [3], para cada configuración dada, los N espines de la red tienen la misma probabilidad de ser seleccionados. De este modo se escoge una casilla aleatoriamente y se calcula el incremento de energía $\bigtriangledown E$ necesario para cambiar el espín en esa casilla.

El algoritmo de *Glauber* tiene una aplicación específica para modelos tipo *Ising*. Cómo hemos comentado anteriormente en la sección 2.3 hemos considerado el modelo de *Ising*, con la interacción entre primeros vecinos pero sin considerar el efecto del campo externo. En el algoritmo de *Glauber* el estado del espín S_i se obtiene al azar de manera independiente del estado del espín en el instante anterior. Por otra parte, en el caso en que $\bigtriangledown E < 0$, la probabilidad de transición utilizando este algoritmo no es necesariamente 1. Estas características permiten una interpretación física del algoritmo cómo un proceso dinámico. En consecuencia, al modelo de *Ising* dotado de esta dinámica se lo suele denominar modelo de *Ising* cinético.

Este algoritmo fue propuesto en un principio cómo un modelo físico de la transición al equilibrio del modelo de *Ising*. Este induce transiciones aleatorias entre los espines, las cuales dependen de la temperatura y del campo local instantáneo. Podemos estimar el tiempo medio entre dos saltos sucesivos de un espín de varios ordenes de magnitud mayor que las escalas temporales de las vibraciones en un sólido.

En nuestro trabajo es especialmente importante cómo se actualiza el estado de cada elemento con esta dinámica de *Glauber*. Esto se hace generando una dinámica en el sistema de acuerdo con la distribución de energía del mismo:

$$P(s_i(t+1)) = \left[1 + e^{-\beta 2 E_i s_i(t+1)}\right]^{-1} \tag{2.3}$$

Siendo la energía E la mencionada en la ecuación 2.4. El estado del modelo se actualiza aplicando esta dinámica en cada paso de la simulación a todos los elementos, en un orden aleatorio. La energía que necesita una neurona para cambiar su signo es $2E_i s_i(t+1)$. Tras realizar la simulación, el modelo de *Ising* con estas características alcanzará el equilibrio en un estado de máxima entropía:

$$P(s) = \frac{1}{Z} exp[\beta(\sum_i h_i s_i + \sum_{i<j} J_i j s_i s_j)] \tag{2.4}$$

Esta distribución sigue una tendencia exponencial $P(s) = \frac{1}{Z} e^{-\beta E(s)}$, dónde Z es el valor de normalización y el tipo de red que se utilice determinará la distribución de correlaciones del modelo.

Capítulo 3

MODELO DE REFERENCIA

Con la finalidad de entender de donde vienen los nuevos resultados y aportaciones, se va a empezar describiendo el modelo del articulo de referencia [1] y los resultados de los que partimos para el desarrollo del proyecto. La principal característica de este modelo es que, con unas condiciones determinadas, presenta una transición de fase que tiene solución analítica. Para calcular esta solución analítica, el articulo en el que nos basamos propone una red de *Ising* tipo *Lattice* (cuadricula) 20 x 20.

3.1. Generar correlaciones

Nuestro objetivo es reproducir las correlaciones de un sistema que se encuentra en un punto crítico para así poder posicionar a otro sistema distinto en un punto crítico similar. Esto se conoce cómo la "clase de la universalidad", ya que en la criticalidad muchos sistemas tienen propiedades similares.

Posteriormente, se selecciona un modelo de *Ising* bidimensional 20 x 20 y se simula el funcionamiento del modelo actualizando el estado de cada elemento con la dinámica de *Glauber* descripta anteriormente. En los casos de redes bidimensionales, es conocido que las distribuciones de correlaciones siguen la función asintótica $c(r) \propto 1/r^n$, dónde $\eta = 1/4^2$ y r es la distancia entre unidades. Debido a que el modelo está lejos del límite termodinámico, en lugar de utilizar la función descrita anteriormente, se calculan las correlaciones sobre el modelo descrito. Durante la simulación de la red, se han calculado tanto las correlaciones, C , cómo la media, m . Debido a que los campos h de todas las unidades son igual a cero, la media de todos los elementos también será igual a 0. La distribución de las correlaciones obtenida es la siguiente:

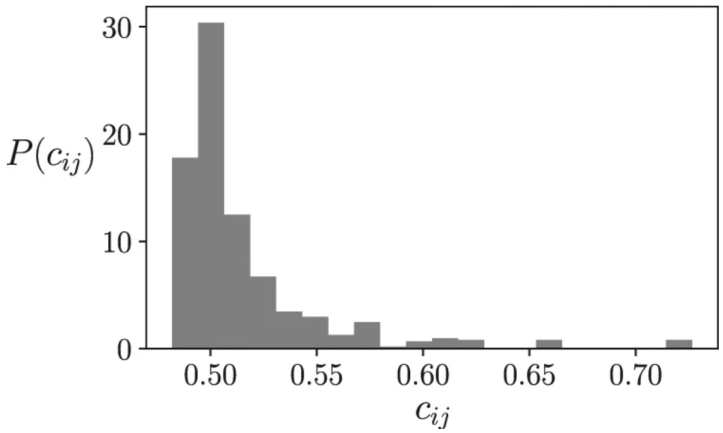

Figura 3.1: Distribución de correlaciones obtenidas en el modelo teórico. La imagen se ha obtenido del artículo de referencia [1].

3.2. Entrenamiento de la red neuronal

Una vez se obtiene la distribución de correlaciones en un modelo teórico posicionado en un punto crítico, se procede a entrenar la red neuronal que gobierna el comportamiento de cada agente. A cada neurona se le asigna un valor de c aleatoriamente, siguiendo la distribución obtenida anteriormente. En cada paso de la simulación, se calculará la correlación real de cada neurona con sus neuronas vecinas. A continuación, se tratará de obtener h y J a partir de las correlaciones obtenidas. Este proceso es conocido cómo el modelo de *Ising* inverso. Dicho modelo se resuelve con una regla de descenso del gradiente. Esta, junto con la distribución de correlaciones, es la regla de aprendizaje del algoritmo:

$$h_i \to h_i + \mu(m_i^* - m_i^m) \tag{3.1}$$

$$J_{ji} \to J_{ji} + \mu(c_{ij}^* - c_{ij}^m) \tag{3.2}$$

Dónde:

μ: Tasa de aprendizaje constante. El valor se toma cómo 0.01.
m_i^*: Valor de la media de referencia.
m_i^m : Valor de la media real.
c_{ij}^* : Correlación de referencia.

c_{ij}^m : Correlación real.

Para una simulación de un tiempo T el cálculo de la media se realiza con la expresión $m_i = \frac{\sum_t^T s_i}{T}$ y el cálculo de las correlaciones se realiza con la expresión: $c_{ij} = \frac{\sum_t^T (s_i s_j)}{T}$. Dado que cada paso de la simulación es muy costoso computacionalmente debido a que hay que calcular todos los estados de s, se toma un método alternativo. Se calculan las correlaciones aproximadas con la ecuación de la dinámica de *Glauber*. Tras realizar esta simulación, se obtienen los valores asociados a cada neurona de la red neuronal del modelo. Los siguientes apartados del proyecto se centrarán en validar si esta red es capaz de llevar a los agentes a puntos críticos, y de observar si la red es capaz de hacer que los agentes se adapten a los entornos, consiguiendo los distintos objetivos.

Para comprobar esto se utiliza la capacidad calorífica [1]. Una condición suficiente que indique que el sistema se encuentra cerca de un punto crítico es una divergencia en la capacidad calorífica, C. Además, esta magnitud es un indicador de la complejidad de la red neuronal. La capacidad calorífica del sistema se define cómo:

$$C(\beta) = -\beta \frac{\partial H}{\partial \beta} = \beta^2 (E^2(s)) - (E(s))^2) \tag{3.3}$$

Dónde:

$$E(s) = -\sum_i h_i s_i - \sum_{i<j} J_{ij} s_i s_j \tag{3.4}$$

Debido a la definición de la capacidad calorífica, si se detecta que la entropía presenta un pico continuo, en el que su derivada (capacidad calorífica) diverja tal cómo aumente el tamaño del sistema, se garantizará que el sistema está en un punto crítico. En el modelo de *Ising*, la entropía del sistema es: $H = \sum_s P(s) log(P(s))$. Por lo tanto, se calculará en diferentes puntos, con diferentes tamaños, para concluir si el sistema se encuentra en un punto crítico.

[1] La capacidad calorífica es una magnitud física que se define cómo la cantidad de calor que hay que suministrar a la unidad de masa de una sustancia o sistema termodinámico para elevar su temperatura en una unidad

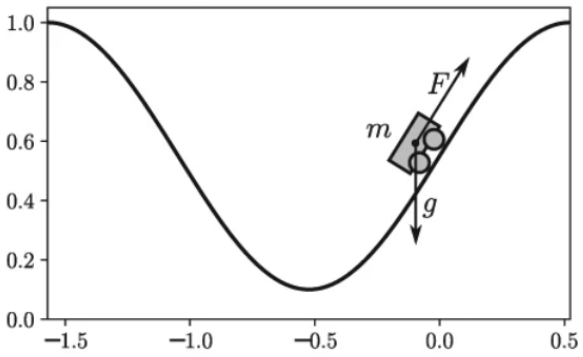

Figura 3.2: Descripción gráfica del modelo *Mountain Car*. La imagen se ha obtenido del artículo de referencia [1].

3.3. Conclusiones del articulo

El entorno *Mountain Car* 3.2 está formado por dos montañas y un valle. En este entorno hay un coche de masa m que inicialmente se encuentra en el valle y que se mueve en este entorno bidimensional. La parte problemática es que el motor no tiene suficiente fuerza para vencer la gravedad y subir la montaña. Por lo tanto, el agente debe aprender a aplicar la acción del motor, la cual puede tomar los valores -1, 0 ó 1. En cada instante temporal el coche tendrá que aprender cual es la acción más beneficiosa para poder subir la montana aprovechando la inercia del movimiento.

La red neuronal que controla el agente está formada por 4 neuronas sensitivas, 2 motoras y un número variable de neuronas ocultas (N_h , *hidden neurons*). El tamaño de la red es igual a: $N = N_m + N_s + N_h = 2 + 4 + N_h = 6 + N_h$. Los sensores recibirán un *input* externo, comparable al campo externo $(h_i = I_i)$. El valor de dicho input se definirá en función del estado del sistema. Las neuronas motoras definen la acción que realizarán los agentes. En el caso del coche, definen si el motor ejercerá fuerza hacia delante, hacia atrás, o si no ejercerá fuerza. El modelo de red neuronal que se escogió es el siguiente: Los sensores y las neuronas motoras únicamente estarán conectadas a las neuronas ocultas, y las neuronas ocultas estarán conectadas al resto de neuronas del sistema.

A pesar de que la distribución de la red no es crítica para el modelo, se escogió esta debido a que es una que está ampliamente extendida y permite conexiones recurrentes entre neuronas ocultas. En este entorno se entrenaron 10 agentes, realizando 1000 iteraciones con $T = 5000$ para redes de diferentes tamaños: 7, 8, 10, 14, 22, 38 y 70. El número de sensores y neuronas motoras se mantuvo para cualquier tamaño de la red. Las redes se entrena-

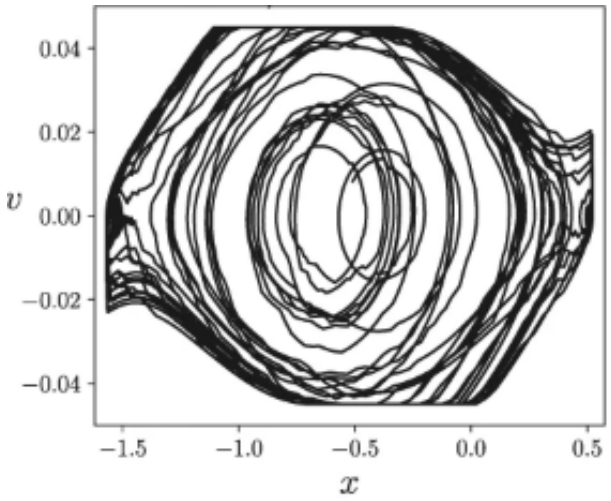

Figura 3.3: Velocidad frente a la posición del *Mountain Car* con 64 neuronas ocultas y $\beta = 1$. La imagen se ha obtenido del artículo de referencia [1].

ron siguiendo el método explicado anteriormente. El primer punto del análisis consistió en observar la distribución de las correlaciones obtenidas y comprobar que eran iguales a las mencionadas anteriormente.

A continuación, se realizó una simulación del comportamiento del agente en el entorno. Se observa que el agente es capaz de subir la montaña y adquirir velocidad aumentando la energía del sistema 3.3. Esto es un indicador de que el sistema se está adaptando al entorno y probablemente se encuentre en un punto crítico.

Cabe destacar que no se espera que la red consiga hacer que cualquier agente consiga sus objetivos en cualquier entorno. Por ejemplo, un entorno en el que el objetivo sea mantener el equilibrio (mantener una persona de pie, mantener una barra en vertical moviendo horizontalmente su base...) probablemente no sea adecuado para esta red.

Capítulo 4

MODELO PROPUESTO

4.1. Cálculo de la red neural infinita

Para el calculo de la red necesitamos calcular cómo son el tipo de correlaciones que se dan en este tipo de redes. Para esto, asumimos que la longitud de las correlaciones asociadas con el orden de los parámetros diverge en el punto crítico. Pero por supuesto, esperamos que ambas definiciones sean completamente equivalentes. Para discutir este punto de vista utilizamos el lenguaje del Modelo de *Ising*, dado por la energía en el cual la suma se realiza sobre los pares de los vecinos cercanos [18].

En el punto crítico se satisface que la energía es igual a 0 ($H = 0$) y la temperatura coincide con la temperatura critica $T = T_c$. La ecuación de correlaciones 4.1 expresa un comportamiento asintótico.

$$C(r) \rightarrow B_d r^{-(d-2+\eta)} \tag{4.1}$$

Esta es la función de correlaciones entre neuronas en una red infinita [18], en la que B_d es un factor no universal y $\eta = 1/4$ para todas las neuronas cuando estamos en una red bidimensional $d = 2$, cómo es nuestro caso.

El valor de B_d lo calculamos nosotros mismos a partir de las consideraciones del articulo [3]. B_d representa la covarianza del sistema, la cual sabemos que es igual a la correlación del mismo menos la media al cuadrado. Estos valores de la correlación han sido definidos en el articulo anterior y sabemos que la energía interna del sistema representa las correlaciones y la media se refiere a la magnetización. Por tanto la fórmula que computamos es:

TABLE I. Thermodynamic limits for the 2D lattice Ising model: here $\kappa \equiv 2(\sinh 2\beta/\cosh^2 2\beta)$.

Critical inverse temperature	$\beta_c = \frac{1}{2}\log\left(1+\sqrt{2}\right)$
Magnetization	$\mathcal{M} = \pm(1-\sinh^{-4}2\beta)^{1/8}$ for $T < T_c$, $\mathcal{M} = 0$ for $T \geq T_c$
Free energy	$-2\beta\mathcal{F} = \log(2\cosh^2 2\beta) + (2/\pi)\int_0^{\pi/2}\log\left(1+\sqrt{1-\kappa^2\sin^2\theta}\right)d\theta$
Internal energy	$-\mathcal{U} = \coth 2\beta\left[1+(2/\pi)(\kappa\sinh 2\beta - 1)\int_0^{\pi/2}(d\theta/\sqrt{1-\kappa^2\sin^2\theta})\right]$

Figura 4.1: Limites termodinámicos para una 2D *Lattice net* del Modelo de *Ising* [3].

$B_d = 0{,}5U - M^2$. Los valores de U y M los obtenemos de la figura 4.1. El factor de 0,5 que hemos incluido se debe a lo siguiente: Cuando calculamos la energía interna del sistema, cada espin de *Ising* se multiplica por todos sus vecinos y después sumamos todos esos pares. Cómo estamos sumando pares de espines en vez de espines, estamos sumando cada espin dos veces y por consiguiente tenemos que dividir la energía interna entre dos, dando así lugar a ese factor 0,5. Cabe destacar que tanto el factor U cómo el factor M solamente dependen de β, y por lo tanto B_d y C_{ij} serán solamente función de β. Para calcular las correlaciones de la red en el punto crítico, simulamos la red con $\beta = \beta_c$. Este calculo de las correlaciones nos sirve para indicarle al agente las correlaciones necesarias para situarse en el punto crítico. De esta forma inicializaremos las correlaciones de las neuronas siguiendo esta distribución obtenida.

4.2. Evaluación de la red neuronal infinita

El problema de la anterior red bidimensional es que la criticalidad es un fenómeno que se da en redes de tamaño infinito así que una red de 20 x 20 nodos va a ser una versión aproximada de un fenómeno que se da en redes de tamaño muy grande. Para poder aproximar esto a las situaciones reales que se dan en la naturaleza, calculamos las correlaciones que se dan en una red infinita y posteriormente cogemos una cuadricula de 20 x 20. Con esto conseguimos una aproximación infinitamente más realista puesto que estamos utilizando las correlaciones reales. Una representación visual de esto se puede ver en la figura 4.2.

Comprobamos que el comportamiento de ambas redes es prácticamente similar. Para ello simulamos para ambas redes las funciones de probabilidad de sus correlaciones $P(c_{ij})$. En la figura 4.3 podemos comprobar que sus distribuciones tienen prácticamente la misma distribución pero con un desfase. Sin embargo, únicamente con esta gráfica no podemos discernir con claridad si ambas redes presentan la misma tendencia. Para aclarar esta disyuntiva re-

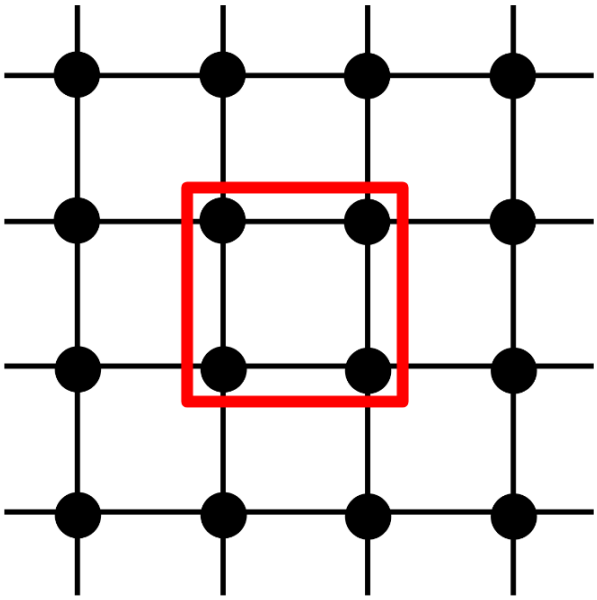

Figura 4.2: Esquematización visual de la red infinita calculada y en rojo la cuadricula de 20 x 20 de esa red infinita que consideraremos para nuestros cálculos.

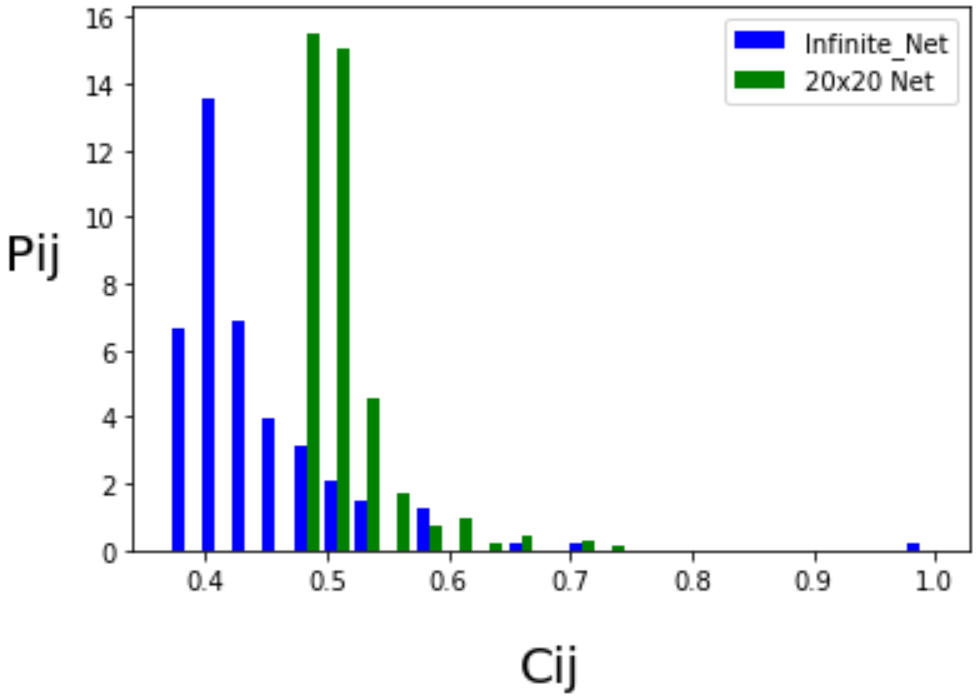

Figura 4.3: Distribución probabilística de las correlaciones del modelo.

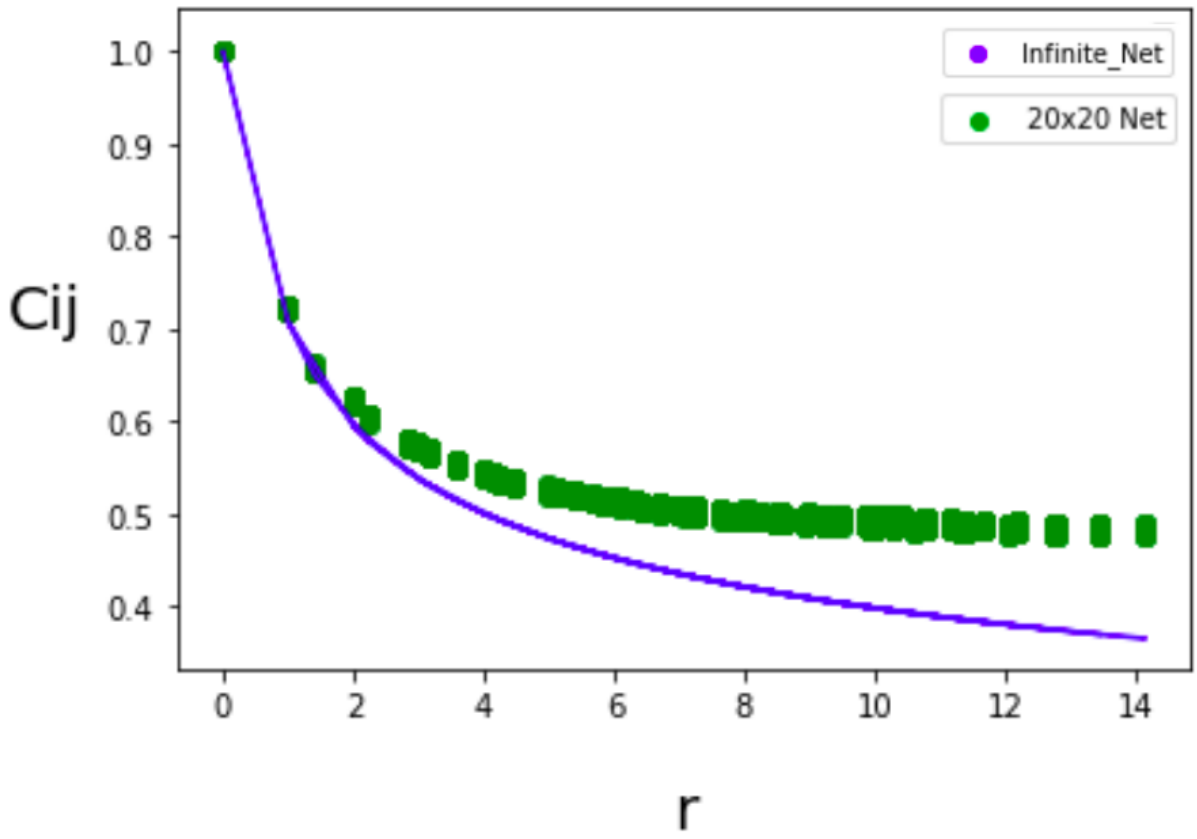

Figura 4.4: Comparativa de las correlaciones obtenidas entre los dos tipos de redes neuronales respecto a la distancia entre sus neuronas.

presentamos en la figura 4.4 la relación entre sus correlaciones c_{ij} respecto a r. Esta ultima variable básicamente expresa la distancia euclídea entre los puntos de la cuadricula. Para dos puntos $i = (x, y)$ y $j = (u, v)$ se da la siguiente relación para calcular su distancia: $r = \sqrt{(x-u)^2 + (y-v)^2}$. Cómo la relación entre el las correlaciones y la distancia entre las neuronas son simulares, podemos determinar que ambas distribuciones tienen la misma tendencia.

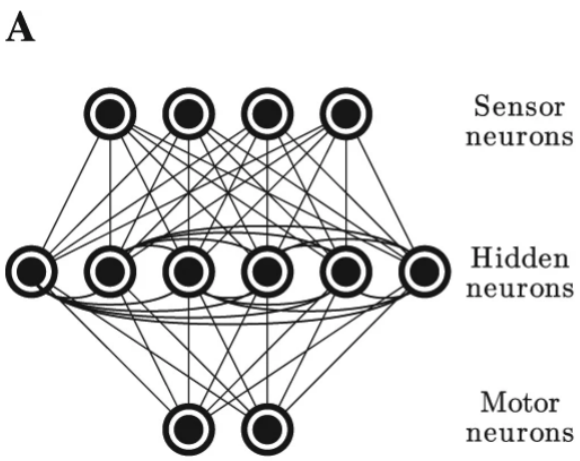

Figura 4.5: Distribución de la red neuronal actuadora. La imagen se ha obtenido del artículo de referencia [1].

4.3. Red neuronal actuadora

Con las nuevas correlaciones ya calculadas pasamos a trabajar con la segunda red neuronal. Esta red neuronal es la que gobierna explícitamente el movimiento de los agentes. Tiene la misma forma que en el articulo de referencia 4.5, pero con algunas modificaciones. Por ejemplo, diferente número de neuronas sensitivas y/ó motoras para adaptarla a otros entornos, en nuestro caso, el *Swimmer*.

El funcionamiento de esta red neuronal se ha esquematizado en la figura 4.6 explicando cómo es el funcionamiento del algoritmo de una manera visual y esquemática. Dependiendo del entorno de análisis la posición inicial será una u otra, pero siempre en la posición de mayor reposo.

Atendiendo a la figura 4.6 observamos que al introducir una acción al agente, este produce una respuesta que es detectada por medio de la neuronas sensitivas. Cabe destacar que esta señal es binarizada para permitir su procesamiento por parte de la red neuronal actuadora. Este procedimiento se explicará en detalle en la siguiente sección.

Siguiendo el esquema de la figura 4.6 y una vez que la posición del agente detectada por los sensores se encuentra binarizada, podemos pasarle esta información a la red neuronal actuadora. A partir de esta información, la red neuronal ajustará su nivel de correlaciones entre neuronas para mantenerse en la criticalidad. Este reajuste en las correlaciones provocará un

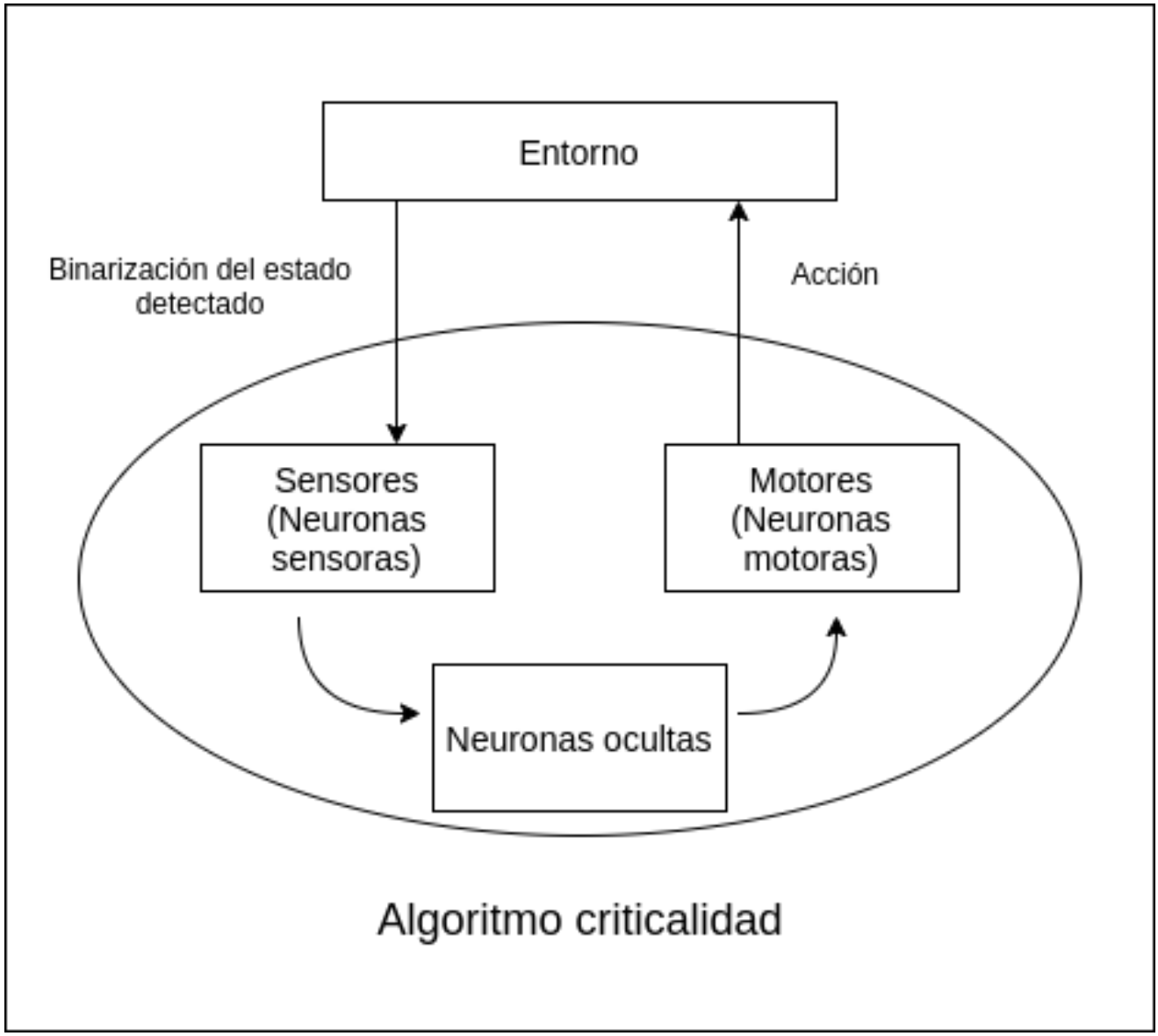

Figura 4.6: Descripción gráfica del funcionamiento del algoritmo de criticalidad.

tipo de output u otro en la red.

En nuestro caso es muy importante la diferencia entre el entrenamiento y la simulación. En el entrenamiento, la red neuronal actuadora intenta adaptar el nivel de correlaciones que hemos calculado con la anterior red neuronal infinita. Este entrenamiento se realiza para situar al agente en su punto crítico de funcionamiento. Sólo si el entrenamiento se produce de manera exitosa el agente experimentará comportamientos complejos e interesantes en la simulación que tendrán que ver con su posicionamiento en la transición de fases.

Los valores de las correlaciones entre neuronas en el entrenamiento adaptan el output que tiene que generar la red neuronal para hacer que este se mantenga en el punto critico que le hemos indicado. Posteriormente en la simulación comprobamos si el agente ha conseguido realmente hacer bien esta tarea con las gráficas que presentamos en la sección de resultados.

Aunque durante la simulación no se utiliza el algoritmo de criticalidad, podemos asegurar que este mantiene una estructura de correlaciones casi idéntica a la obtenida al acabar el entrenamiento. Esto es debido a que el entrenamiento se realiza en periodos muy largos y la evaluación son solo algunas simulaciones. Por lo tanto, las correlaciones al acabar la simulación mantienen la misma distribución.

Más en detalle, estas correlaciones están relacionadas con el nivel de energía de los agentes, puesto que la distribución de correlaciones que se da en el punto critico corresponde con un nivel de energía mínimo. Esto se puede entender fácilmente con un ejemplo gráfico: Imaginemos que nuestro agente es una canica en un espacio con pequeños agujeros en el mismo. Cuando la canica se acerca a un agujero, esta tiende a ir a este antes que a otra posición debido al nivel más bajo de energía potencial. En nuestro ejemplo ocurre algo similar, el agente tiende a ir a los estados de menor energía. La diferencia reside en que no es la gravedad la responsable de esta tendencia, sino la acción de la red neural haciendo que se mantenga un cierto nivel de correlaciones.

Este funcionamiento con la red neuronal puede ser conseguido gracias a que es una red probabilística: Cada estado al que el agente puede saltar tiene una probabilidad asociada dependiendo del nivel de energía. El agente decide si saltar o no al estado siguiente dependiendo de la diferencia de energías. Si el estado al que va a saltar tiene un nivel de energía asociado muy bajo, las probabilidades de saltar a este serán mucho mayores que a otro con una energía más alta.

4.4. Binarización

La binarización es realizada para que la red neuronal pueda tener en cuenta los *inputs* enteros en cualquier rango. Este método es el que se propone en el articulo de referencia [1], con la diferencia de cambiar el numero de sensores según el entorno que utilicemos. La función presentada codifica un numero entero en un numero binario. Con esto hace que se pueda utilizar el algoritmo de criticalidad para cualquier tamaño de red.

$$output = bitfield(floor(\frac{x+x_{max}}{2x_{max}+\epsilon 2^{sensores}})) \tag{4.2}$$

La función de binarización recibe cómo *inputs* x y x_{max}. Para estos *inputs* y un tamaño de red dado, binariza estos valores. En el caso del *Mountain Car* coge los valores de velocidad y esto lo hace con un total de 4 sensores, osea 2^4. Cómo podemos observar, esta formula actúa a modo de limitante, poniendo cómo máximo 1 y cómo mínimo 0.

Esta función funciona de la siguiente manera: Partimos de la base de que los valores enteros pueden ir de $-x_{max}$ a x_{max}, por lo tanto tiene una distancia numérica total de $2x_{max}$. Para acotar los números entre 0 y 1, referimos la distancia entre ese valor y el origen ($x-(-x_{max})$ respecto de la distancia máxima total ($2x_{max}$).

Esto nos dará un numero entre 0 y 1. Para determinar si es 0 ó 1, subdividimos el espacio en 2^S subdivisiones, dependiendo del numero de sensores (S) que tengamos y elegimos un índice con la función *floor*. Después, transformamos ese número en un número binario con la función *bitfield* de S *bits* (p.ej. si el índice es 5 y $S = 4$, tenemos $s' = [0, 1, 0, 1]$).

Por último, cómo el modelo de *Ising* es +1/-1 transformamos el output binario de la formula 4.2 haciendo $s = s'(2-1)$. Dando esto cómo resultado $s = [-1, 1, -1, 1]$ que es el vector de 4 dígitos por cada input que hemos detectado con los sensores.

El parámetro ϵ es un parámetro conocido cómo *Machine epsilon*. Este parámetro da un límite superior al error relativo debido al redondeo en la aritmética de coma flotante. Con esto se consigue que el valor del denominador sea siempre al menos este parámetro mayor que el valor del numerador.

Capítulo 5

RESULTADOS Y ANÁLISIS

Para la obtención de los resultados tal y cómo hemos comentado anteriormente se entrena a un agente haciendo que este intente ajustar el nivel de correlaciones de una red neuronal situada en el punto crítico. Tras este entrenamiento, evaluamos si la red neuronal actuadora ha sido capaz de adaptar su nivel de correlaciones para hacer que el agente al que controla opere en la transición entre dos fases. Esta transición es el punto en el que se maximizan sus fluctuaciones de energía. Este máximo en las fluctuaciones significa precisamente el punto en el que se dan los comportamientos más complejos.

Para todos los métodos siguientes definimos una red neuronal que consiste en un modelo de *Ising* que describe una red de 2 neuronas motoras, 4 neuronas sensoras y N_h neuronas ocultas. Las unidades motoras son las responsables de definir las acciones que hacen los agentes y las sensoras las que detectan la posición del agente. Gracias a estas neuronas sensoras las cuales funcionan cómo input de la red, el agente será accionado con una acción u otra dependiendo de donde se encuentre.

Todo el código que se ha utilizado para la obtención de los siguientes resultados se ha hecho de libre acceso en *GitHub* [1].

[1] `https://github.com/VictorOtin/Master_thesis_UNIZAR`

5.1. Mountain Car

El objetivo de este entorno es que el agente aproveche la energía potencial para subir la montaña, evitando así quedarse en el valle. La red neuronal recibe la velocidad del sistema cómo input por medio de las neuronas sensitivas. La dirección de la acción que el coche aplica se controla con las neuronas motoras (*output*).

Antes de simular el agente, tenemos que entrenarlo. La finalidad de este entrenamiento es que la red aprenda las correlaciones necesarias para llevar al agente a la criticalidad. El entrenamiento no tiene otro objetivo explicito más que el de ajustar las correlaciones del sistema e intentar igualarlas a una muestra de la distribución de correlaciones de la nueva red neuronal calculada. Si el agente consigue realizar este ajuste, podremos comprobar cómo el agente es capaz de generar comportamientos complejos sin habérselos definido explícitamente.

Durante el entrenamiento, los agentes se inicializan en las posición inicial aleatoria en la parte inferior de entornos $x \in [0{,}4, 0{,}6]$ e $y = 0$. El estado de la red neuronal es aleatorio y los parámetros iniciales h_i y J_{ij} se ponen a cero. Los agentes son ejecutados 1000 iteraciones de 5000 *steps* cada iteración. En cada iteración los valores m_i^m y c_{ji}^m son computados. Al final de cada iteración se computa la ecuación 3.1 3.4 con las cuales se actualizan estos estados h_i y J_{ji}.

Comprobamos los resultados obtenidos en este entorno para la nueva arquitectura propuesta. Para ello, analizamos el comportamiento de los controladores neuronales y los patrones de comportamiento de los agentes con respecto a las posibilidades de sus parámetros. El objetivo es comprobar si la regla de aprendizaje propuesta consigue llevar a los agentes cerca de los puntos críticos.

Para poder comparar los resultados con los del articulo variaremos también los valores de β para explorar el espacio de parámetros del agente. Variar el valor de β equivale a un reescalado global de los parámetros del agente ya que estamos cambiando la forma de actualizar los estados h_i y J_{ji}.

Primero, evaluamos la presencia de criticalidad en el controlador neuronal del agente. En concreto, buscamos la presencia de una transición de fase continua en la que un parámetro de orden del sistema (la entropía) presenta una transición brusca mostrando una divergencia

de su derivada (la capacidad calorífica). En segundo lugar, analizamos no solo el comportamiento del controlador neural sino el agente en su conjunto, buscando transiciones de comportamiento de los agentes y divergencias de la susceptibilidad del comportamiento del agente a cambios paramétricos.

Una divergencia en la capacidad calorífica es indicador suficiente para determinar la presencia de una transición de fases en ese punto. Esto es debido a que el punto crítico representa una transición de fase continua en la que el sistema es máximamente sensible a los cambios paramétricos. En las figuras 5.1 y 5.2 calculamos entonces la entropía y la capacidad calorífica para distintos valores de β. En la figura 5.1 observamos cómo el sistema muestra una divergencia de la capacidad calorífica similar a la del modelo de *Ising*, lo que sugiere que el controlador neuronal del robot se encuentra en un punto crítico. Este punto crítico es parecido a una transición de fase continua, lo que sugiere que el controlador se auto organiza para presentar la máxima sensibilidad a los cambios en su espacio de parámetros.

Un cambio brusco en la entropía del sistema, así cómo una divergencia en la capacidad calorífica, son indicadores suficientes para demostrar que el funcionamiento del agente se sitúa en un punto crítico. En nuestro caso, esto se da para $\beta = 1$ ya que, cómo hemos mencionado en la sección 2.3, lo hemos definido en este punto.

Con la finalidad de entender mejor que significa una variación en el valor de β, vamos a representar diferentes valores del mismo para otras variables. Para comprobar si el entrenamiento ha sido exitoso y el funcionamiento del agente se ha conseguido lleva a una transición de fase. Para ello, buscamos puntos distintos de los del entrenamiento para comprobarlo. En nuestro caso, escogemos $\beta = 0{,}75$ y $\beta = 1{,}2$ aunque podrían haberse seleccionado otros puntos distintos. Los valores menores de 1 representan, según la dinámica de *Glauber*, puntos de mayor aleatoriedad y los mayores de 1 son puntos que muestran un comportamiento más fijo y estable.

En la figura 5.3 podemos observar que cuando la temperatura es menor de la temperatura critica ($\beta = 0{,}75$) el agente no consigue llegar a la cima de la montaña ninguna vez, debido a su comportamiento aleatorio. Cuando esta es mayor ($\beta = 1{,}2$), este presenta un comportamiento muy rígido, experimentando unas trayectorias que van de una cima a otra. Sin embargo, en $\beta = 1$ el agente muestra un comportamiento mucho más flexible y variado en el que se aprecian comportamientos de ambas fase. Por consiguiente, este comportamiento puede ser calificado cómo un comportamiento más complejo puesto que tiene comportamien-

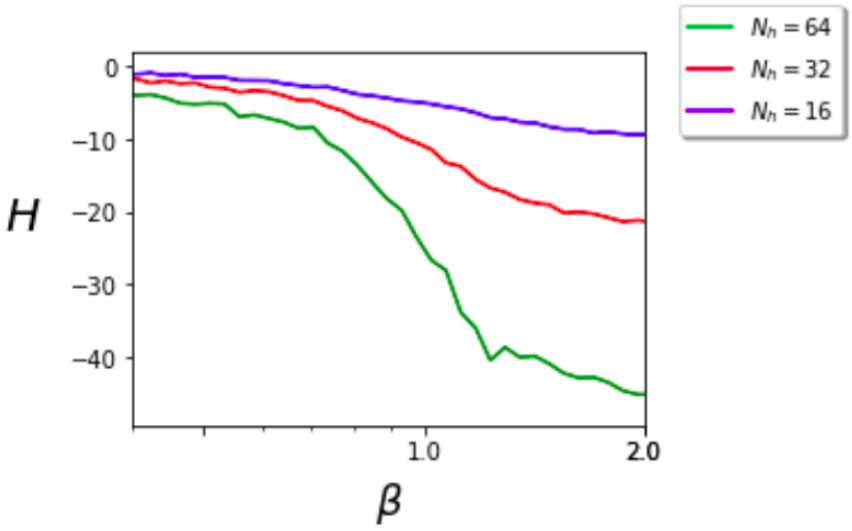

Figura 5.1: Entropía del controlador neural del agente para diferentes tamaños hasta $N = 70$ ($N_h = 64$) para 101 valores de β.

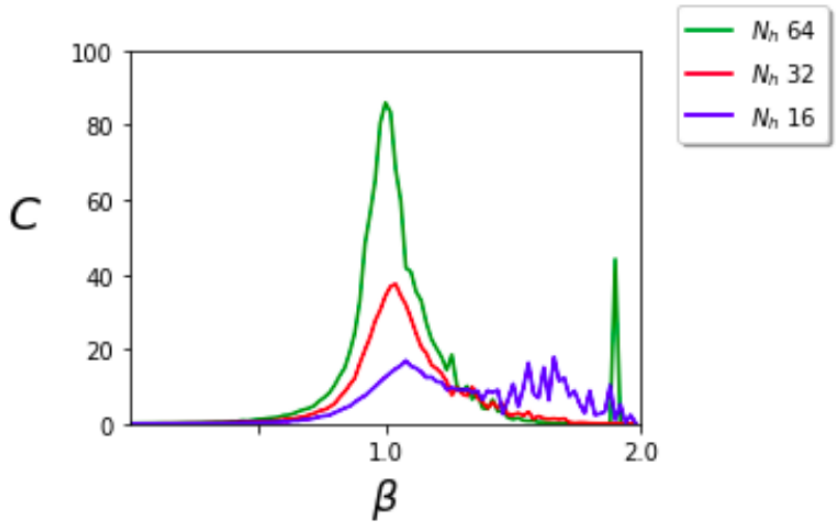

Figura 5.2: Capacidad calorífica de las neuronas del agente calculada cómo $C(\beta) = -\beta\frac{\partial H(\beta}{\partial \beta}$. Se aprecia una divergencia en su capacidad calorífica a medida que aumenta el número de neuronas N, lo que sugiere que el controlador neuronal del sistema está cerca de una transición de fase continua.

tos de ambas fases 5.9.

Para obtener una imagen más general de diferentes agentes y tamaños, podemos analizar las transiciones de comportamiento en variables relevantes de los sistemas agente-entorno.

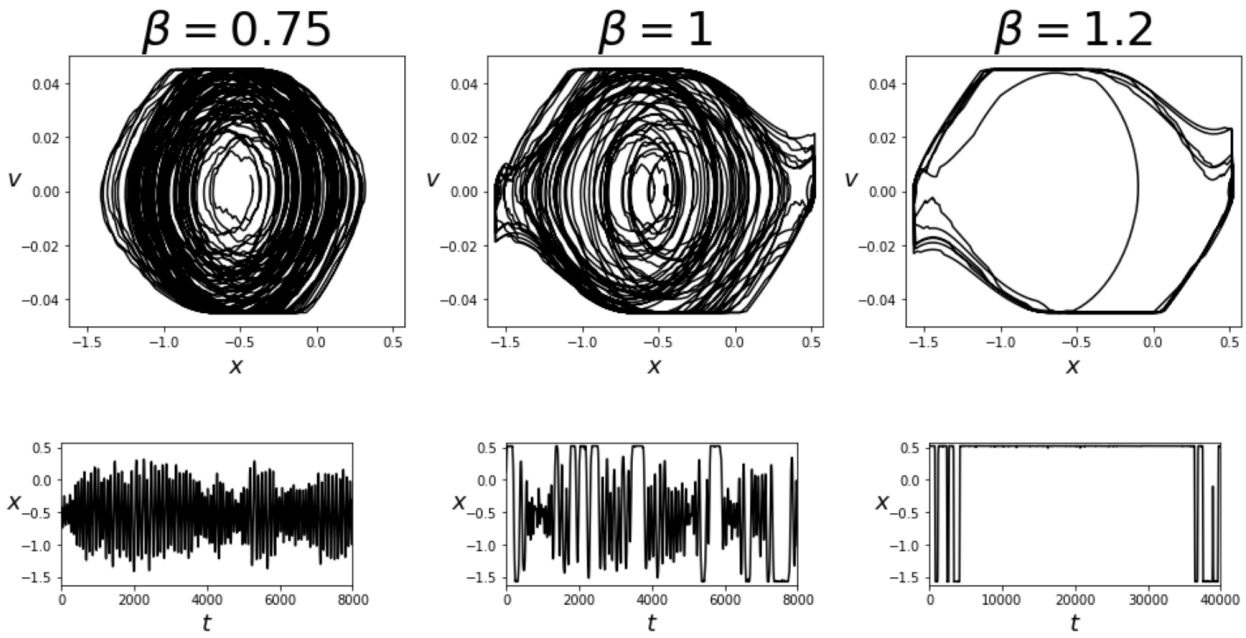

Figura 5.3: Transición en el régimen de comportamiento de los agentes con $N_h = 64$.

Además, analizamos si el comportamiento del agente, y no solo su controlador neuronal, está cerca de un punto crítico. Para inspeccionar esto, calculamos la altura media de los agentes y en nuestras simulaciones (figura 5.4), y calculamos la susceptibilidad de este valor de altura cómo $X_y(\beta) = \beta \frac{\partial y}{\beta}$ (figura 5.5). En este caso, la susceptibilidad parece aumentar de manera monótona con el tamaño cuando se duplica el tamaño, incluso si estos aumentos no son tan uniformes cómo en las anteriores figuras 5.1 y 5.2. Aunque más pruebas de criticidad podrían validar si la criticidad también se encuentra en todo el sistema agente-entorno, las cifras sugieren que la transición de fase continua del controlador neuronal se corresponde con una transición brusca en el comportamiento del agente.

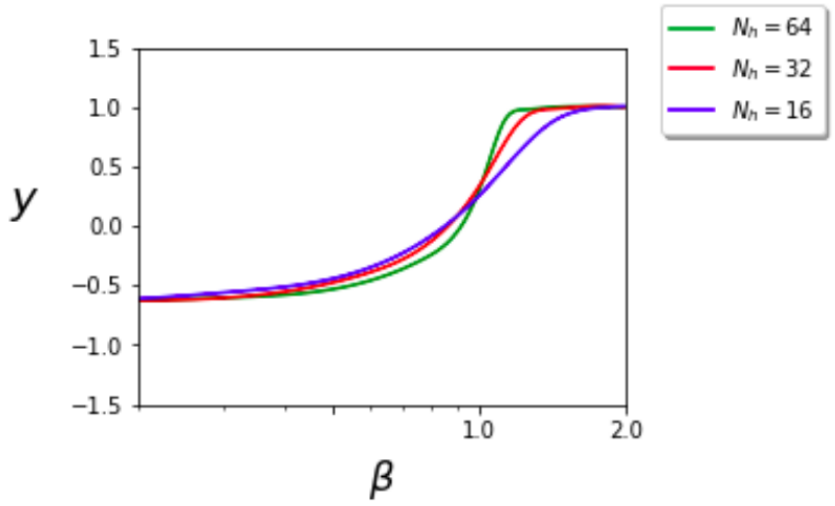

Figura 5.4: Altura media y de los agentes para diferentes tamaños hasta $N = 70$ ($N_h = 64$) y 101 valores de β.

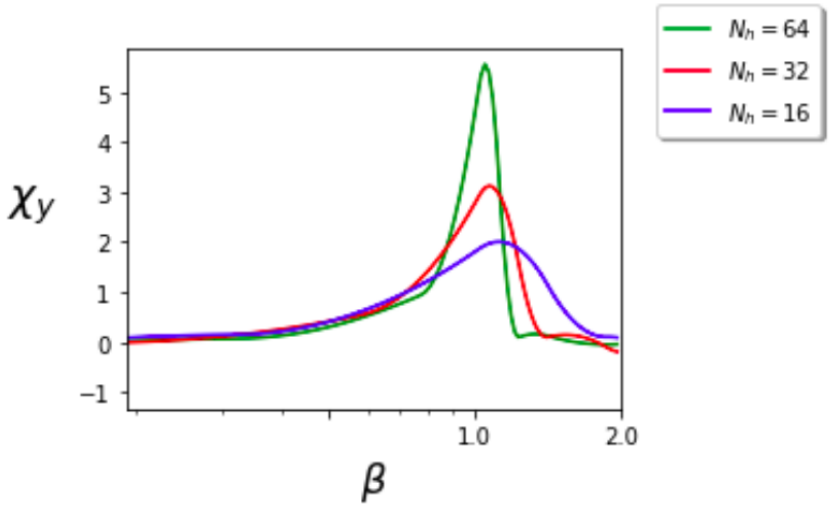

Figura 5.5: Susceptibilidad del comportamiento del agente, calculada cómo $X_y(\beta) = \beta\frac{\partial y}{\beta}$.

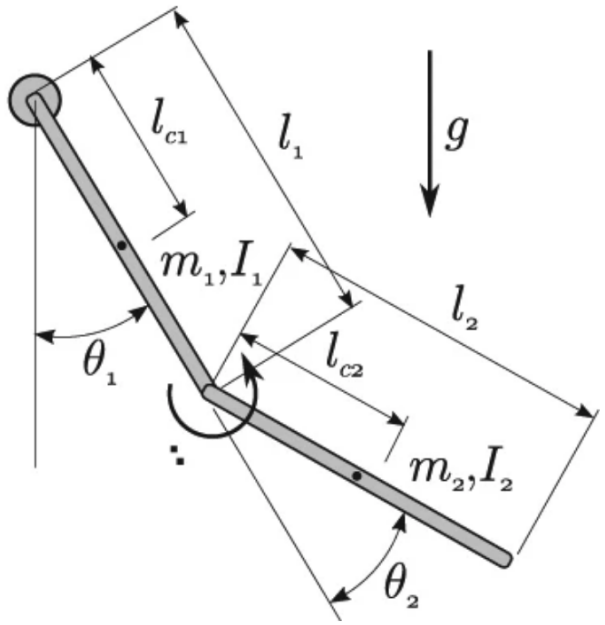

Figura 5.6: Descripción gráfica del modelo *Acrobot*. Imagen obtenida del articulo de referencia [1].

5.2. Acrobot

El segundo entorno sobre el que se analiza el algoritmo se aprendizaje con esta nueva consideración es el *Acrobot* 5.6. Este entorno es un péndulo de dos eslabones con sólo la segunda articulación activada. Inicialmente, ambos enlaces apuntan hacia abajo. El objetivo es balancear el efector final a una altura de al menos la longitud de un eslabón por encima de la base. Ambos eslabones pueden oscilar libremente y pueden pasar uno al lado del otro, es decir, no chocan cuando tienen el mismo ángulo.

Inicializamos el péndulo en una posición de reposo: $\theta_1, \theta_2, \dot{\theta_1}, \dot{\theta_2} :\in [-0{,}1, 0{,}1]$. Igual que en el anterior caso el estado de la red neuronal es aleatorio y los parámetros iniciales h_i y J_{ij} se ponen a cero. El numero de iteraciones y *steps* se mantienen igual: $Iteraciones = 1000$ y $steps = 5000$. Para comprobar si la regla de aprendizaje consigue llevar a este agente a la criticalidad verificaremos cómo se comporta la entropía 5.7 y la capacidad calorífica 5.8 frente a β así cómo y vs x e y vs t. Todos los análisis, igual que en el caso anterior los realizamos no solo para diferentes valores de los del entrenamiento sino tambien para valores inferiores y superiores. De este modo podremos comprobar si el algoritmo de aprendizaje ha sido capaz de situar al agente en una transición de fase.

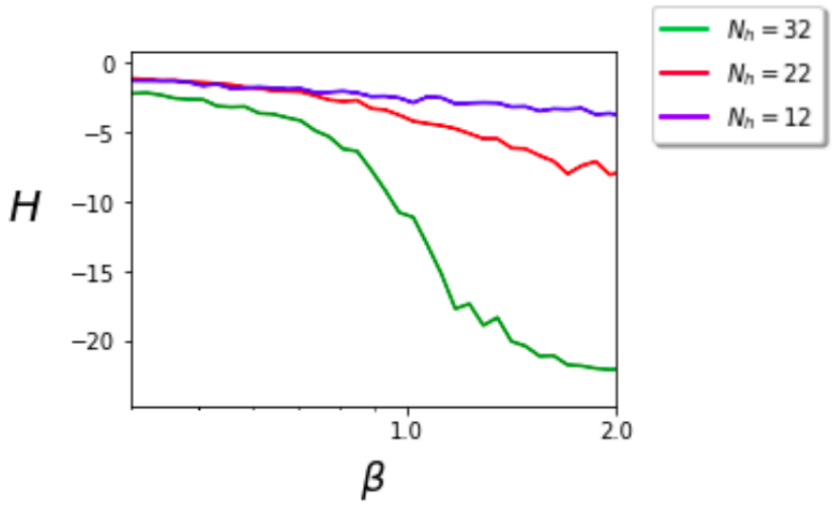

Figura 5.7: Entropía del controlador neural del agente para diferentes tamaños hasta $N_h = 32$ para 101 valores de β.

Cómo hemos comentado anteriormente, un cambio brusco en la entropía del sistema es una característica suficiente para determinar que el sistema se encuentra en una transición de fase ya que en ese punto el sistema es más sensible a cambios paramétricos. En la representación de la entropía se puede apreciar esta transición de fase en $\beta = 1$. Además, se representa también la capacidad calorífica, la cual, para un sistema variable es igual a la derivada parcial de la entropía respecto al factor β.

Atendiendo a las figuras 5.8 y 5.7 se aprecia claramente esta divergencia para el valor de temperatura al que se ha llevado a cabo el entrenamiento, $\beta = 1$. Esto significa que el sistema es máximamente sensible a cambios paramétricos en este punto y por lo tanto, se puede asegurar que el entrenamiento ha sido exitoso y que se ha logrado posicionar al agente en el punto crítico de funcionamiento.

Para comprender más ampliamente cómo varían los comportamientos del conjunto agente-entorno, comprobamos cual es el comportamiento del robot con estas mismas variaciones pero atendiendo a su relación entre su posición (x frente a y) así cómo su altura respecto al tiempo.

Observamos que el agente *Acrobot* con $N_h = 32$ en $\beta = 1$ muestra un rango diverso de comportamientos, pudiendo llegar a la cima del plano mientras que, cuando se baja o aumenta β, se deriva a otros modos de comportamiento en regímenes menos diversos. Aunque

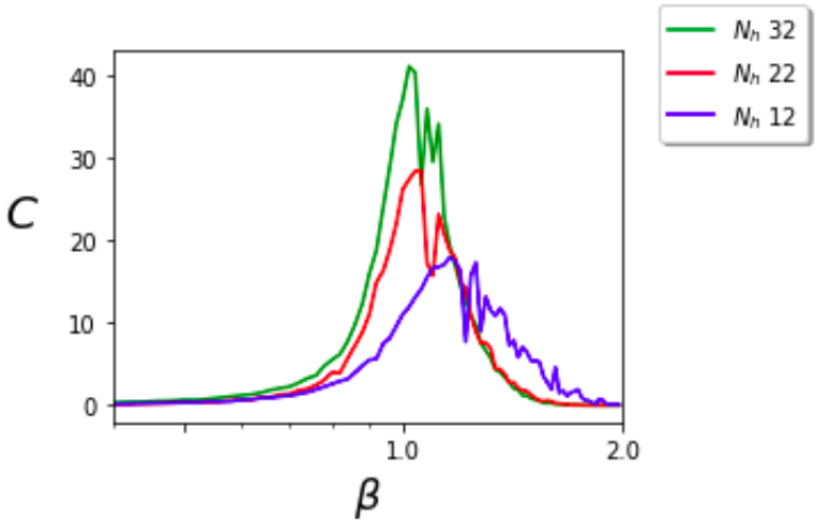

Figura 5.8: Capacidad calorífica de las neuronas del agente. Capacidad calorífica calculada cómo $C(\beta) = -\beta\frac{\partial H(\beta}{\partial \beta}$. Se aprecia una divergencia en su capacidad calorífica a medida que aumenta el número de neuronas N, lo que sugiere que el controlador neuronal del sistema está cerca de una transición de fase continua.

solo se representa un agente para cada entorno, los resultados de la figura 5.9 son similares en todos los agentes y tamaños.

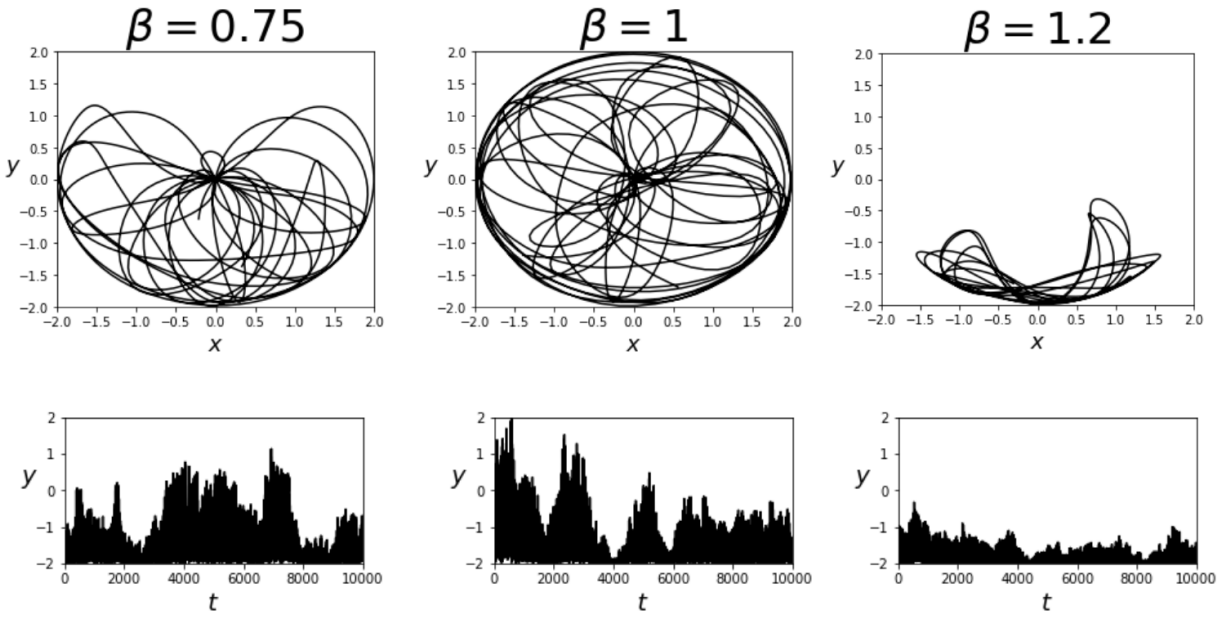

Figura 5.9: Transición en el régimen de comportamiento de los agentes con $N_h = 32$.

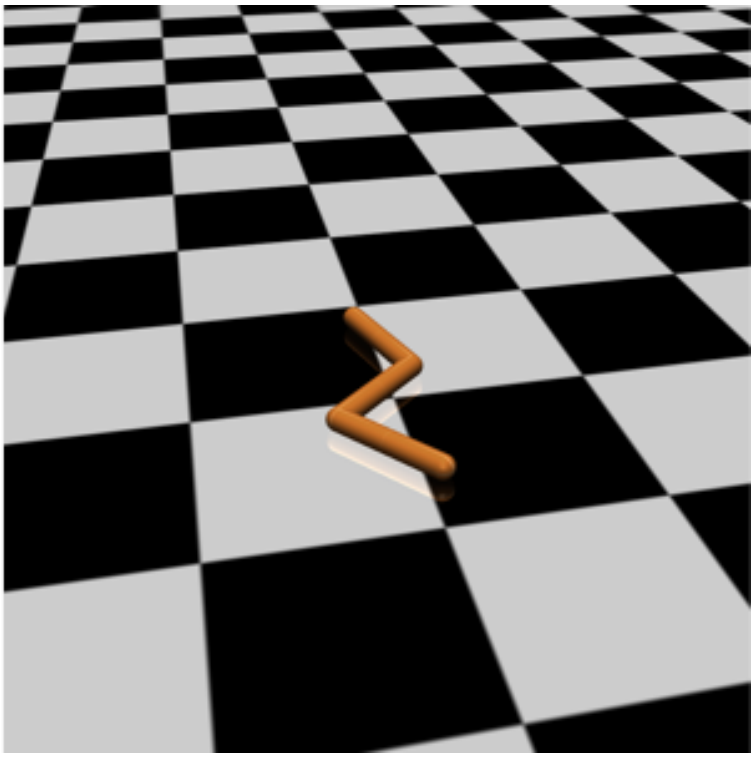

Figura 5.10: Entorno *Swimmer*. Entorno obtenido del *physics engine simulator MuJoCo* [19]

5.3. Swimmer

Este entorno consiste en un robot nadador de tres eslabones que se desplaza en un fluido viscoso, figura 5.10. El objetivo de este robot es hacer que se desplace lo más rápido y lejos posible. Nuestro objetivo sin embargo es estudiar si el comportamiento del mismo se encuentra en una transición de fase. Para ello analizaremos esto de una manera algo diferente al resto de experimentos para así mostrar que hay diferentes formas de demostrar esta misma hipótesis.

Para evaluar la complejidad del movimiento que el agente es capaz de generar, utilizaremos las variables de distancia recorrida y de variedad de movimientos. Cabe destacar que para esta comprobación, al igual que en los casos anteriores, es muy importante la comparación entre los casos con una temperatura mayor y menor a la critica. Si el agente se encuentra en unan transición de fase mostrará comportamientos de ambas.

Para poder detectar y actualizar los estados del *Swimmer*, se han incorporado una serie de nuevos sensores al agente. Las variables que serán medidas son: La posición y la velocidad. El número total de sensores elegido es de 8: Dos sensores que detectan la posición, uno

para el eje x y otro para el y, así mismo dos más para las derivadas de la posición. Uno para la velocidad en el eje x, $\dot{x}$ y otro para la velocidad en el eje y, $\dot{y}$. Estos 4 sensores, se ponen en ambas articulaciones, lo cual hace un total de 8.

Los sensores detectan y actualizan la nueva posición del *Swimmer* en cada *step*. Cómo utilizamos 8 sensores, necesitaremos 32 neuronas sensitivas cómo input de la red, ya que cada input se binariza en 4 dígitos para que el algoritmo pueda ser aplicado. Además de estas 32 neuronas sensitivas, se utilizan 2 unidades motoras. Estas neuronas motoras son las encargadas de, según el valor de los pesos de la red neuronal, tomar una acción u otra. Una vez se ha definido claramente la estructura de la red neuronal, se realiza todo el proceso de entrenamiento de la misma durante 1000 iteraciones, durante T=3600 para una red neuronal de un tamaño total de 66 neuronas (32 sensitivas, 32 neuronas ocultas y 2 motoras).

Para realmente comprobar que el algoritmo de aprendizaje lleva al agente a la criticalidad y es capaz de generar comportamientos complejos sin indicárselos explícitamente, la manera más inequívoca de hacerlo es comparando el comportamiento del agente para distintos valores de β. Cabe destacar que esta forma de comprobación es equivalente a las presentadas anteriormente en las figuras 5.2 y 5.9 ya que en ambos casos comprobamos que influencia tienen los distintos valores de β en el comportamiento del agente. En este caso este comportamiento será comprobado a partir de su posición x frente a y, y en los casos anteriores a partir de su posición y velocidad respecto al tiempo. Para todas las gráficas se han ploteado 10 trayectorias, cada una ha sido ploteada de un color.

Partiendo de que el entrenamiento se ha realizado a la temperatura crítica, lo que hacemos cómo primer experimento es reducir el valor de esta temperatura para ver el comportamiento del agente. Esta disminución de la temperatura, igual que en experimentos anteriores, significa añadir ruido y hacer el comportamiento del agente más aleatorio. Este es el caso de $\beta = 0{,}75$. Los resultados tras el entrenamiento los podemos ver en la figura 5.11. Atendiendo a los valores de los ejes se puede comprobar que el agente no logra avanzar prácticamente nada y se distingue claramente un comportamiento aleatorio. El siguiente valor que se comprueba es el de $\beta = 1{,}2$. Para este valor β, se dan, igual que en los anteriores entornos un comportamiento claramente rígido. Las trayectorias que traza el agente son completamente rectas y sin ninguna variación en la misma. En la figura 5.12 podemos observar la rigidez de este comportamiento.

Comparamos estos comportamientos con el caso en el que presuntamente se da la criti-

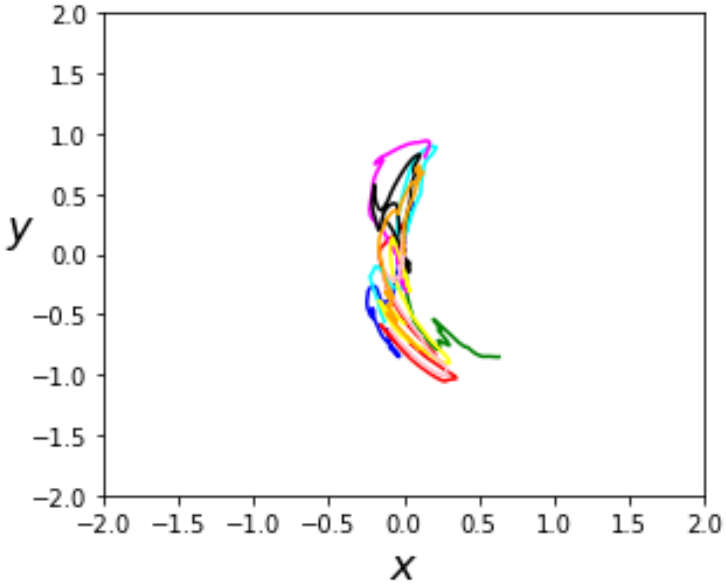

Figura 5.11: Trayectorias generadas haciendo pasar un *input* aleatorio por el algoritmo de aprendizaje con $\beta = 0{,}75$.

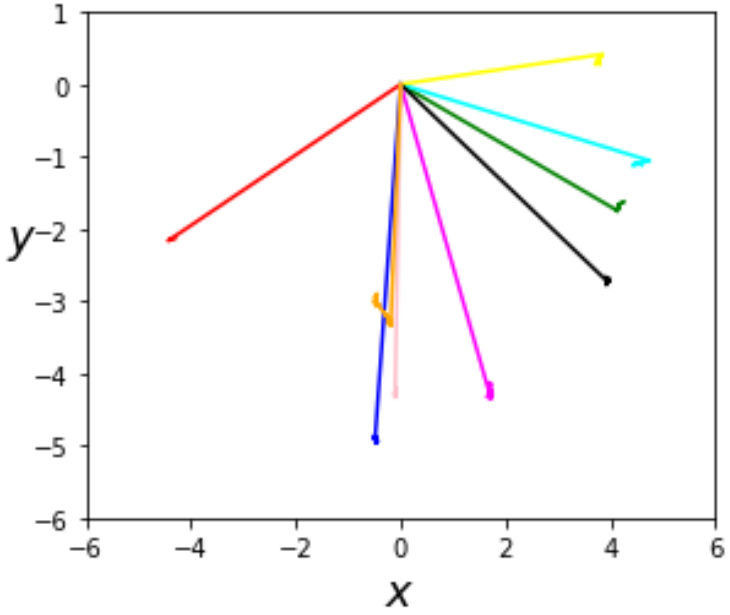

Figura 5.12: Trayectorias generadas haciendo pasar un *input* aleatorio por el algoritmo de aprendizaje con $\beta = 1{,}2$.

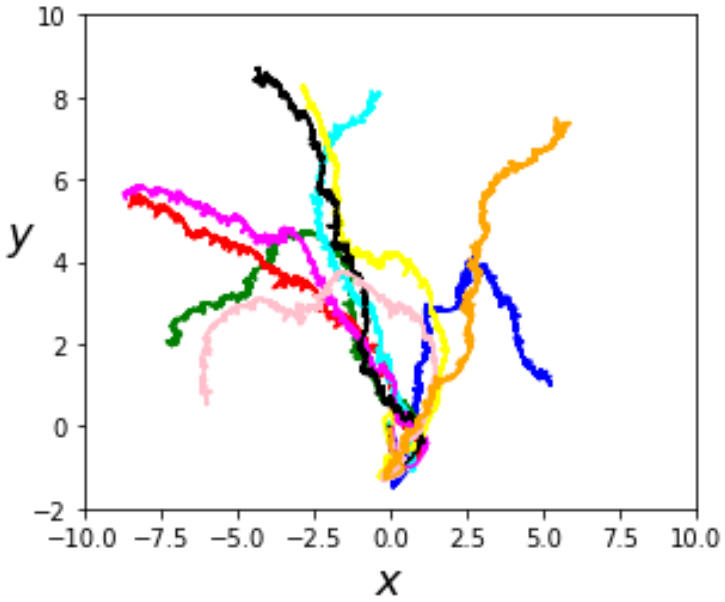

Figura 5.13: Trayectorias generadas haciendo pasar un *input* aleatorio por el algoritmo de aprendizaje con $\beta = 1$.

calidad $\beta = 1$. Decimos presuntamente porque solo si el entrenamiento ha sido efectivo habremos podido posicionar al agente en un punto crítico. Por contra, si el entrenamiento no ha sido efectivo el agente no estará funcionando en una transición de fase y por lo tanto no se apreciarán comportamientos complejos ni que pertenezcan a ambas fases. El punto crítico lo inducimos con el aprendizaje, pero sin aprendizaje, $\beta = 1$ no tendría por qué ser punto crítico.

Para comprobar esto, analizamos las trayectorias y comportamientos del agente con este valor de $\beta = 1$. En la figura 5.13 comprobamos por lo tanto que el agente desarrolla comportamientos que pertenecen a las dos anteriores fases descritas. Tiene la parte de $\beta = 1{,}2$ que representa comportamientos ordenados y rígidos porque consigue recorrer distancia grandes, pero también la parte de aleatoriedad ya que realiza múltiples cambios de trayectoria, tomando diferentes curvas y direcciones.

Dada esta comparación del comportamiento del conjunto agente-entorno para los diferentes valores de β, se puede determinar que el efecto del entrenamiento ha sido exitoso y se ha logrado posicionar al *Swimmer* en el punto crítico de funcionamiento.

Capítulo 6

TRABAJO FUTURO

El principal objetivo de esta linea de investigación es explorar mecanismos que reproduzcan algunas propiedades de los sistemas biológicos. Estos mecanismo pueden explorar el desarrollo de una inteligencia artificial general AGI ya que hay muchos sistemas biológicos que presentan este tipo de inteligencia. Hasta ahora los algoritmos de inteligencia artificial se diseñan para lograr objetivos muy concretos por lo que después, no son capaces de aplicar esa inteligencia a otros campos. Por lo tanto, aplicar inteligencia artificial local ALI para resolver una gran variedad de escenarios es computacionalmente intratable ya que para cada situación, el algoritmo debe sufrir cambios de diseño muy significativos.

Con este modelo de red neuronal, se abre una nueva posibilidad. Se ha comprobado que la red neuronal es capaz de controlar, de forma satisfactoria, diferentes agentes en situaciones muy distintas. En el futuro, se podrían explorar redes neuronales basadas en principios similares a los propuestos en este proyecto ya que es posible que, con estos modelos, sea mucho más fácil la AGI de las redes neuronales.

La continuación más intuitiva para verificar este concepto de AGI sería probar el algoritmo en otros entornos y con tamaños de red mayores. Por ahora, todos los entornos que analizamos son en 2D y su complejidad es mucho menor a la que estamos acostumbrados a resolver los seres humanos. Además, las modificaciones que se deben de hacer al algoritmo en materia de sensores o motores varia dependiendo de las características del mismo.

Otra posible linea de investigación estaría más relacionada con el entendimiento del cerebro humano a partir de los puntos críticos. En la neurociencia siempre se ha trabajado con la hipótesis de que era posible entender cada parte del cerebro por separado. El hecho de que este operase en un punto crítico indicaría que es necesario tener en cuenta las interacciones

entre distintas partes del mismo. Se ha especulado ya con la posibilidad de que la criticalidad podría ser el punto de partida para otros fenómenos más complejos que ocurren en nuestro cerebro, cómo por ejemplo la consciencia [16].

Cabe destacar que ambas lineas de trabajo están implícitamente relacionadas: El poder entender cómo funciona nuestro cerebro nos acercaría a pasos agigantados a poder replicar su funcionamiento de manera artificial. El hecho de intentar replicar el funcionamiento del cerebro humano sin entender en detalle su funcionamiento se torna en una tarea extremadamente compleja.

Capítulo 7

CONCLUSIÓN

Recapitulando las ideas principales presentadas hasta ahora, hemos probado cómo, tomando un conjunto de correlaciones elegidas al azar de una distribución generada por un modelo de *Ising* en un punto crítico, podemos construir un nuevo modelo que parece estar también cerca de un punto crítico.

En un artículo en el que nos basamos se propone un algoritmo de aprendizaje que intenta replicar estos comportamientos. En dicho artículo se prueba una red neuronal finita en dos entornos y se comprueba que es capaz de hacer que varios agentes se adaptaran a su entorno llevándolos a un punto crítico. En este punto crítico, los agentes presentan comportamientos espontáneos sin habérselos definido específicamente.

En este proyecto se modifica y analiza una nueva red neuronal. Se trata de una red neuronal infinita de la que se coge una cuadricula de 20 x 20. Esta nueva red se introduce no solo con la finalidad de replicar de una manera más fidedigna el comportamiento de los seres vivos en el mundo real sino también, porque tiene un coste computacional mucho menor. Esto es debido a que se puede calcular directamente con la fórmula teórica. Una vez calculada, se comprueba si es capaz de hacer que otros sistemas se posicionen en un punto crítico y se adapten a su entorno generando comportamientos espontáneos. Primero se realizan las pruebas sobre el *Mountain Car* y el *Acrobot*. Tras analizar los resultados se comprueba que comparten la misma tendencia que los del articulo tomado cómo referencia. Por consiguiente, se pueden considerar cómo satisfactorios y se pueden generalizar los resultados del articulo.

Tras realizar las pruebas en los entornos descritos, se afronta un reto más ambicioso sobre un entorno denominado *Swimmer*. En este entorno se observa si la red es capaz de hacer que el *Swimmer* avance trazando trayectorias complejas e interesantes.

Dichos objetivos no se definen de forma explícita. Simplemente se le da la consigna a la red de alcanzar un nivel especifico de correlaciones. A partir de aquí, con el entrenamiento de la red, el *Swimmer* se adapta al entorno generando comportamientos complejos y espontáneos. Este es un resultado muy destacable ya que la red neuronal ha sido capaz de hacer que un agente complejo sea capaz de coordinar los movimientos de sus articulaciones para poder avanzar sin habérselo definido de forma explícita.

Tras realizar todos los cálculos se observa que la red neuronal ha sido capaz de posicionarlos en un punto crítico. Además, nuestros resultados muestran que la criticidad podría generarse mediante mecanismos bastante simples que solo se basan en información local, manteniendo correlaciones específicas alrededor de un valor dado.

Cómo conclusión, se considera que el resultado del proyecto es satisfactorio. Se ha observado que la nueva red neuronal es capaz de posicionar los sistemas en una transición de fase, y de esta forma hacer que se adapten a su entorno consiguiendo los objetivos, pero sin definírselos de forma explícita.

Capítulo 8

HERRAMIENTAS

Para trabajar con los entornos y sus agentes se ha utilizado la librería *OpenAI* [6]. A esta librería pertenece también el *physics engine MuJoCo*, [19] el cual necesita del entorno Ubuntu-Linux [20] para funcionar correctamente. El código se ha compartido utilizando el repositorio *GitHub*. Para el resto de procedimientos cómo la comprobación del funcionamiento o la elaboración de gráficas se ha utilizado *Python*, especialmente las librerías *Numpy* [15]y *MatPlotLib* [8].

Bibliografía

[1] Miguel Aguilera y Manuel G Bedia. "Adaptation to criticality through organizational invariance in embodied agents". En: *Scientific reports* 8.1 (2018), págs. 1-11.

[2] Per Bak y Dante R Chialvo. "Adaptive learning by extremal dynamics and negative feedback". En: *Physical Review E* 63.3 (2001), pág. 031912.

[3] Lionel Barnett y col. "Information flow in a kinetic Ising model peaks in the disordered phase". En: *Physical review letters* 111.17 (2013), pág. 177203.

[4] John M Beggs y Dietmar Plenz. "Neuronal avalanches in neocortical circuits". En: *Journal of neuroscience* 23.35 (2003), págs. 11167-11177.

[5] Nils Bertschinger y Thomas Natschläger. "Real-time computation at the edge of chaos in recurrent neural networks". En: *Neural computation* 16.7 (2004), págs. 1413-1436.

[6] Greg Brockman y col. "Openai gym". En: *arXiv preprint arXiv:1606.01540* (2016).

[7] Gerald Hahn y col. "Neuronal avalanches in spontaneous activity in vivo". En: *Journal of neurophysiology* 104.6 (2010), págs. 3312-3322.

[8] J. D. Hunter. "Matplotlib: A 2D graphics environment". En: *Computing in Science & Engineering* 9.3 (2007), págs. 90-95. DOI: 10.1109/MCSE.2007.55.

[9] Stuart A Kauffman y col. *The origins of order: Self-organization and selection in evolution*. Oxford University Press, USA, 1993.

[10] Stuart A Kauffman y Sonke Johnsen. "Coevolution to the edge of chaos: coupled fitness landscapes, poised states, and coevolutionary avalanches". En: *Journal of theoretical biology* 149.4 (1991), págs. 467-505.

[11] Christopher Langton. *Computation at the edge of chaos: Phase transition and emergent computation*. Inf. téc. Los Alamos National Lab., NM (USA), 1990.

[12] Yann LeCun y col. "A tutorial on energy-based learning". En: (2006).

[13] Robert Legenstein y Wolfgang Maass. "Edge of chaos and prediction of computational performance for neural circuit models". En: *Neural networks* 20.3 (2007), págs. 323-334.

[14] Melanie Mitchell, Peter Hraber y James P Crutchfield. "Revisiting the edge of chaos: Evolving cellular automata to perform computations". En: *arXiv preprint adap-org/9303003* (1993).

[15] Travis E Oliphant. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA, 2006.

[16] J. Ouellette. "A Fundamental Theory to Model the Mind". En: 2017.

[17] Norman H Packard. "Adaptation toward the edge of chaos". En: *Dynamic patterns in complex systems* 212 (1988), pág. 293.

[18] Silvio Salinas. *Introduction to statistical physics*. Springer Science & Business Media, 2001.

[19] Emanuel Todorov, Tom Erez y Yuval Tassa. "Mujoco: A physics engine for model-based control". En: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, págs. 5026-5033.

[20] *Ubuntu official site*. `https://ubuntu.com/`. Accessed: 2020-07-01.

Índice de figuras