VOLUNTARIO 1. MODELO DE HOPFIELD DE RED NEURONAL

01/07/2025

Universidad de Granada, Facultad de Ciencias

Grado de Física

ÁNGEL PASTOR REGADES

Índice

Página

1 Introducción y Problema **2**

2 Metodología y Resultados: Programas empleados **3**
2.1 Calculos.cpp . . . 3
2.1.1 Variables e Inputs . . . 4
2.1.2 Outputs . . . 5
2.2 Visualizer.py . . . 5
2.2.1 Variables e Inputs . . . 5
2.2.2 Outputs . . . 5
2.3 Solapamiento.py . . . 6
2.3.1 Variables e Inputs . . . 6
2.3.2 Outputs . . . 6

3 Metodología y Resultados: Actividades **6**
3.1 Solapamiento con un patrón . . . 6
3.2 Variación del solapamiento con la temperatura con un patrón . . . 8
3.3 Solapamiento con varios patrones . . . 10
3.4 Variación del solapamiento con la temperatura con varios patrones . . . 12
3.5 Patrones recordados en función de la cantidad de patrones almacenados . . . 14

4 Conclusiones **15**

5 Bibliografía **15**

1. Introducción y Problema

Esta entrega propone usar el algoritmo de Metrópolis para simular y poner a prueba una red neuronal de Hopfield en función de la temperatura. En especial se nos pide indagar en las variaciones del solapamiento de la red en función del número de configuraciones almacenadas y de la temperatura del sistema. Este factor de solapamiento mide la cercanía entre nuestro estado actual y uno de nuestros patrones.

Comenzamos explicando brevemente los algoritmos y modelos empleados. El algoritmo de Metrópolis se basa en las cadenas de Markov para calcular la probabilidad de que un sistema cambie de un estado a otro. Empezamos eligiendo un estado inicial y un estado al que se pueda acceder desde nuestro estado inicial. Entonces calculamos la probabilidad de que ocurra este cambio de estado en nuestro sistema actual y, empleando una distribución uniforme contínua en el intervalo [0,1], determinamos si ocurre el cambio de estado o no. Si se efectúa el cambio, el sistema adquirirá el nuevo estado cambiado. Si no, nuestro sistema se mantiene en el estado anterior al cambio.

En los sistemas que vamos a estudiar, los elementos se afectan unos a otros a la hora de darse un cambio de estado, y sus respectivos valores tienen distintos pesos para el resto de elementos del sistema. Por sencillez, vamos a tomar nuestra red neuronal como una bidimensional de N por N nodos.

Los cambios de estado que emplearemos serán alteraciones de un nodo a la vez. Sin embargo, debemos darle la oportunidad de cambiar a todos los elementos de nuestro sistema, por lo que definimos nuestra unidad de tiempo como pasos Monte Carlo. Esto equivale a N^2 intentos de cambio de valor de un nodo aleatorio.

Se nos indica en el guion que tenemos condiciones de contorno periódicas, pero como veremos más adelante, se tienen en cuenta todos los nodos de la red a la hora de efectuar un cambio del estado de nuestro sistema, por lo que esta condición de periodicidad no afecta directamente a los cálculos como sí ocurría en el modelo de Ising.

Por otro lado, una red neuronal de Hopfield es aquella diseñada para recuperar o recordar patrones previamente guardados. Es una red simétrica, sin autoconexión pero completamente conectada.

Queremos a lo largo de esta práctica, crear una red neuronal de Hopfield y ponerla a prueba dándole estados ruidosos o incompletos y observando cómo cambia nuestro sistema tras varios pasos Monte Carlo a distintas temperaturas.

La energía de cada uno de nuestros estados viene dada por la siguiente ecuación del Hamiltoniano.

$$H(s) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N}\omega_{nm,kl}s_{n,m}s_{k,l} + \sum_{n=1}^{N}\sum_{m=1}^{N}\theta_{n,m}s_{n,m} \tag{1}$$

Los valores de cada neurona, $s_{n,m}$, serán 0 o 1, aunque también podrían ser -1 y 1. Los términos $\omega_{nm,kl}$ son los pesos sinápticos, los cuales determinan el efecto entre la neurona (n,m) y la neurona (k,l). Estos solo dependen de los patrones guardados por la red, tal que:

$$\omega_{nm,kl} = \begin{cases} \frac{1}{N^2}\sum_{\mu=1}^{P}\left(\xi_{n,m}^{\mu} - a^{\mu}\right)\left(\xi_{k,l}^{\mu} - a^{\mu}\right) & , \quad \text{si } (n,m) \neq (k,l), \\ 0 & , \quad \text{si } (n,m) = (k,l), \end{cases} \tag{2}$$

donde el término a^{μ} es el valor promedio de cada configuración guardada ξ^{μ} de la red, y viene dado por la siguiente expresión:

$$a^{\mu} = \frac{1}{N^2}\sum_{n=1}^{N}\sum_{m=1}^{N}\xi_{n,m}^{\mu}. \tag{3}$$

Los términos $\xi^{\mu}_{n,m}$ se refieren al valor de la neurona (n,m) en la configuración guardada ξ^{μ}. La variable μ va desde 1 hasta P, siendo P el número total de patrones guardados por la red neuronal.

Por último, en el cálculo del hamiltoniano H del sistema, se hace uso de los umbrales de disparo $\theta_{n,m}$, los cuales determinan finalmente la activación de cada neurona. Estos requieren del peso sináptico entre cada neurona para obtenerse.

$$\theta_{n,m} = \frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{N}\omega_{nm,kl}. \tag{4}$$

En cuanto al cálculo del solapamiento, obtendremos su valor en base a la siguiente ecuación:

$$m^{\mu}(\mathbf{s}) = \frac{1}{N^2 a^{\mu}\left(1-a^{\mu}\right)}\sum_{n=1}^{N}\sum_{m=1}^{N}\left(\xi^{\mu}_{n,m}-a^{\mu}\right)\left(s_{n,m}-a^{\mu}\right). \tag{5}$$

Esta emplea los valores extraídos de un patrón y del sistema para devolvernos el factor de aproximación entre ambos estados.

Vamos a empezar con temperaturas bajas analizando cómo el solapamiento de una red neuronal con un solo patrón es afectado por la forma del sistema original. Empezaremos la red en un estado aleatorio y en un estado próximo en solapamiento al patrón guardado de la red, para ver cómo es afectado el solapamiento con el estado inicial. Luego cambiaremos también la temperatura para ver su efecto en ambos casos.

Posteriormente, estudiaremos estos mismos factores para una red neuronal con varios patrones guardados.

Por último, vamos a comprobar cuán eficaz es la red neuronal recordando patrones a medida que la cantidad de estos aumente.

2. Metodología y Resultados: Programas empleados

Comenzando el análisis del problema, la variable omega que almacena el peso sináptico entre partículas almacena una cantidad de datos del orden N^4, por lo que a medida que el tamaño de nuestro sistema se agranda, iría ocupando mucho espacio en la memoria. Optamos por crear una función llamada omega que calcule y devuelva en el momento el valor de $\omega_{nm,kl}$, con el fin de no desbordar la memoria del ordenador.

Posteriormente nos percatamos de que dentro de cada paso Monte Carlo, calcular el Hamiltoniano del nuevo estado desde cero realizaría centenares de cálculos innecesarios, pues la interacción entre dos partículas que no han cambiado su valor de un estado a otro no varia, y su influencia en el Hamiltoniano es igual antes que después. Decidimos calcular completamente el Hamiltoniano solo una vez, para nuestro estado inicial. Luego, cuando cambie el estado de una neurona del sistema, quitaremos la influencia de esa neurona en el Hamiltoniano y añadiremos su nueva influencia con su nuevo valor en el nuevo estado al Hamiltoniano. Esto reduce drásticamente el tiempo de cómputo del programa.

Hemos creado 3 programas principales para estudiar y poner a prueba las magnitudes requeridas por el guión. Vamos a explicar las características de estos uno por uno para facilitar la reproducción de los resultados.

2.1. Calculos.cpp

Se trata de un programa de C++ donde se realizan todos los cálculos del problema. Este devuelve ficheros con el formato necesario para representar los estados y los solapamientos del sistema tras

cada paso Monte Carlo en los otros programas creados.

2.1.1. Variables e Inputs

Aquí enumero los valores y nombres que deben proporcionarse a este programa antes de ejecutarlo para que devuelva los datos que buscamos y no sobrescriba los ficheros erróneamente. De todas formas, el programa cuenta con un recordatorio en el terminal antes de ejecutarse que permite abortarlo si no se han cambiado los nombres de los ficheros correctamente.

- Variable **reps**: Comienza como valor 1, y solo cambiaremos su valor para el último ejercicio. En vez de crear un nuevo programa, cambiar este valor a un número entero distinto de 1 hace que el programa efectúe los cálculos necesarios para la actividad 4.
- Variable **inicial**: Sirve para elegir los valores iniciales de la red neuronal.

  inicial = 1 inicia todos los nodos con valor 1.

  inicial = 0 inicia todos los nodos con valor 0.

  inicial = -1 inicia el valor de cada nodo aleatoriamente entre 0 o 1.

  Cualquier otro valor entero hace que los valores de los nodos sean los almacenados en el archivo a elección, por lo que esta será la opción a elegir cuando queramos estudiar la evolución del sistema comenzando desde un estado premeditado.
- Variable **T**: Sirve para elegir la temperatura del sistema.
- Variable **Monte_Carlo**: Su valor será la cantidad de pasos Monte Carlo que efectuará el programa.
- Variable **N**: Su valor será el número de neuronas por lado de nuestro sistema bidimensional de N^2 neuronas.
- Variable **P**: Debe tener como valor el número de patrones almacenados del archivo de patrones que queramos usar. Cada archivo de patrones tiene un número de patrones asociado, por lo que este valor depende directamente del archivo de patrones elegido.
- String **nombre_inicial**: Sirve para elegir el nombre del archivo txt que almacenará (en los casos inicial = 1, 0, -1) o que tiene almacenado (en el caso inicial $\neq$ 1, 0, -1) el estado inicial del sistema.

  Cabe recalcar que el programa sobrescribe el estado inicial en el archivo inicial, por lo que si no queremos machacar el nuevo estado inicial en un archivo sin quererlo, recomendamos cambiar nombre_inicial a ”predeterminado.txt”.
- String **ruta_fichero_sistema_inicial**: Esta variable debe contener la ruta de la ubicación en nuestro dispositivo donde se tienen guardados y donde se guardarán los archivos de estados iniciales. Solo se debe rellenar una vez.
- String **nombre_configs**: Sirve para elegir el archivo txt que tiene los patrones guardados de la red.
- String **ruta_fichero_config_guardadas**: Esta variable debe contener la ruta de la ubicación en nuestro dispositivo donde se encuentran los archivos de configuraciones guardadas. Solo se debe rellenar una vez.
- String **nombre_results**: Sirve para elegir el nombre del archivo txt donde se guarda el estado de la red al acabar cada paso Monte Carlo.
- String **ruta_fichero_resultados**: Esta variable debe contener la ruta de la ubicación en nuestro dispositivo donde se guardarán los archivos con los estados de la red tras cada paso Monte Carlo. Solo se debe rellenar una vez.

Todas estas variables se encuentran en las primeras 33 líneas de código y agrupadas todo lo posible para su fácil acceso. Recomendamos dejar escrito en las líneas de código 31, 32 y 33 las rutas de las ubicaciones donde se guarda o donde se encuentra cada fichero, pues esto no deberá cambiarse a lo largo del problema.

Justo al ejecutar el programa se nos pregunta en el terminal si hemos elegido correctamente los nombres de los ficheros para evitar sobrescribir sin querer en un fichero de un estado inicial o en un fichero de resultados. Escribir la letra ”n” en el terminal aborta el programa para que podamos modificar los nombres de los ficheros. Escribir cualquier otra letra hace comenzar los cálculos y la creación de ficheros.

2.1.2. Outputs

El programa escribirá en el terminal en qué paso Monte Carlo se encuentra en los cálculos para ver su progreso. Si se ha elegido el valor de la variable inicial = 1, 0 o -1, el fichero llamado ”nombre_inicial” guardará el estado inicial del sistema, machacando el estado que estuviese guardado ahí antes si lo hubiera.

El programa crea un archivo txt con el nombre ”nombre_results” en ”ruta_fichero_resultados” que contiene todos los estados del sistema a lo largo de los pasos Monte Carlo. Se genera también otro txt con el nombre solap.”nombre_results” también en ”ruta_fichero_resultados”, donde se habrán almacenado los solapamientos de nuestro sistema con los patrones guardados en ”nombre_configs” a lo largo de cada paso Monte Carlo.

Cuando la variable reps sea distinta a 1, el programa tendrá un funcionamiento distinto, pues solo mostrará en el terminal los patrones que se recordaron tras comenzar el sistema en el mismo estado de cada patrón.

2.2. Visualizer.py

Este programa sirve para observar en formato gif la evolución del sistema a lo largo del tiempo.

2.2.1. Variables e Inputs

Para ejecutar correctamente este programa, se deben tener en cuenta las siguientes variables y archivos.

- **N**: Igual que en el programa Calculos.cpp, se trata del número de neuronas por lado de la red neuronal bidimensional que queramos visualizar.
- **nombre_results**: Igual que en el programa Calculos.cpp, se debe escribir el nombre del archivo que contenga el estado del sistema en cada paso Monte Carlo. En este programa no se sobrescriben los resultados, sino que se leen para representarlos.

  El fichero llamado ”nombre_results” debe encontrarse en la misma carpeta que el programa Visualizer.py para que este lo encuentre y lo utilice. Esto debe tenerse en cuenta a la hora de elegir ”ruta_fichero_resultados”.
- **fps**: Esta variable representa los frames que se muestran por segundo en el gif.

Las tres variables se encuentran en las 10 primeras líneas de código.

2.2.2. Outputs

El programa Visualizer.py no escribe nada en el terminal, ni modifica ni genera ficheros. Lo único que hace es mostrar el gif con la evolución del sistema en una ventana del ordenador.

2.3. Solapamiento.py

Este último programa se emplea para visualizar en una gráfica los distintos solapamientos que tiene una red neuronal con los patrones guardados, y cómo estos cambian con el tiempo.

2.3.1. Variables e Inputs

La única variable que debemos modificar del programa para ejecutarlo es el nombre del fichero que queremos representar, **nombre_solap**. Se encuentra en la sexta línea de código.

Como ocurre con el anterior programa, los ficheros con los datos que queramos representar se deben encontrar en la misma ubicación que el programa Solapamiento.py. Si la elección de ”ruta_fichero_resultados” fue correcta y se visualizan los gif del programa Visualizer.py, no debería haber problema en ejecutar las gráficas de Solapamiento.py, puesto que ambos programas se encuentran en la misma ubicación y la ruta para los archivos que almacenan el solapamiento es la misma que para los archivos que contienen los estados a lo largo del tiempo.

2.3.2. Outputs

Este programa genera un pdf llamado igual que nuestra variable nombre_solap, pero sin la extensión txt, para que no quepa duda sobre qué gráfica le corresponde a cada archivo txt. El pdf se guarda en ”ruta_fichero_resultados”.

3. Metodología y Resultados: Actividades

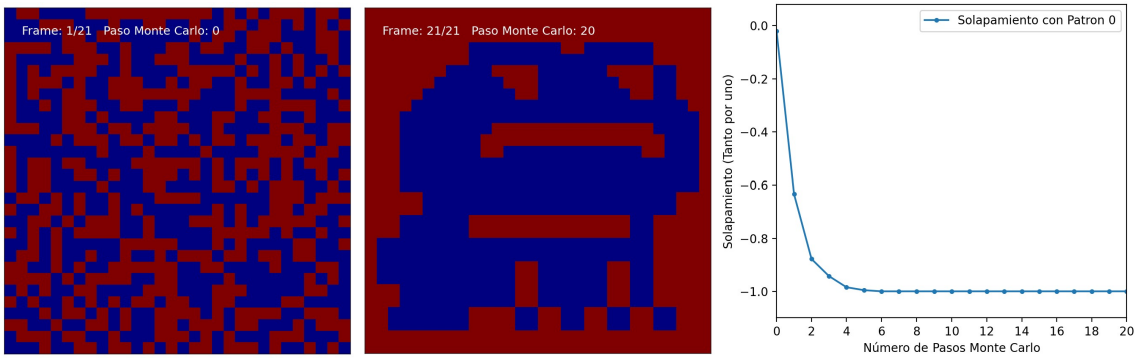
3.1. Solapamiento con un patrón

Empezamos partiendo de un sistema inicial aleatorio, con un solo patrón almacenado y temperatura de 10^{-4}. Rellenamos las siguientes variables para este caso en el programa Calculos.cpp:

inicial	T	Monte_Carlo	N	P	nombre_inicial	nombre_configs	nombre_results
-1	0.0001	20	30	1	”predeterminado.txt"	”config1.txt"	”result1.1.txt"

Tabla 1: Valores de las variables requeridas por el sistema para el solapamiento con un patrón.

Obtenemos los siguientes resultados, visibles empleando los programas Visualizer.py y Solapamiento.py.



(a) Estado inicial del sistema. (b) Estado final del sistema. (c) Solapamiento del sistema con el tiempo.

Fig. 1: Estudio del solapamiento de una red neuronal a baja temperatura con un solo patrón guardado.

A lo largo del gif, debido a la baja temperatura del sistema, vemos poca variación en el valor de las neuronas. Cuando una neurona alcanza su posición de menor energía respecto al resto de neuronas, en ningún momento tiene la energía suficiente como para que sea probable un cambio de valor que no favorezca a reducir el Hamiltoniano. Podemos ver este hecho también en la forma de la curva del solapamiento con el tiempo, pues en todo momento se acerca su valor a -1 y a partir del sexto paso Monte Carlo, el valor se mantiene constantemente en -1 sin perturbaciones.

Cabe recalcar que estos valores, pese a ser negativos, son correctos, porque por cada patrón guardado que tengamos, nuestro sistema puede aproximarse también a su antipatrón, como ocurre en este caso.

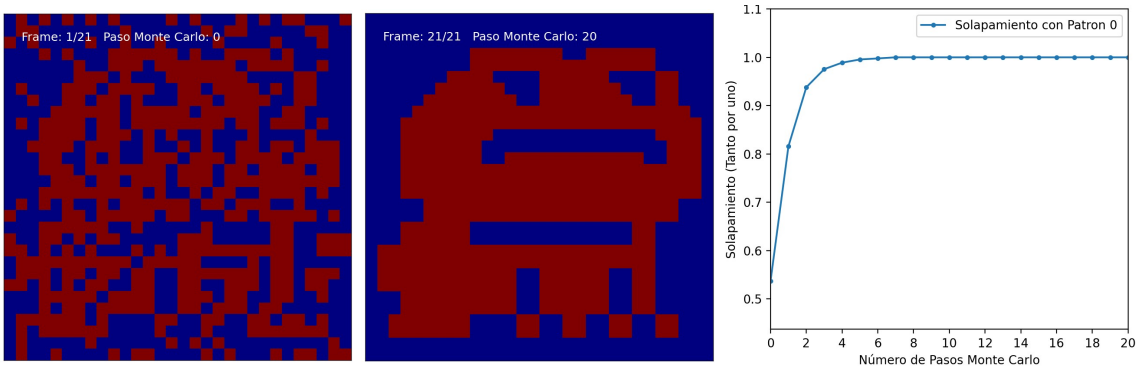
Por último, notemos que en el comienzo del sistema, este se parecía poco al patrón almacenado, como se puede comprobar en el factor de solapamiento próximo a 0.

Comprobemos ahora cómo emplear un estado próximo en solapamiento al patrón deseado influye a la evolución de este. Cambiamos los valores de las variables tal que:

inicial	T	Monte_Carlo	N	P	nombre_inicial	nombre_configs	nombre_results
2	0.0001	20	30	1	”patron1alter.txt"	”config1.txt"	”result1.2.txt"

Tabla 2: Valores de las variables requeridas por el sistema para el solapamiento con un patrón, comenzando con un estado próximo al patrón.

Los resultados obtenidos son los siguientes visibles en la figura (2).



(a) Estado inicial del sistema. (b) Estado final del sistema. (c) Solapamiento del sistema con el tiempo.

Fig. 2: Estudio del solapamiento de una red neuronal a baja temperatura con un solo patrón guardado e iniciando el sistema en un estado próximo al del patrón.

Podemos ver cómo la recuperación del patrón ocurre casi a la vez que cuando elegimos un patrón aleatorio. El programa pasa por todas las neuronas del sistema proponiéndoles un cambio de valor. Si solamente se lo propusiera a las neuronas con valor distinto al correspondiente en el patrón más próximo, al iniciar la red neuronal en un estado próximo al patrón se tardaría menos pasos Monte Carlo en alcanzar solapamiento completo. En nuestro caso es coherente que tarde el mismo tiempo porque, independientemente del sistema inicial, el programa tarda aproximadamente lo mismo en proponer el cambio de valor a cada una de las neuronas.

Esta vez el patrón que ha recordado la red ha sido el verdadero. Sin embargo, el mayor cambio con respecto a la primera prueba ocurre al inicio del sistema, donde podemos ver que comienza con un solapamiento mayor a 0.5 en vez de próximo a 0. Tiene sentido que sea así, ya que el patrón de la figura (2a) se ha formado en base al de la figura (2b) cambiando varios de los valores de las neuronas manualmente.

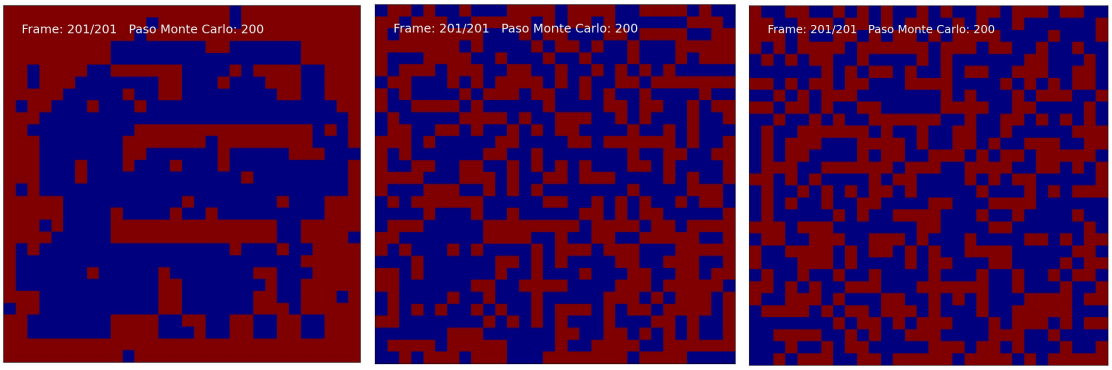
3.2. Variación del solapamiento con la temperatura con un patrón

Pasemos ahora a estudiar este mismo ejemplo pero aumentando la temperatura. Primero comenzaremos el sistema en un estado aleatorio. Usaremos los 3 siguientes conjuntos de valores para nuestras variables en el programa Calculos.cpp:

inicial	T	Monte_Carlo	N	P	nombre_inicial	nombre_configs	nombre_results
-1	0.01	200	30	1	"predeterminado.txt"	”config1.txt"	”result1.1_T1.txt"
-1	0.03	200	30	1	"predeterminado.txt"	”config1.txt"	”result1.1_T2.txt"
-1	0.04	200	30	1	"predeterminado.txt"	”config1.txt"	”result1.1_T3.txt"

Tabla 3: Valores de las variables iniciales del sistema para los casos a distintas temperaturas.

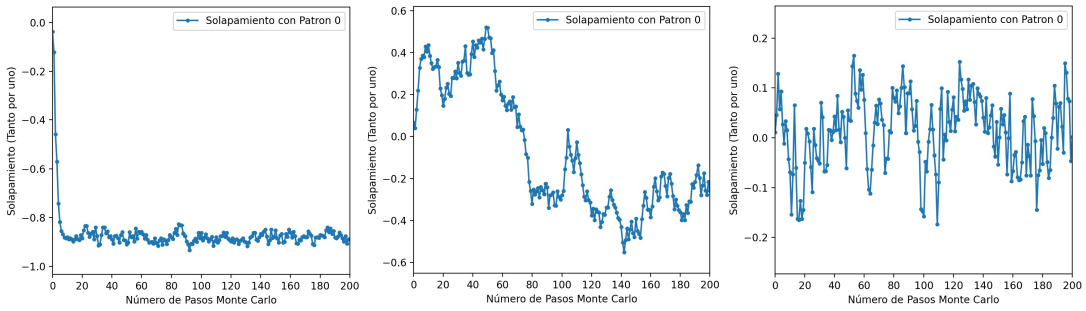
Hemos elegido estos valores de temperatura ya que consideramos que están alrededor del caso límite y representan bien las distintas situaciones por las que puede pasar el sistema. Además, hemos aumentado enormemente el número de pasos Monte Carlo del sistema para darle tiempo a que evolucione en los 3 casos. Los resultados del solapamiento han sido los siguientes, tras ejecutar Visualizer.py con los archivos correspondientes.



(a) Estado final para T = 0.01. (b) Estado final para T = 0.03. (c) Estado final para T = 0.04.

Fig. 3: Estados finales de una red neuronal tras 200 pasos Monte Carlo a diferentes temperaturas con un solo patrón guardado.

Podemos apreciar que, a medida que aumenta la temperatura, la proximidad al patrón guardado se va perdiendo. A bajas temperaturas son solo unas pocas neuronas las que alcanzan valores que no minimizan el Hamiltoniano y que distan de su valor en el patrón esperado. Pero conforme aumentamos la temperatura, más neuronas cambian su valor aleatoriamente hasta ser irreconocible el patrón. Analizamos ahora estos cambios desde la perspectiva del solapamiento.



(a) Solapamiento para T = 0.01. (b) Solapamiento para T = 0.03. (c) Solapamiento para T = 0.04.

Fig. 4: Solapamientos de una red neuronal durante 200 pasos Monte Carlo a diferentes temperaturas con un solo patrón guardado.

El primer caso de la figura (4a) es el más parecido al que podíamos encontrar en la figura (1c). Sin embargo, hay presentes fluctuaciones aleatorias durante toda la evolución del sistema, debido a las neuronas que, por tener energía suficiente, toman valores que se alejan del patrón más cercano. En este caso las fluctuaciones no son lo suficientemente frecuentes como para que el sistema sufra una inversión y cambie el patrón más cercano a la red en cada momento.

Sin embargo, esto sí ocurre en el caso con T = 0.03. Se puede ver que, aunque el patrón se asemeje inicialmente al patrón guardado, a partir del paso Monte Carlo 70, el sistema sufre una inversión y pasa a aproximarse al antipatrón. Ese efecto ocurre porque suficientes neuronas cambian sus valores a la vez debido a las fluctuaciones como para que el solapamiento pase de valores positivos a negativos o viceversa. De forma aleatoria, el patrón se aproxima más al patrón o al antipatrón durante ciertos periodos.

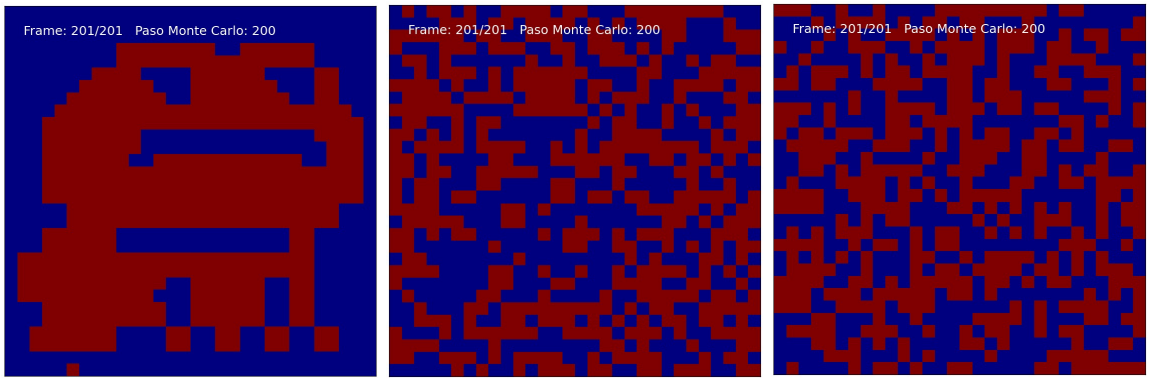
Por último, para temperaturas T = 0.04 y superiores, veremos solapamientos como el de la figura (4c). En este caso, el solapamiento se encuentra siempre en torno al valor 0, ya que las neuronas están en constante cambio aleatorio de sus valores y no se mantienen en la configuración que minimice el Hamiltoniano. Las inversiones antes mencionadas ocurren con tanta frecuencia que no le da tiempo al sistema a acercarse lo suficiente a la configuración de un patrón como para determinar que la recuerda.

Estudiemos el caso en el que se inicia el sistema en un estado próximo al patrón. Emplearemos el mismo estado inicial visible en la figura (2a). Para ello, damos los siguientes valores al programa Calculos.cpp:

inicial	T	Monte_Carlo	N	P	nombre_inicial	nombre_configs	nombre_results
2	0.01	200	30	1	”patron1alter.txt"	”config1.txt"	”result1.2_T1.txt"
2	0.03	200	30	1	”patron1alter.txt"	”config1.txt"	”result1.2_T2.txt"
2	0.04	200	30	1	”patron1alter.txt"	”config1.txt"	”result1.2_T3.txt"

Tabla 4: Valores de las variables iniciales del sistema para los casos a distintas temperaturas pero iniciando el sistema en un estado próximo al patrón.

Los resultados representados en nuestros programas Visualizer.py son los siguientes:

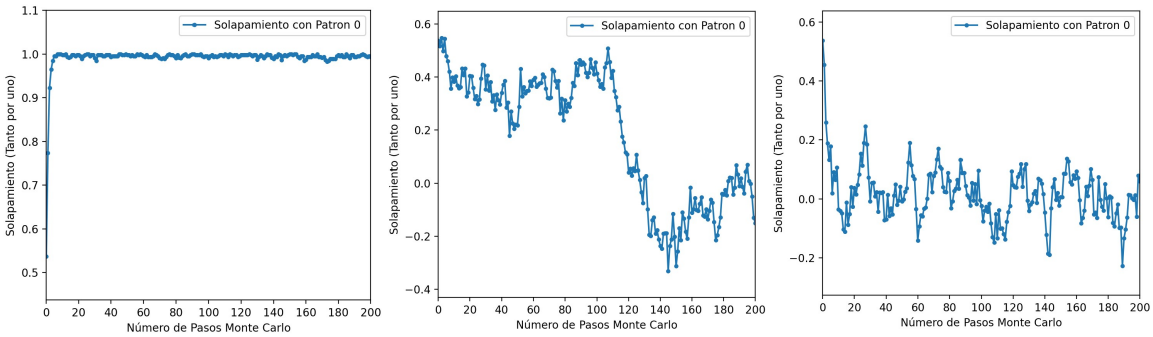


(a) Estado final para T = 0.01. (b) Estado final para T = 0.03. (c) Estado final para T = 0.04.

Fig. 5: Estados finales de una red neuronal tras 200 pasos Monte Carlo a diferentes temperaturas con un solo patrón guardado e iniciando el sistema en un estado cercano al del patrón guardado.

Se puede apreciar que para una temperatura no demasiado alta en la que no ocurran inversiones del sistema, comenzar en un estado próximo al del patrón guardado ayuda al sistema a recordar dicho patrón, puesto que aunque la temperatura sea la misma que para el caso anterior, hay muchas menos neuronas con valores contrarios al óptimo. Se puede notar la mejora comparando la nueva figura (5a) con la anterior (3a).

Por otro lado, los sistemas a temperaturas T = 0.03 y T = 0.04 se comportan de la misma manera que sus contrapartes en el apartado anterior, siendo apenas reconocible, o irreconocible en el caso de T = 0.04, el patrón a lo largo del tiempo.



(a) Solapamiento para T = 0.01. (b) Solapamiento para T = 0.03. (c) Solapamiento para T = 0.04.

Fig. 6: Solapamientos de una red neuronal durante 200 pasos Monte Carlo a diferentes temperaturas con un solo patrón guardado e iniciando el sistema en un estado próximo al del patrón guardado.

Estudiando los solapamientos del sistema para estos casos vemos cómo el solapamiento para la primera temperatura es mucho más próximo a 1 que cuando se usaba un estado inicial aleatorio. Se mantienen las perturbaciones que mencionamos antes para el caso de esta temperatura, pero el valor oscila alrededor de la unidad (sin superarla en ningún momento) en vez de alrededor de 0.9 como ocurría en la figura (4a).

En cuanto a las otras dos temperaturas estudiadas, muestran curvas muy similares a las del sistema iniciado con valores aleatorios. La única diferencia es que, en vez de comenzar en solapamiento 0, comienzan en un solapamiento mayor a 0.5, pero pronto vuelven a seguir la misma tendencia comentada para la figura (4). El estado inicial, por lo tanto, no afecta a la recuperación de un patrón si nos encontramos en temperaturas elevadas donde ocurren inversiones del solapamiento.

3.3. Solapamiento con varios patrones

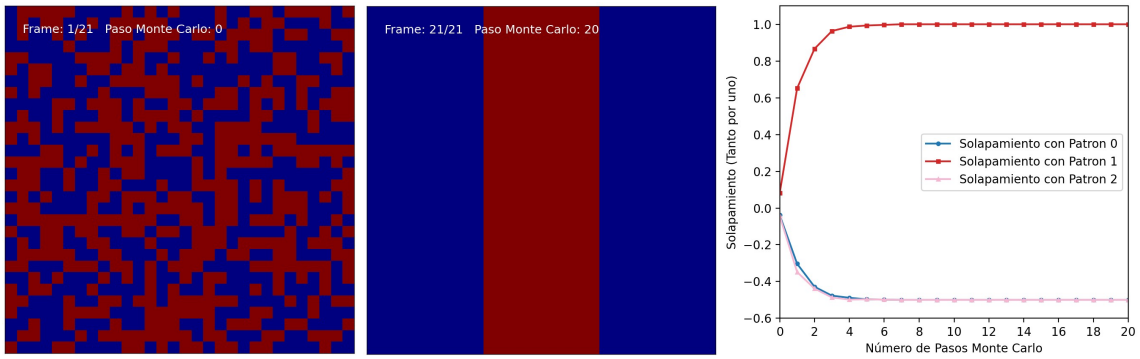
Hemos optado por patrones sencillos y ortogonales entre sí para este apartado, con tal de evitar la aparición de estados espúreos entre patrones y para que todos tengan la misma probabilidad de ser recordados por el sistema. En las primeras pruebas, usamos los dígitos del 0 al 9, pero el sistema siempre se estancaba en una mezcla de varios patrones (un mínimo espúreo), ya que eran los más probables y compartían muchos valores de nodos. Sin embargo, usando rectángulos verticales lo más ortogonales entre sí posibles, podemos obtener resultados que nos den más información sobre el solapamiento de la red neuronal.

Al igual que en el apartado 3.1, investigamos cómo cambia con el tiempo el sistema comenzando con un estado aleatorio. Cambiamos los valores de las variables de Calculos.cpp:

inicial	T	Monte_Carlo	N	P	nombre_inicial	nombre_configs	nombre_results
-1	0.0001	20	30	3	”predeterminado.txt"	”config2.txt"	”result2.1.txt"

Tabla 5: Valores de las variables requeridas por el sistema para el solapamiento de varios patrones.

Obtenemos los siguientes resultados, visibles empleando los programas Visualizer.py y Solapamiento.py.



(a) Estado inicial del sistema. (b) Estado final del sistema. (c) Solapamiento del sistema con el tiempo.

Fig. 7: Estudio del solapamiento de una red neuronal a baja temperatura con varios patrones guardados.

El comportamiento de la red es el mismo que para un solo patrón, solo que en este caso, en vez de decantarse solo por un patrón y su antipatrón guardado, la red puede tomar 6 estados finales distintos con un solapamiento de valor 1 en módulo.

El patrón al que más se asemejaba la red en su comienzo es el que acaba siendo completamente recordado, debido a la baja temperatura que hace que el sistema no explore libremente todos los estados posibles. Y los valores del solapamiento de los otros dos patrones se deben a los valores que comparten con el patrón recordado.

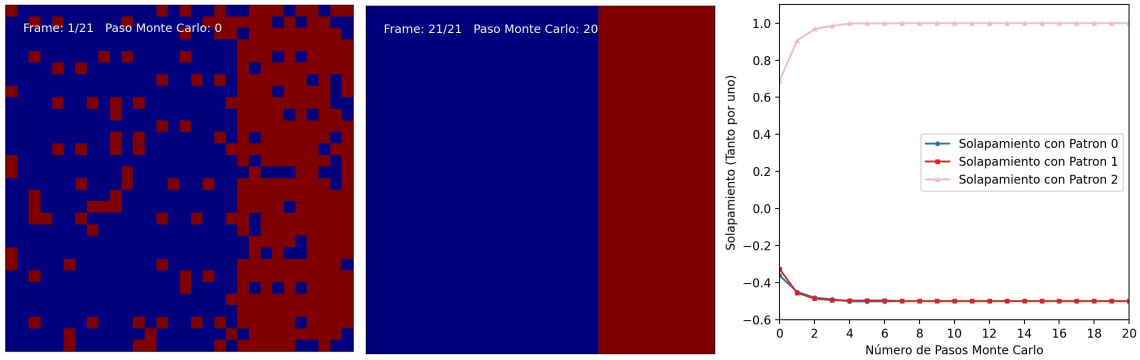
Todo esto es coherente con nuestro ejemplo de un solo patrón y mantiene la explicación que dimos anteriormente sobre la baja temperatura y la carencia de fluctuaciones de la red que conlleva.

Pasamos a estudiar el solapamiento comenzando el sistema en un estado próximo al patrón. La configuración del programa Calculos.cpp que usaremos es la siguiente:

inicial	T	Monte_Carlo	N	P	nombre_inicial	nombre_configs	nombre_results
2	0.0001	20	30	3	”patron2alter.txt"	”config2.txt"	”result2.2.txt"

Tabla 6: Valores de las variables requeridas por el sistema para el solapamiento de varios patrones, iniciando el sistema en un estado próximo a uno de ellos.

De los cuales obtenemos los siguientes resultados:



(a) Estado inicial del sistema. (b) Estado final del sistema. (c) Solapamiento del sistema con el tiempo.

Fig. 8: Estudio del solapamiento de una red neuronal a baja temperatura con varios patrones guardados y comenzada en un estado próximo a uno de los patrones.

Tal y como ocurría con un patrón, se obtienen los mismos solapamientos usando un inicio alea-

torio y usando otro próximo a uno de los patrones, aunque esta vez el patrón recordado ha sido otro. Lo único que cambia en este caso es que el valor inicial del solapamiento es más cercano a su valor final, por la misma razón que cuando solo estudiábamos la red con un patrón almacenado.

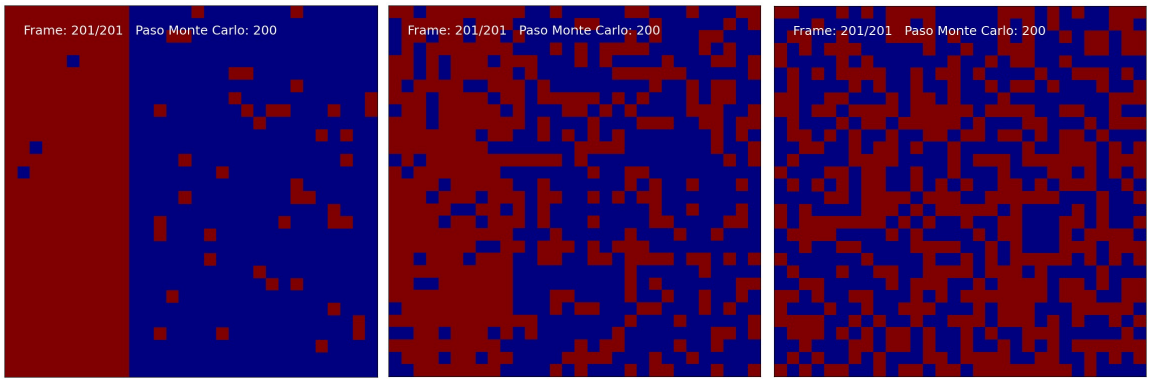
3.4. Variación del solapamiento con la temperatura con varios patrones

Pasamos a estudiar el efecto de la temperatura en nuestras redes neuronales. Emplearemos los siguientes valores en Calculos.cpp en el caso de estado inicial aleatorio.

inicial	T	Monte_Carlo	N	P	nombre_inicial	nombre_configs	nombre_results
-1	0.02	200	30	1	”predeterminado.txt"	”config2.txt"	”result2.1_T1.txt"
-1	0.036	200	30	1	”predeterminado.txt"	”config2.txt"	”result2.1_T2.txt"
-1	0.05	200	30	1	”predeterminado.txt"	”config2.txt"	”result2.1_T3.txt"

Tabla 7: Valores de las variables iniciales del sistema para los casos a distintas temperaturas.

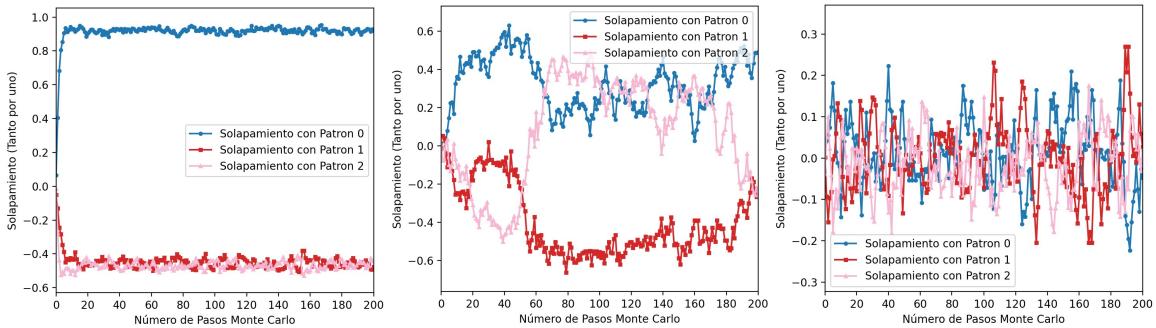
Al ser otros patrones, tuvimos que cambiar un poco las temperaturas respecto al apartado anterior para obtener los resultados límite más óptimos. Los estados finales representados en nuestros programas Visualizer.py son los siguientes:



(a) Estado final para T = 0.02. (b) Estado final para T = 0.036. (c) Estado final para T = 0.05.

Fig. 9: Estados finales de una red neuronal tras 200 pasos Monte Carlo a diferentes temperaturas con varios patrones guardados e iniciando el sistema en un estado aleatorio.

Y sus solapamientos se pueden visualizar en Solapamiento.py, resultando en las siguientes gráficas:



(a) Solapamiento para T = 0.02. (b) Solapamiento para T = 0.036. (c) Solapamiento para T = 0.05.

Fig. 10: Solapamientos de una red neuronal durante 200 pasos Monte Carlo a diferentes temperaturas con varios patrones guardados e iniciando el sistema en un estado aleatorio.

En general, la misma explicación de este ejercicio para un solo patrón puede ser usada para explicar los fenómenos de este caso con 3 patrones. Al igual que antes, para la primera temperatura hemos elegido un valor de esta que haga oscilar el valor del solapamiento, pero donde no ocurren inversiones. El solapamiento sigue la misma tendencia de no variar en gran medida su valor cuando el sistema recuerda un patrón.

Para la segunda temperatura sí se pueden apreciar inversiones en el gif y en el solapamiento. Sin embargo, en este caso con varios patrones, las inversiones pueden ocurrir no solo para el antipatrón del patrón más próximo del sistema, sino que también puede derivar a otros estados más próximos a otros patrones. Un ejemplo de esto se puede observar en torno al paso Monte Carlo 60 de la figura (10b).

La tercera temperatura se comporta de la misma manera para varios patrones, sin que prevalezca por mucho tiempo ninguno de los solapamientos, puesto que la elevada temperatura del sistema favorece la aleatoriedad de los valores de las neuronas y estas no se estabilizan en ningún patrón guardado con el tiempo.

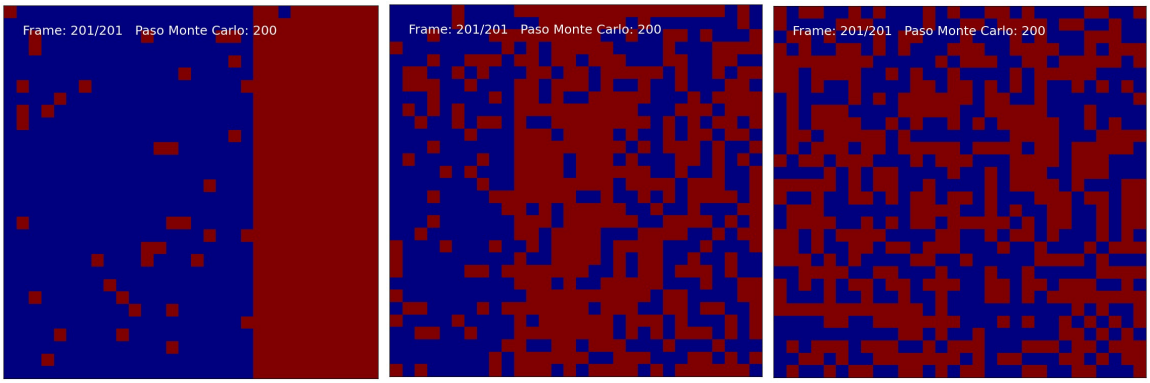
Por último en este apartado, comprobemos qué le ocurre al sistema si lo comenzamos con un estado próximo al tercer patrón, el mismo que se puede ver en la figura (8a) del apartado 3.3.

Rellenamos las variables de Calculos.cpp con los siguientes valores:

inicial	T	Monte_Carlo	N	P	nombre_inicial	nombre_configs	nombre_results
2	0.02	200	30	1	”patron2alter.txt"	”config2.txt"	”result2.2_T1.txt"
2	0.036	200	30	1	”patron2alter.txt"	”config2.txt"	”result2.2_T2.txt"
2	0.05	200	30	1	”patron2alter.txt"	”config2.txt"	”result2.2_T3.txt"

Tabla 8: Valores de las variables iniciales del sistema para los casos a distintas temperaturas, iniciando el sistema en un estado cercano al del uno de los patrones.

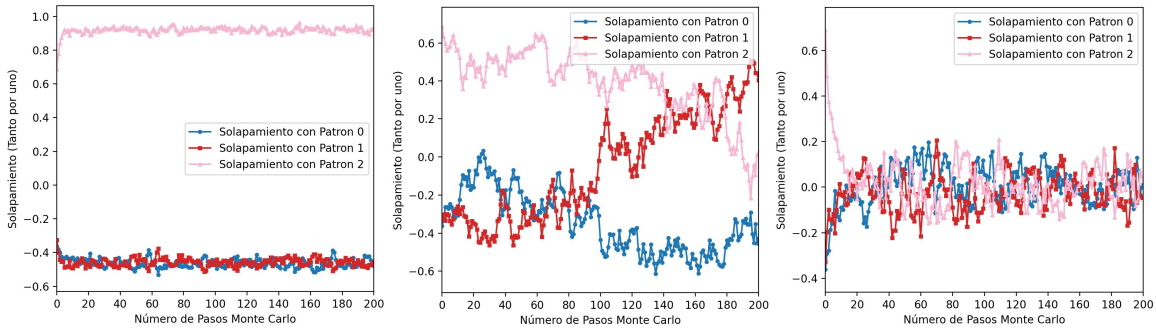
Empleamos Visualizer.py y Solapamiento.py para representar los resultados obtenidos:



(a) Estado final para T = 0.02. (b) Estado final para T = 0.036. (c) Estado final para T = 0.05.

Fig. 11: Estados finales de una red neuronal tras 200 pasos Monte Carlo a diferentes temperaturas con varios patrones guardados e iniciando el sistema en un estado próximo al de uno de los patrones.

Los resultados finales tienen las mismas características que para el apartado anterior. Cabe destacar que el caso con la menor de las temperaturas es el único que se ve influenciado directamente por el estado inicial, pues este determina el patrón más próximo al estado en el que acaba el sistema al no haber inversiones. Para las otras dos temperaturas, tener un estado inicial u otro no determina nada con seguridad, puesto que el estado final del sistema y el patrón al que más se aproxime son aleatorios en mayor o menor medida.



(a) Solapamiento para T = 0.02. (b) Solapamiento para T = 0.036. (c) Solapamiento para T = 0.05.

Fig. 12: Solapamientos de una red neuronal durante 200 pasos Monte Carlo a diferentes temperaturas con varios patrones guardados, iniciando el sistema en un estado cercano al de uno de los patrones.

La única diferencia entre el comportamiento del solapamiento en las gráficas de la figura (12) y en la anterior figura (11) es el valor con el que comienzan. Por lo demás, los solapamientos presentan las mismas características: pequeñas perturbaciones pero sin inversión para la menor temperatura, inversiones cada cierta cantidad de pasos para la segunda temperatura y valores oscilando alrededor de 0 para la tercera temperatura.

3.5. Patrones recordados en función de la cantidad de patrones almacenados

En cuanto a esta actividad final, se nos pedía calcular cómo disminuye el número de patrones recordados en función de la cantidad de patrones guardados. Lo que debíamos hacer es elegir una cantidad de patrones almacenados y comenzar el sistema con cada uno de ellos. Tras varios pasos Monte Carlo, comprobamos que todos ellos se recuerden, o lo que es lo mismo, que el factor de solapamiento de la red para ese patrón sea mayor a 0.75 en valor absoluto.

Tal y como se nos indica en el guión, los patrones de este ejercicio se han generado de forma aleatoria. Cien patrones se encuentran almacenados en el fichero llamado ”config3.txt”, de los cuales, usando la variable P elegiremos la cantidad de estos que queramos usar. Debemos notar también que no nos hace falta un archivo con el estado inicial del sistema para ejecutar el programa, por lo que recomendamos dejar la variable nombre_inicial con el valor ”predeterminado.txt” para evitar sobrescribir en ella erróneamente.

El resto de variables del programa Calculos.cpp deben quedar de la siguiente manera para que el programa nos diga si se ha recuperado cada patrón:

reps	inicial	T	Monte_Carlo	N	nombre_configs	nombre_results
2	2	0.0001	20	20	”config3.txt"	”result3.txt"

Tabla 9: Valores de las variables iniciales del sistema para la actividad 4.

Hemos elegido 20 pasos Monte Carlo debido a la baja temperatura del sistema, que como hemos visto en apartados anteriores, hace que el primer patrón al que nos acerquemos sea el estado más probable al que evolucione el sistema. Esto unido a que comenzamos el sistema con solapamiento 1 en uno de los patrones cada vez, nos hace pensar que con 20 pasos Monte Carlo podemos tener suficiente información de si se recuerda el patrón estudiado en cada caso o no.

Tras hacer esto y ejecutar el programa para varios valores de P, hemos contado en cada uno los patrones que se recuperan y los que no, consiguiendo así la figura (13) que expone el número de patrones recordados frente al número de patrones almacenados.

La prueba realizada con 55 patrones guardados es la última en la que el sistema consigue recordar todos los patrones de forma eficaz. A partir de dicho valor, tal y como se ve en la figura, ningún

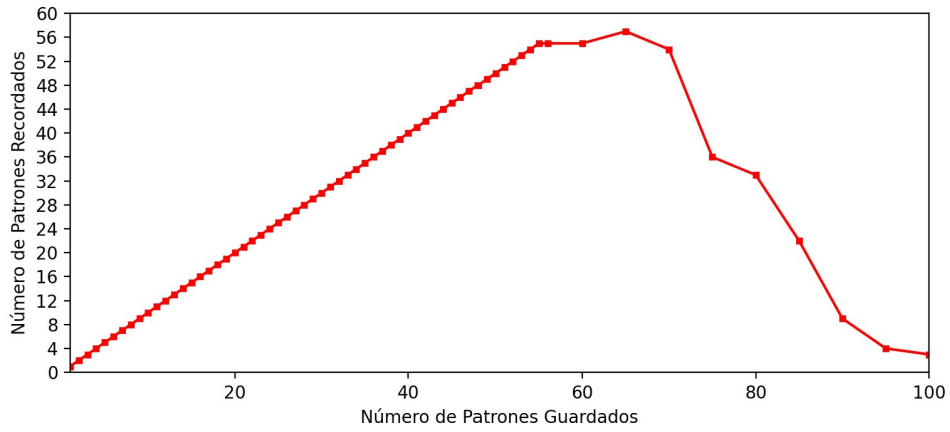


Fig. 13: Patrones recordados frente a patrones almacenados para una red de 20x20 neuronas y temperatura T = 0.0001 tras 20 pasos Monte Carlo.

valor de P consigue que el sistema recuerde todos los patrones guardados. Además, a partir del P = 65, el número de patrones recordados disminuye drásticamente. La fracción máxima que se nos pide, $\alpha_c = \frac{P_c}{N^2}$ devuelve el siguiente valor:

$$\alpha_c = \frac{P_c}{N^2} = \frac{55}{400} = 0{,}1375 \tag{6}$$

4. Conclusiones

En esta serie de actividades propuestas hemos desarrollado una red neuronal de Hopfield funcional a la que podemos enseñarle diversos patrones y esta evoluciona hacia el que más se asemeje.

En un comienzo hemos estudiado la semejanza entre el estado de nuestra red neuronal con el estado de los patrones, el llamado solapamiento. Hemos visto cómo se ve afectado por el estado inicial del sistema y por la temperatura, siendo esta última la variable que más influye en la evolución y la estabilidad del solapamiento a largo plazo. De manera coherente, sistemas con energías más bajas tienden a comportarse de forma más estable y determinista que a temperaturas altas. Todo esto ha sido estudiado tanto para 1 patrón guardado como para 3 diferentes.

Posteriormente nos hemos centrado en aumentar la cantidad de patrones almacenados y ver cómo afecta al solapamiento la cantidad de patrones de nuestro sistema. Hemos encontrado el momento en el que, al aumentar la cantidad de patrones, nuestra cantidad de patrones recordados se aleja de la recta identidad, debido al número limitado de estados posibles para la red y la aparición de estados espúreos por las similitudes de unos patrones guardados con otros.

Por último hemos calculado el punto crítico de nuestra red neuronal a la hora de recordar todos los patrones que le enseñamos, el cual se encuentra cuando se cumple la igualdad de la ecuación (6). En la teoría, esta capacidad crítica para una red de Hopfield debe ser aproximadamente 0.138 (Emina, 2021, pág 6). La nuestra presenta un valor de 0.1375, lo que denota un buen funcionamiento de nuestro sistema completamente conectado.

5. Bibliografía

· Emina, F. (2021). *Mejoramiento de la capacidad de almacenamiento de una red neuronal autoasociativa a través de la conectividad selectiva* (Tesis de grado, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires). Recuperada de

https://hdl.handle.net/20.500.12110/seminario_nFIS000125_Emina