Voluntario Física Computacional

Modelo de Hopfield de red neuronal

Fco Javier Archidona Corrales

Julio de 2022

UNIVERSIDAD
DE GRANADA

Resumen

El objetivo de este trabajo es estudiar el comportamiento del modelo de Hopfield de red neuronal. Se estudia la curva de solapamiento para un patrón inicial completamente aleatorio y otro deformado, comprobando que esta converge más rápidamente a la unidad si se parte de un estado deformado. Además, se varía la temperatura para observar que cuanto más aumenta esta, más desordenado se vuelve el patrón final. Este mismo estudio se aplicará también para tres patrones en lugar de uno. Y por último, se ve como decae la recuperación de la memoria en función del número de patrones.

Repositorio Github donde se incluyen los plt's, todas las gráficas y algunos *gifs*: Repositorio Gnuplot

Índice

1 Introducción

El modelo de Hopfield de red neuronal fue inventado por John Hopfield en 1982. Normalmente se suele utilizar para problemas de optimización, así como para reconocer patrones.
El modelo consta de una matriz bidimensional de neuronas, de dimensión NxN, que en nuestro caso vendrán expresadas por $s_{i,j}$. Estas neuronas se pueden encontrar en dos estados, activas (si $s_{i,j} = +1$), e inactivas (si $s_{i,j} = 0$). Cada neurona está conectada con el resto excepto con ella misma . $\omega_{ij,kl}$ describe el peso de la conexión entre las neuronas (i,j) y (k,l), siendo $\omega_{ij,ij} = 0$. A continuación se muestra una representación gráfica de estas conexiones.

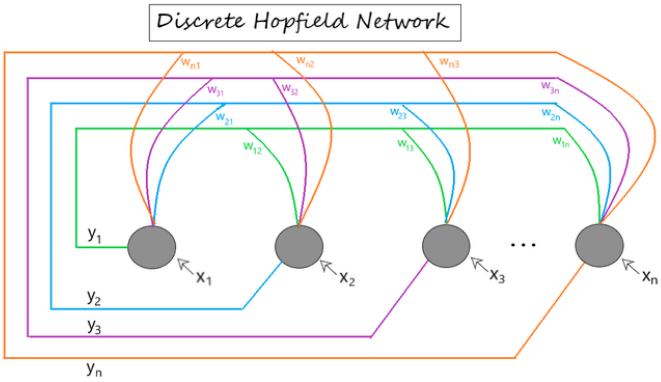


Figura 1: Representación gráfica de red neuronal de Hopfield (imagen de [2])

Este modelo parte de varios patrones (P patrones en total, utilizamos μ para identificar cada uno) previamente establecidos que tratará de recordar, que expresamos como $\xi^{\mu}_{i,j}$. Se define el número de neuronas activas en cada patrón con el parámetro $a^{\mu} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\xi^{\mu}_{i,j}$, y de esta forma tenemos que los pesos sinápticos vienen dados por:

$$\omega_{ij,kl} = \begin{cases} \frac{1}{N^2}\sum_{\mu=1}^{P}(\xi^{\mu}_{i,j} - a^{\mu})(\xi^{\mu}_{k,l} - a^{\mu}), & \text{si } (i,j) \neq (k,l) \\ 0, & \text{si } (i,j) = (k,l) \end{cases} \tag{1.1}$$

Se define además el umbral de disparo $\theta_{i,j} = \frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{N}\omega_{ij,kl}$.

El Hamiltoniano para cada configuración del sistema $s = s(i,j)$ viene dado por:

$$H(s) = -\frac{1}{2}\sum_{i}^{N}\sum_{j}^{N}\sum_{k}^{N}\sum_{l}^{N}\omega_{ij,kl}s_{i,j}s_{k,l} + \sum_{i}^{N}\sum_{j}^{N}\theta_{i,j}s_{i,j} \tag{1.2}$$

Para implementar el método necesitamos calcular el valor de $\triangle H = H(s') - H(s)$. Si en cada iteración se calculase $H(s')$ y $H(s)$ por separado, el programa correría demasiado lento, por lo que en el Anexo hemos desarrollado una expresión más compacta para calcularlo.

Existe otro parámetro que se calcula una vez en cada paso Monte-Carlo, llamado solapamiento $m^{\mu}(s)$, que cuantifica cómo la red se aproxima al patrón a medida que el tiempo aumenta. El solapamiento se calcula como:

$$m^{\mu}(s) = \frac{1}{N^2 a^{\mu}(1-a^{\mu})} \sum_{i=1}^{N} \sum_{j=1}^{N} (\xi_{i,j}^{\mu} - a^{\mu})(s_{i,j} - a^{\mu}) \tag{1.3}$$

Una vez descritos todos los parámetros que intervienen en el algoritmo, se procede a presentar los pasos que sigue el mismo:

- (1) Lee la imagen que se quiere recordar ('prueba.txt') y el patrón deformado ('deformado.txt').
- (2) Calcula a^{μ}, $\omega_{ij,kl}$ y $\theta_{i,j}$.
- (3) Elige un punto al azar (n,m) de la red.
- (4) Evalua $p = min(1, exp - [- \triangle H/T])$.
- (5) Genera un número aleatorio $chi \in [0,1]$. Impone la condición de que si $chi < p$, entonces se cambia el valor de dicho punto de la red (si era +1 se cambia a 0 y viceversa).
- (6) Calcula el solapamiento $m^{\mu}(s)$ y se vuelca el dato en al archivo 'solapamiento.txt' en función del paso Monte-Carlo en el que se encuentre el algoritmo.
- (7) Vuelve al paso (3).

En el archivo llamado 'datos.txt' se encuentra el estado de la red en cada paso Monte-Carlo, se hace de esta forma para poder representar el gif posteriormente con el software *Gnuplot*. En el archivo 'estadoinicial.txt' se encuentra el patrón deformado y en 'estadofinal.txt' se encuentra el último estado de la red.

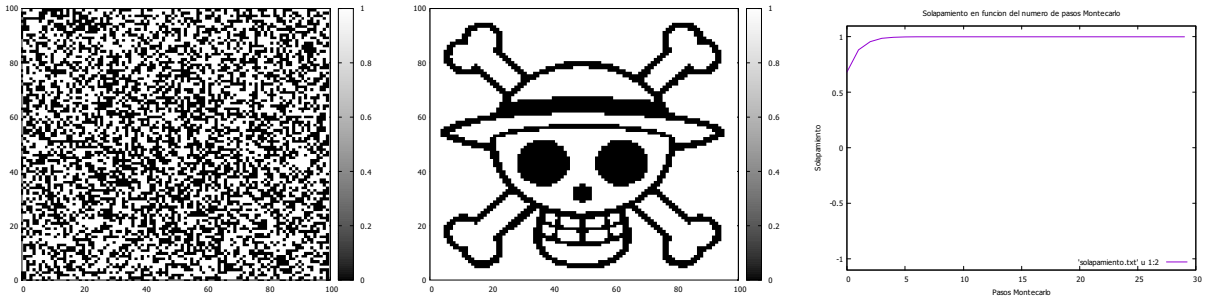
2 Resultados

En el repositorio de Gnuplot se podrán encontrar todas las gráficas mostradas en el informe, así como algunos gifs a modo de curiosidad y sus respectivos *.plt*.

Repositorio Gnuplot

2.1 Un patrón

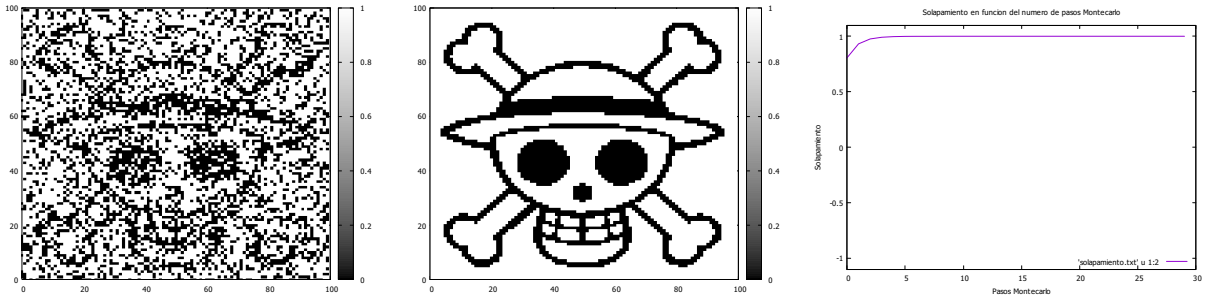
A continuación se va a estudiar como se comporta nuestra red para N=100, y 30 pasos Montecarlo. Se realiza un estudio para diferentes temperaturas, partiendo de un patrón completamente aleatorio y otro deformado. Para generar el patrón deformado, véase el programa *'desordenar.cpp'*, donde se indica el valor de N, y el parámetro p señala el número de elementos de la red que se van a cambiar. En cambio, para generar un patrón aleatorio, en el mismo programa *'hopfield1patron.cpp'* se genera la red aleatoriamente. Primero se va a partir de un estado inicial aleatorio, con $T = 10^{-4}$, obteniendo:



(a) Estado inicial aleatorio (b) Estado final (c) Solapamiento

Figura 2: Patrón inicial aleatorio, final y solapamiento para $T = 10^{-4}$

Seguidamente se procede a hacer lo mismo pero esta vez partiendo de un patrón deformado un 40 %:



(a) Estado inicial deformado un 40 % (b) Estado final (c) Solapamiento

Figura 3: Patrón inicial deformado un 40 %, final y solapamiento para $T = 10^{-4}$

Como se puede comprobar observando el solapamiento en ambos casos, en el deformado empieza en un valor más alto y alcanza antes la unidad, ya que el patrón inicial es similar al final, y por tanto tarda menos en alcanzar este último estado.

2.1.1 Solapamiento frente a la temperatura

En este subapartado se hará lo mismo que en el anterior, pero con objetivo de estudiar cómo cambia la curva de solapamiento frente a los pasos Montecarlo para diferentes

valores de temperatura. Partiendo de un estado inicial aleatorio, se obtiene:

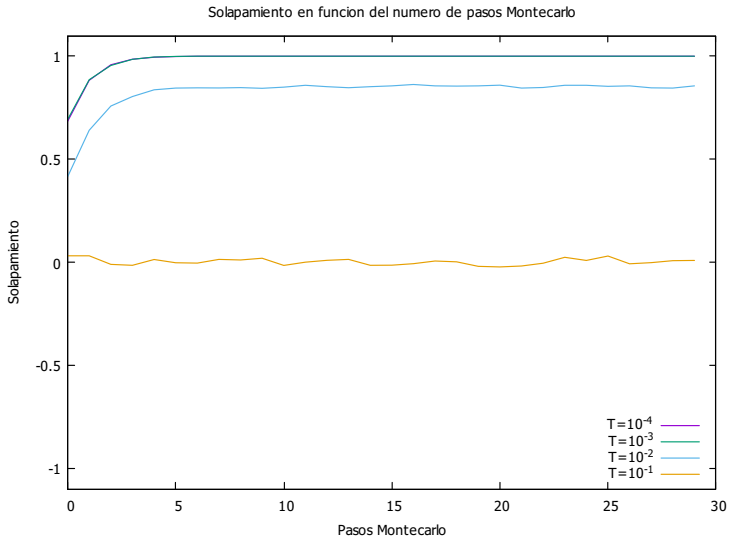
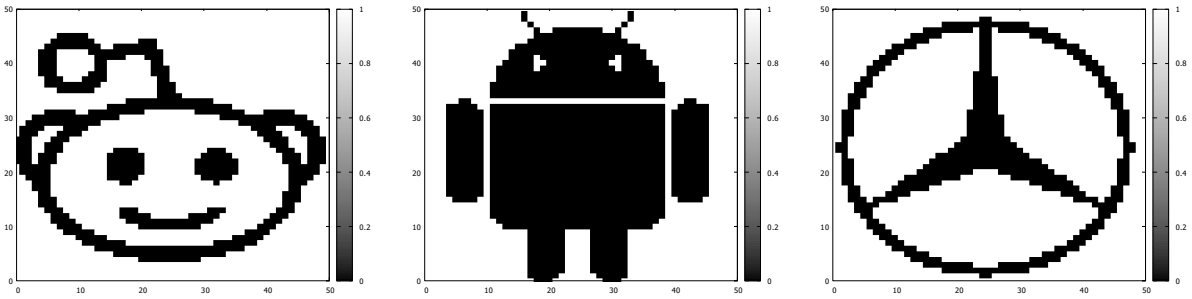


Figura 4: Curva de solapamiento para diferentes temperaturas en función de los pasos Monecarlo

Se observa que para una temperatura de $T = 10^{-4}$ y $T = 10^{-3}$ la curva de solapamiento converge rápidamente a la unidad. Sin embargo, al aumentar la temperatura, como en el caso de $T = 10^{-2}$, el solapamiento no converge tan rápido, y no llega a la unidad, esto se traduce en un estado final parecido al patrón prueba, pero no igual. Por último, como se aprecia en la curva para $T = 0.1$, no llega a formarse la imagen.

2.2 Tres patrones

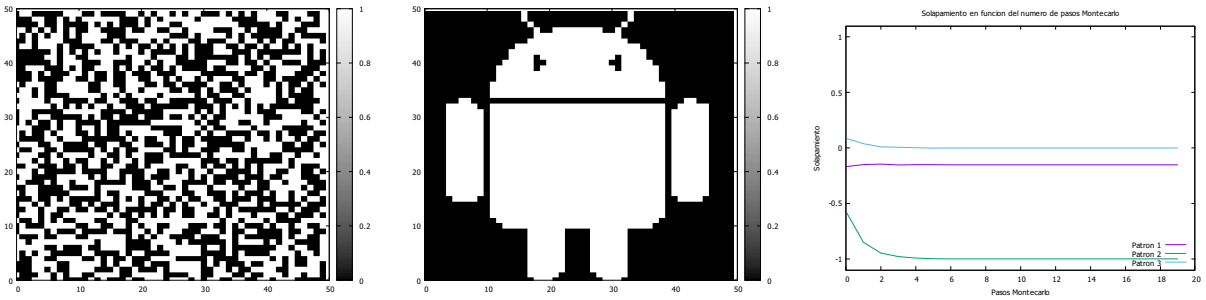
En este apartado se realizan los mismos estudios que en el anterior (con $T = 10^{-4}$), pero con la diferencia que ahora se cuenta con 3 patrones. Debido a esto, se redujo el valor de N a N=50, y se siguieron 20 pasos Monecarlo, ya que con los valores anteriores de estos parámetros, el programa requería mucha capacidad de cálculo y se demoraba en generar los resultados.
Los patrones escogidos son los siguientes:



(a) Patrón 1 (b) Patrón 2 (c) Patrón 3

Figura 5: Conjunto de patrones elegidos

Primero se va a partir de un estado inicial completamente aleatorio, y como en el apartado anterior, se mostrará el estado final, acompañado de la curva de solapamiento en función de los pasos Montecarlo. Cabe destacar, que como ahora estamos tratando con 3 patrones en lugar de 1, partiendo de una configuración aleatoria se podrá formar cualquiera de los tres:

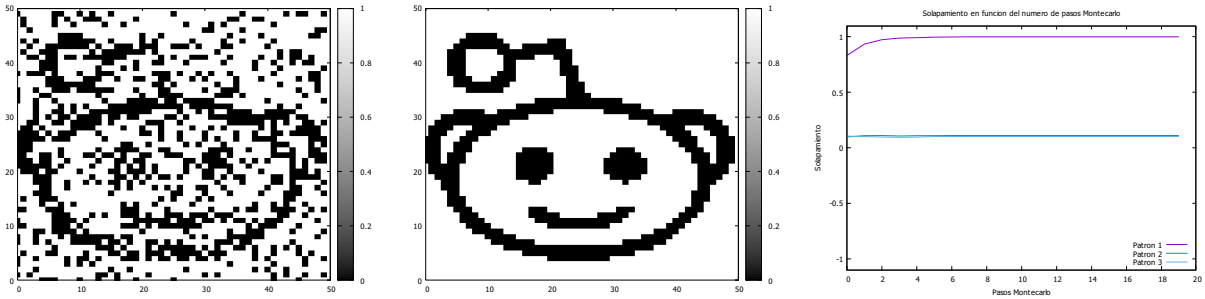


(a) Estado inicial aleatorio (b) Estado final (c) Patrón 3

Figura 6: Patrón inicial aleatorio, final y solapamiento para $T = 10^{-4}$

Partiendo de un estado aleatorio se obtuvo el antipatrón del patrón 2, esto se conoce como estado espúreo. Es decir, el programa converge a la misma configuración que el patrón 2, pero con los 1 y 0 de la red invertidos. Esto, a su vez, hace que el solapamieno converga a -1, en lugar de a la unidad positiva como hemos visto en los casos mostrados hasta ahora.

A continuación, se parte del patrón 1 deformado un 30 % :

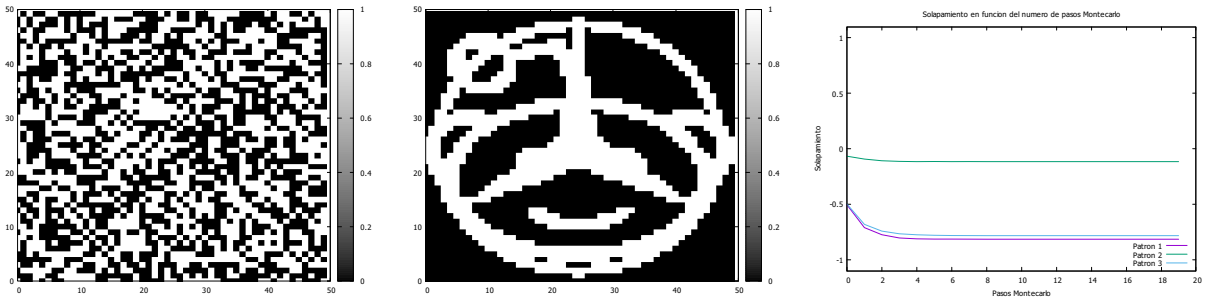


(a) Estado inicial deformado un 30 % (b) Estado final (c) Patrón 3

Figura 7: Patrón inicial deformado un 30 %, final y solapamiento para $T = 10^{-4}$

Como cabía esperar, el programa ha reconocido que el patrón inicial es muy similar al patrón 1. Esto puede observarse en el solapamiento, ya que la curva del patrón 1 empieza en un valor mucho más cercano a la unidad que el resto y evoluciona rápidamente a ella. En resumen, ocurre lo mismo que partiendo de un estado inicial deformado para el caso de un patrón.

Como curiosidad, una de las veces que se compiló el programa cuando se partía de un estado inicial aleatorio, el estado final resultó ser el antipatrón de la superposición del patrón 1 y 3. En la siguiente figura se muestra el estado inicial de la red, el final y las curvas de solapamiento:



(a) Estado inicial aleatorio (b) Estado final (c) Solapamiento

Figura 8: Patrón inicial aleatorio, final y solapamiento para $T = 10^{-4}$

El programa recordó el antipatrón 1 y 3 a la vez, por lo que en el estado final se muestra una superposición de ambos (¡y además invertidos!). Esto ocurre ya que ambos patrones son bastante similares en cuanto a forma. Es interesante observar la curva de solapamiento para el patrón 1 y 3, ya que se aprecia como ambos parten de un valor inicial muy similar (por no decir el mismo), y convergen al mismo tiempo a un solapamiento cercano a -1. Obviamente ninguno de los dos solapmientos converge a la unidad ya que el estado final es una superposición de dos patrones y no ninguno en concreto. De nuevo, en el repositorio de Github se incluye el *gif* de esta evolución.

2.3 Recuperación de la memoria en función del número de patrones

En este apartado se estudia cómo decae la memoria en función del número de patrones a $T = 10^{-4}$. Para ello, se ha creado un programa llamado *'hopfield-memoria.cpp'*, se propone un tamaño de red de 150x150, y se siguen 20 pasos Monecarlo. Para ello, se hacen k iteraciones, y en cada una se generan k patrones aleatorios, para ver cómo evoluciona el estado inicial de nuestra red. Al final de cada iteración, se calculará el solapamiento, y se considera que un patrón ha sido recordado cuando este es mayor a 0.75. Partiendo de estas bases, se comprueba que se recuerdan un total de 2 patrones como máximo, y de esta forma, $\alpha = P_c/N^2 = 0.9 \cdot 10^{-4}$.
Estos resultados no son demasiado fiables ya que el valor de N es pequeño, por lo que se necesitaría un N mucho mayor para que estadísticamente los resultados fueran fiables. No ha sido posible aumentar mucho más el valor de N debido a la potencia de cálculo requerida.

3 Conclusiones

A lo largo de este estudio se han podido comprobar diferentes aspectos del modelo de Hopfield. Primeramente para el caso de un solo patrón se demostró que partiendo de una deformación del 40 % del patrón prueba el valor inicial del solapamiento es mayor que si se parte de un estado inicial aleatorio, y este converge a la unidad mucho más rápido. Por debajo de ese porcentaje de deformación, el solapamiento converge aproximadamente igual de rápido a la unidad que partiendo de un estado inicial aleatorio.

Además, se estudió la curva de solapamiento en función de la temperatura, donde se vio que conforme aumentaba la misma, la red formaba un patrón más desordenado, llegando a no formar ninguna imagen para un valor de $T = 0.1$.

Otra parte importante de este estudio fue comprobar cómo evoluciona la red con 3 patrones en lugar de uno. Con un estado inicial aleatorio, la red evolucionó hasta formar el antipatrón del segundo patrón prueba, es decir, su complementario. Como se comentó, este estado se llama 'estado espúreo'. En este caso, el solapamiento converge a la unidad negativa, en lugar de a la positiva. En el caso en el que se partía de un estado inicial deformado, se eligió deformar un 30 % el patrón 1, y como se esperaba, la red evolucionó hasta formar el mismo patrón. Se vuelve a comprobar como en el caso de un solo patrón que el valor del solapamiento del patrón 1 en la primera iteración es bastante cercano a la unidad, lo que nos indica que el programa desde el primer momento recordó este patrón.

Igualmente fue interesante estudiar el estado final obtenido formado por la superposición de los antipatrones 1 y 3, donde se comprobó que el solapamiento de ambos patrones evolucionó hasta un valor próximo a la unidad negativa al mismo tiempo.

En cuanto a la recuperación de la memoria en función del número de patrones, se comprobó que el algoritmo era capaz de recordar como máximo dos patrones para $T = 10^{-4}$ y un valor de N=150. Este resultado era de esperar, ya que el programa en cuanto recuerda un patrón, crece el solapamiento del mismo, y al tratarse de patrones generados aleatoriamente, es difícil que el porgrama recuerde más de un patrón para uno prueba y haya dos patrones con un solapamiento similar. Igualmente, como se mencionó en dicho apartado, estadísticamente no son resultados muy fiables debido a que se debería experimentar con más patrones almacenados y para un mayor valor de N.

4 Anexo

4.1 Cálculo de expresión para $\triangle H$

$$H = -\frac{1}{2}\sum_{i}^{N}\sum_{j}^{N}\sum_{k}^{N}\sum_{l}^{N}\omega_{ij,kl}s_{i,j}s_{k,l} + \sum_{i}^{N}\sum_{j}^{N}\theta_{i,j}s_{i,j} = \frac{1}{2}\sum_{i}^{N}\sum_{j}^{N}s_{i,j}\sum_{k}^{N}\sum_{l}^{N}\omega_{ij,kl}s_{k,l} + \sum_{i}^{N}\sum_{j}^{N}\theta_{i,j}s_{i,j}$$

$$\triangle H = H(s') - H(s) = -\frac{1}{2}\left[\sum_{i}^{N}\sum_{j}^{N}\sum_{k}^{N}\sum_{l}^{N}\omega_{ij,kl}s'_{i,j}s'_{k,l} - \sum_{i}^{N}\sum_{j}^{N}\sum_{k}^{N}\sum_{l}^{N}\omega_{ij,kl}s_{i,j}s_{k,l}\right] + \sum_{i}^{N}\sum_{j}^{N}\theta_{i,j}s'_{i,j} - \sum_{i}^{N}\sum_{j}^{N}\theta_{i,j}s_{i,j} =$$

Haciendo (i,j)=(0,0):

$$= -\frac{1}{2}\left[\sum_{k}^{N}\sum_{l}^{N}\omega_{00,kl}s'_{00}s'_{k,l} - \sum_{k}^{N}\sum_{l}^{N}\omega_{00,kl}s_{00,j}s_{k,l}\right] + \theta_{00}s'_{00} - \theta_{00}s_{00} =$$

$$= -\frac{1}{2}\sum_{k}^{N}\sum_{l}^{N}\omega_{00,kl}s_{k,l}\left(s'_{00} - s_{00}\right) + \theta_{00}\left(s'_{00} - s_{00}\right) =$$

Llamando a $s'_{00} - s_{00} = 1 - 2s$ nos queda:

$$= -\frac{1}{2}\sum_{k}^{N}\sum_{l}^{N}\omega_{00,kl}s_{k,l}\left(1 - 2s\right) + \theta_{00}\left(1 - 2s\right) =$$

$$= (1 - 2s)\left[\theta_{00} - \frac{1}{2}\sum_{k}^{N}\sum_{l}^{N}\omega_{00,kl}s_{k,l}\right]$$

$$= (1 - 2s)\left[\theta_{00} - \frac{1}{2}\sum_{k\neq 0}^{N}\sum_{l\neq 0}^{N}\omega_{00,kl}s_{k,l}\right]$$

Por lo que, la expresión general queda:

$$\triangle H = \left(1 - 2s_{ij}\right)\left[\theta_{ij} - \frac{1}{2}\sum_{k\neq 0}^{N}\sum_{l\neq 0}^{N}\omega_{ij,kl}s_{k,l}\right]$$

Referencias

[1] G. Galán-Marín and J. Muñoz-Pérez (2001)
"Design and Analysis of Maximum Hopfield Networks" ,
IEEE Transactions of Neural Networks, 12, 2.

[2] *Hopfield Neural Network*, from Geeksforgeeks.org, website: https://www.geeksforgeeks.org/hopfield-neural-network/