3 Redes Recurrentes y Autónomas

3.1 Introducción

Las redes de neuronas artificiales son modelos matemáticos inspirados en las redes de neuronas biológicas y en el conocimiento que tenemos actualmente del cerebro. El cerebro humano posee una capacidad sorprendente para procesar la información y resolver muchos problemas donde los modernos ordenadores no pueden competir. La capacidad del cerebro para realizar una gran variedad de tareas, como el reconocimiento del habla, el reconocimiento y la identificación de los objetos de una escena en centésimas de segundo, el control manual de objetos y la capacidad de recordar (recuperar la información almacenada), son algunas de las características que lo hacen muy superior a los ordenadores actuales, a pesar de que las neuronas tienen respuestas lógicas mucho más lentas (del orden de milisegundos) que las puertas lógicas de los chips de silicio (del orden de nanosegundos).

En este capítulo vamos a estudiar un modelo de red neuronal artificial conocido como red de Hopfield, y su aplicación a la resolución de problemas de optimización, como son: el problema del viajante, el problema de la bipartición de un grafo, el problema del recubrimiento o el problema de la localización de centros de servicio. Por lo tanto, las redes de neuronas artificiales constituyen una clara alternativa a los algoritmos convencionales de la Investigación Operativa y a las técnicas heurísticas.

3.2 El modelo de Hopfield discreto

Este modelo fue introducido en 1982 por el físico norteamericano John Hopfield, en su trabajo titulado "*Neural Networks and Physical Systems with Emergent Collective Computational Abilities*" que tuvo gran impacto en la comunidad científica que trabajaba en Inteligencia Artificial.

El elemento básico de computación es una unidad de proceso bipolar. Una **unidad de proceso bipolar** es una función matemática con dominio el conjunto *N*-dimensional $\{-1,1\}^N$ y rango el conjunto {-1,1}, definida por la siguiente expresión:

$$f(x_1, x_2, ..., x_N) = \begin{cases} 1 & \text{si} \quad w_1 x_1 + w_2 x_2 + ... + w_N x_N \geq \theta \\ -1 & \text{si} \quad w_1 x_1 + w_2 x_2 + ... + w_N x_N < \theta \end{cases}$$

donde los parámetros $w_1, w_2, \ldots, w_N$, se llaman **pesos sinápticos** y son los pesos con los que se ponderan los valores de entrada $x_1, x_2, \ldots, x_N$, o argumentos de la función; la suma ponderada $u = w_1 x_1 + w_2 x_2 + ... + w_N x_N$ se llama potencial sináptico y el parámetro θ se

llama **umbral** o sesgo. También se puede expresar la función f mediante la función signo, es decir,

$$f(x_1, x_2, ..., x_N) = \text{sgn}(u - \theta)$$

siendo la función signo,

$$\text{sgn}(x) = \begin{cases} 1 & \text{x} \geq 0 \\ -1 & \text{x} < 0 \end{cases}$$

Análogamente, se define una unidad de proceso binaria como una función matemática con dominio el conjunto *n*-dimensional $\{0,1\}^n$ y rango el conjunto {0,1}, definida por la siguiente expresión:

$$f(x_1, x_2, ..., x_N) = \begin{cases} 1 & \text{si} \quad w_1 x_1 + w_2 x_2 + ... + w_N x_N \geq \theta \\ 0 & \text{si} \quad w_1 x_1 + w_2 x_2 + ... + w_N x_N < \theta \end{cases}$$

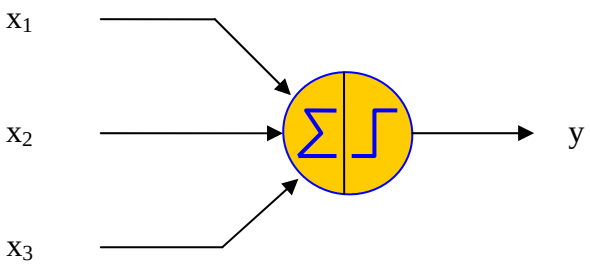


Figura 1. Unidad de proceso bipolar.

Cuando la salida de la unidad de proceso es igual a 1 se dice que dicha unidad de proceso está activada o encendida y presenta el estado 1, mientras que si su salida es igual a cero se dice que está desactivada o apagada, presentando el estado 0.

Una **red de Hopfield bipolar** está constituida por *N* unidades de proceso bipolares completamente conectadas entre sí, de manera que las entradas de cada unidad de proceso son las salidas de las demás unidades. La matriz de pesos sinápticos, es decir, constituida por las cantidades con las que se ponderan las salidas de las unidades de proceso, va a ser una matriz simétrica con los elementos de la diagonal principal iguales a cero. Por lo tanto,

$w_{ii} = 0, \quad i=1,2,\ldots,N$ (sin autoconexiones)
$w_{ij} = w_{ji}$, $i,j=1,2,\ldots,N.$ (simetría de los pesos)

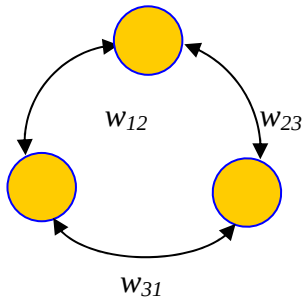


Figura 2. Red de Hopfield.

La red de Hopfield bipolar evoluciona de la siguiente manera: comienza a partir de un estado inicial de las unidades de proceso, que representaremos por $s_1(0)$, $s_2(0),\ldots,s_n(0)$, donde $s_i(0)\in\{-1,1\}$ representa el estado inicial de la unidad de proceso *i*, y en cada iteración se actualiza una unidad de proceso seleccionada aleatoriamente (diferente de la anterior) siguiendo la siguiente **regla de actualización recurrente** que nos determina la dinámica de computación de la red:

$$s_i(k+1)=\begin{cases}1 & \text{si} \quad \sum_{j=1}^{N} w_{ij}s_j(k) > \theta_i \\ s_i(k) & \text{si} \quad \sum_{j=1}^{N} w_{ij}s_j(k) = \theta_i \\ -1 & \text{si} \quad \sum_{j=1}^{N} w_{ij}s_j(k) < \theta_i \end{cases} \tag{1}$$

si en la iteración *k+1* hemos seleccionado la unidad de proceso *i.* Cuando después de actualizar todas las unidades de proceso ninguna de ellas cambia su estado anterior se dice que la red se ha **estabilizado** y el estado que presentan las unidades de proceso constituye una configuración final.

Puede observarse que la regla de actualización anterior también se puede expresar de la siguiente manera:

$$s_i(k+1)=s_i(k)+2\,\mathrm{sgn}(\sum_{j=1}^{N} w_{ij}s_j(k)-\theta_i) \tag{2}$$

Es decir,

$$\Delta s_i(k)=s_i(k+1)-s_i(k)=2\,\mathrm{sgn}(\sum_{j=1}^{N} w_{ij}s_j(k)-\theta_i) \tag{3}$$

Para actualizar las unidades de proceso podemos seleccionar una aleatoriamente y actualizarla, a continuación otra, y así sucesivamente, en cuyo caso diremos que estamos siguiendo una actualización **secuencial**, no sincronizada o asíncrona. La elección aleatoria es para asegurar imparcialidad en el proceso de actualización. En cambio, si actualizamos simultáneamente todas las unidades de proceso en cada iteración entonces diremos que la actualización se hace en **paralelo** o de forma sincronizada.

Ahora surgen cuatro preguntas:

- ¿Qué se persigue con esta dinámica de la computación?
- ¿Qué papel desempeñan los pesos sinápticos y el umbral?
- ¿Dejarán de cambiar de estado alguna vez las unidades de proceso cuando se actualizan? Es decir, se estabilizará la red en algún momento.
- ¿Cómo son las configuraciones de la red cuando se estabiliza?
- ¿Para qué se puede utilizar esta red neuronal?

Vamos a ir respondiendo a estas preguntas a lo largo de esta lección.

Con la regla de actualización anterior se pretende que la red encuentre una configuración en la que cada par (*i*, *j*) de unidades de proceso, cuyo peso sináptico w_{ij}

es positivo, presenten el mismo estado (concordancia), y esta propiedad se debe cumplir con más fuerza conforme mayor sea el valor positivo del peso sináptico de la conexión, mientras que en el caso contrario, si el peso sináptico w_{ij} es negativo se vean forzadas dichas unidades de proceso a presentar estados diferentes, con más fuerza conforma más se aleje del valor cero. Es decir, si el peso sináptico es positivo entonces debe de haber una correlación directa entre los estado, mientras que si es negativo debe de haber una correlación inversa entre dichos estados. De esta manera, cada peso sináptico w_{ij} representaría una *medida de la correlación* entre el estado de la unidad de proceso *i* y la unidad de proceso *j* cuando la red alcanza alguna configuración estable. Por lo tanto buscamos que

$$\text{si} \quad w_{ij} > 0 \quad \text{entonces} \quad s_i s_j = 1, \text{ y}$$
$$\text{si} \quad w_{ij} < 0 \quad \text{entonces} \quad s_i s_j = -1.$$

Esto es lo mismo que hacer máximo el producto $w_{ij}\, s_i s_j$, o lo que es lo mismo, que sea positivo. Por lo tanto, nuestro objetivo va ser maximizar la expresión:

$$\sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} s_i s_j$$

Sin embargo, como la correlación entre el estado de la unidad de proceso *i* y la unidad de proceso *j* es la misma que entre el estado de la unidad *j* y el estado de la unidad *i*, es por lo que se supone que $w_{ij} = w_{ji}$. Entonces en la expresión anterior cada conexión aparece dos veces por lo que nuestro objetivo va a ser maximizar

$$\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} s_i s_j$$

¿Qué papel desempeña aquí el umbral que ni aparece en la expresión anterior? Vamos a interpretar el umbral θ_i como el peso asociado a una *señal externa* que llega a la unidad de proceso *i* con valor igual a −1 y que pretende activar a la unidad de proceso *i* si el umbral es negativo y desactivarla en caso contrario.. Es decir, si $\theta_i < 0$ entonces trata de activarla (estado 1), con más fuerza conforme más grande sea dicho valor, y si $\theta_i > 0$ entonces trata de desactivarla. En definitiva, que nuestro objetivo va a ser maximizar la función

$$\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} s_i s_j + \sum_{i=1}^{N} \theta_i (-1) s_i$$

Por analogía con el estudio de los sistemas físicos que tienden a estados de mínima energía vamos a tomar como función objetivo el valor opuesto de la expresión anterior, que llamaremos función de energía computacional, y así pasamos de un problema de maximización a un problema de minimización. Es decir, se trata de minimizar la función

$$E(k) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} s_i(k) s_j(k) + \sum_{i=1}^{N} \theta_i s_i(k) \qquad (4)$$

donde E(k) representa el valor de la función de energía computacional en la actualización *k*-ésima.

La propiedad más importante de la red de Hopfield es que conforme evoluciona la red según su dinámica de computación, la energía va decreciendo hasta estabilizarse, es decir, hasta que alcanza un mínimo local de la misma, como se pone de manifiesto a continuación.

Teorema 1. Si en la iteración k+1 actualizamos el estado de la unidad de proceso r según la regla de actualización (1), manteniendo iguales los estados de las unidades de procesos restantes, entonces la función de energía decrece o permanece igual, es decir,

$$E(k+1) \le E(k)$$

Demostración:
En efecto,

$$E(k+1) - E(k) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} \cdot s_i(k+1) \cdot s_j(k+1) + \sum_{i=1}^{N} \theta_i \cdot s_i(k+1) + \\ + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} \cdot s_i(k) \cdot s_j(k) - \sum_{i=1}^{N} \theta_i \cdot s_i(k)$$

Si representamos el incremento de la unidad de proceso i por $\Delta s_i(k)$, es decir,

$$\Delta s_i(k) = s_i(k+1) - s_i(k)$$

entonces podemos sustituir en la función de energía los valores $s_i(k+1)$ en términos de $s_i(k)$ según la expresión

$$s_i(k+1) = s_i(k) + \Delta s_i(k)$$

resultando,

$$= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\left(s_i(k) + \Delta s_i(k)\right)\left(s_j(k) + \Delta s_j(k)\right) + \sum_{i=1}^{N} \theta_i\left(s_i(k) + \Delta s_i(k)\right) + \\ + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\left(s_i(k)s_j(k)\right) - \sum_{i=1}^{N} \theta_i s_i(k)$$

$$= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} \cdot s_i(k) \cdot \Delta s_j(k) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} \cdot \Delta s_i(k) \cdot s_j(k) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} \cdot \Delta s_i(k) \cdot \Delta s_j(k) + \\ + \sum_{i=1}^{N} \theta_i \cdot \Delta s_i(k)$$

Por la simetría de la matriz de pesos sinápticos ($w_{ij}=w_{ji}$):

$$= -\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} \cdot \Delta s_i(k) \cdot s_j(k) \quad -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} \cdot \Delta s_i(k) \cdot \Delta s_j(k) + \quad \sum_{i=1}^{N} \theta_i \cdot \Delta s_i(k) =$$

Sacando factor común $\Delta s_i(k)$ obtenemos

$$= -\sum_{i=1}^{N} \Delta s_i(k) \cdot \left[\sum_{j=1}^{N} w_{ij} \cdot s_j(k) - \theta_i + \frac{1}{2} \cdot \sum_{j=1}^{N} w_{ij} \cdot \Delta s_j(k)\right] \qquad (5)$$

Si sólo actualizamos la unidad de proceso r entonces $\Delta s_i(k)=0$ para todo $i \neq r$, quedando la expresión

$$= -\Delta s_r(k) \cdot \left[\sum_{j=1}^{N} w_{rj} \cdot s_j(k) - \theta_r + \frac{1}{2} w_{rr} \Delta s_r(k)\right]$$

Como w_{rr}=0, pues no hay autoconexiones, se tiene que

$$= -\Delta s_r(k) \cdot \left[\sum_{j=1}^{N} w_{rj} \cdot s_j(k) \quad - \theta_r \right]$$

≤ 0, pues si

$$\sum_{j=1}^{N} w_{rj} \cdot s_j(k) > \theta_r \quad \text{entonces} \quad s_r(k+1)=1 \text{ y } \Delta s_r(k) \geq 0, \quad \text{y si}$$

$$\sum_{j=1}^{N} w_{rj} \cdot s_j(k) < \theta_r \quad \text{entonces} \quad s_r(k+1)=-1 \text{ y } \Delta s_r(k) \leq 0.$$

O bien, se deduce directamente de la expresión (3). ■

Corolario 1. La red recurrente bipolar alcanza un estado estable en un número finito de pasos utilizando la regla de actualización secuencial y dicho estado corresponde a un mínimo local de la función de energía.

Demostración: En efecto, como la red reduce la función de energía en cada iteración (o al menos no cambia) y el número posible de estados que puede alcanzar la red es finito (2^N), se tiene que estabilizar en un número finito de pasos. Además una vez que está estabilizada no se puede reducir la función de energía cambiando el estado de una sola unidad de proceso, es decir, la solución encontrada es un mínimo local.

Un vector bipolar se dice que es un **atractor** si corresponde a un estado de equilibrio de la red (estado estable). En el modelo secuencial está claro que un atractor debe ser un mínimo local (o global) y viceversa.

¿Qué ocurre si la matriz de pesos sinápticos no es simétrica? En este caso no está garantizado que la red sea estable, es decir, puede estar continuamente cambiando de estado. Por ejemplo, si consideramos una red con dos unidades de proceso, donde $w_{12}=-1$, $w_{21}=1$, $\theta_1=0$ y $\theta_2=0$ entonces si comienza en el estado (1,1) y actualizamos la primera unidad de proceso resulta que $s_1(1) = -1$ pues el potencial sináptico es 1(-1), que es menor que cero. Si actualizamos a continuación la segunda unidad resulta que $s_2(2)= -1$, pues (-1)1 es menor que cero. Si actualizamos la primera unidad entonces $s_1(3) = 1$, ya que su potencial sináptico es igual a (-1)(-1), que es positivo. Si ahora actualizamos la segunda unidad entonces $s_2(4) = 1$, y, por lo tanto volvemos al estado inicial. Es decir, la red estaría cambiando siempre los estados de sus unidades de proceso de forma cíclica y no converge a un estado estable.

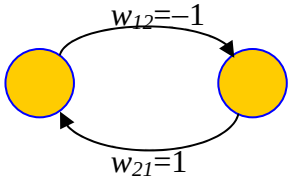


Figura 3. Red con conexiones asimétricas.

Si suponemos ahora que los pesos sinápticos son simétricos, $w_{12}= w_{21}=-1$, $\theta_1=0$ y $\theta_2=0$ (figura 4) entonces los atractores de la red (mínimos locales) son los vectores bipolares (1,−1) y (−1,1). La función de energía viene dada por la expresión $E(k) = s_1 \cdot s_2$. Sus mínimos globales son dichos estados. Así, si la red parte de la configuración (1,1) y actualizamos la primera unidad de proceso, como su potencial sináptico es −1 entonces se desactiva dicha unidad, es decir, pasa a la configuración (−1,1). Si actualizamos la

segunda unidad de proceso esta no cambia de estado, al igual que si actualizamos la primera, por lo que la red se estabiliza en dicha configuración.

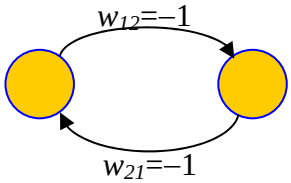


Figura 4. Red con conexiones simétricas.

Hasta ahora hemos actualizado las unidades de proceso de una en una, es decir, de manera **secuencial** (o **asíncrona**). A continuación vamos a estudiar la red recurrente cuando se actualizan al mismo tiempo todas las unidades de proceso, es decir, con **actualización paralela (o síncrona),** analizando bajo qué condiciones de la matriz de pesos sinápticos se garantiza el decrecimiento de la función de energía en cada iteración. El siguiente resultado nos da la respuesta.

Teorema 2. Si la matriz de pesos sinápticos es simétrica y semidefinida positiva, con todos los elementos de la matriz diagonal nulos, entonces la función de energía decrece o permanece igual, en cada actualización simultánea de las unidades de proceso, es decir,

$$E(k+1) \le E(k), \quad k = 1,2,\ldots$$

Demostración: De la expresión (2) tenemos que

$$\begin{aligned} E(k+1) - E(k) &= -\sum_{i=1}^{N} \Delta s_i(k) \cdot \left[\sum_{j=1}^{N} w_{ij} \cdot s_j(k) - \theta_i + \frac{1}{2} \cdot \sum_{j=1}^{N} w_{ij} \cdot \Delta s_j(k) \right] \\ &= -\sum_{i=1}^{N} \Delta s_i(k) \cdot \left[\sum_{j=1}^{N} w_{ij} \cdot s_j(k) - \theta_i \right] - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} \cdot \Delta s_i(k) \cdot \Delta s_j(k) \\ &\le 0 \end{aligned}$$

pues como hemos visto con anterioridad $\Delta s_r(k) \cdot \left[\sum_{j=1}^{N} w_{rj} \cdot s_j(k) - \theta_r \right] \ge 0$ y el término $\sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} \cdot \Delta s_i(k) \cdot \Delta s_j(k)$ es no negativo por ser la matriz de pesos sinápticos semidefinida positiva. ■

Corolario 2. La red recurrente bipolar alcanza un estado estable en un número finito de pasos utilizando la regla de actualización paralela.
Demostración: Análoga a la demostración del corolario 1.

Es importante resaltar que en el modelo paralelo los atractores no tienen por qué ser mínimos locales (globales), mientras que los mínimos locales sí que deben de ser atractores. Como consecuencia, hay muchos más atractores espurios en el modelo paralelo que en la versión secuencial. Sin embargo, el modelo secuencial puede quedar atrapado en mínimos locales más fácilmente que el modelo secuencial. Por ejemplo, si se encuentra en una configuración que difiere en el valor de una unidad de proceso para que sea un mínimo local, la red secuencial quedará atrapada en dicho mínimo local mientras que la paralela puede ser que no, al actualizar todas las unidades de proceso.

3.3 El modelo de Hopfield continuo

Estos modelos vienen caracterizados porque las salidas de sus neuronas son continuas (analógicas), es decir, son números reales entre –1 y 1, y el tiempo también es continuo, es decir, cada unidad de proceso examina constantemente su entrada y actualiza su salida de forma gradual.

En el caso discreto hemos visto que la salida de la unidad de proceso viene dada por la función escalón

$$x_i(k+1) = \operatorname{sgn}\left(\sum_{j=1}^{N} w_{ij} x_j(k)\right) \in \{-1,1\}, \quad k = 1,2,\ldots$$

Ahora vamos a utilizar unidades de proceso analógicas, es decir, que toman valores en el intervalo cerrado [-1,1], y que se actualizan en instantes infinitesimales de tiempo. En lugar del parámetro *k*, vamos a utilizar el parámetro $t \in [0,\infty)$, y las actualizaciones de las unidades de proceso se harán en instantes infinitesimales de tiempo. Así, por analogía con el caso discreto, en el que cuando $\Delta x_i(k) \neq 0$ se tiene que

$$\Delta x_i(k) = x_i(k+1) - x_i(k) = 2\operatorname{sgn}(\sum_{j=1}^{N} w_{ij} x_j(k) - \theta_i),$$

vamos a definir la regla de actualización para un incremento infinitesimal de tiempo, $\Delta t > 0$, de la siguiente manera:

$$\frac{dx_i(t)}{dt} = \lim_{\Delta t \downarrow 0} \frac{\Delta x_i(t)}{\Delta t} = \lim_{\Delta t \downarrow 0} \frac{x_i(t+\Delta t) - x_i(t)}{\Delta t} = \eta f(\sum_{j=1}^{N} w_{ij} x_j(t) - \theta_i)$$

siendo la función f una función continua y estrictamente creciente, con valores en el intervalo [-1,1], y que verifica que $f(0)=0$, como, por ejemplo, la función tangente hiperbólica (ver la figura 5):

$$f(u) = \frac{e^{\beta u} - e^{-\beta u}}{e^{\beta u} + e^{-\beta u}}$$

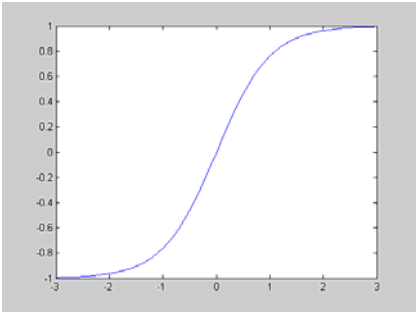


Figura 5. Representación gráfica de la función tangente hiperbólica.

Así, los cambios en las unidades de proceso serán graduales, es decir, que en un incremento infinitesimal del tiempo sólo pueden ocurrir cambios infinitesimales en las salidas de las unidades de proceso. El parámetro η es una cantidad constante positiva y pequeña, llamada tasa de aprendizaje, de cuyo valor depende la velocidad de convergencia.

Sin embargo, con la regla de actualización anterior no se garantiza que las unidades de proceso tomen valores dentro del intervalo [-1,1]. La acotación de los valores de las unidades de proceso es necesaria para facilitar la implementación, asegurar la

convergencia del proceso así como su plausibilidad biológica. Por ello, la regla de actualización que proponemos es la siguiente:

$$\frac{dx_i(t)}{dt} = \begin{cases} 0 & \text{si} \quad x_i(t) = 1 \text{ y } f\left(\sum_{j=1}^{N} w_{ij} x_j(t) - \theta_i\right) > 0 \\ 0 & \text{si} \quad x_i(t) = -1 \text{ y } f\left(\sum_{j=1}^{N} w_{ij} x_j(t) - \theta_i\right) < 0 \\ \eta f\left(\sum_{j=1}^{N} w_{ij} x_j(t) - \theta_i\right) & \text{si} \quad x_i(t) \in (-1,1) \end{cases} \tag{6}$$

Para estudiar la convergencia de la red utilizamos la función de energía computacional utilizada también en el caso discreto,

$$E(t) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} x_i(t) x_j(t) + \sum_{i=1}^{N} \theta_i x_i(t).$$

Vamos a ver que si la red sigue la regla de actualización anterior el sistema se estabiliza en una configuración de valores de las unidades de proceso que es un mínimo (local) de la función de energía computacional.

Teorema 3

En una red recurrente continua guiada por la regla (6) la función de energía computacional disminuye, o por lo menos no cambia, en cada actualización y alcanza un estado estable que es un mínimo local de la función de energía.

Demostración: En efecto, como

$$\frac{dE(t)}{dt} = \lim_{\Delta t \downarrow 0} \frac{E(t+\Delta t) - E(t)}{\Delta t}$$

y $\Delta t > 0$ entonces es suficiente probar que $dE(t)/dt \leq 0$, $t \in [0,\infty)$ (una función es decreciente si su derivada es negativa o nula)

Teniendo en cuenta que

$$\frac{dE(t)}{dt} = \sum_{i=1}^{N}\left(\frac{dE(t)}{dx_i(t)}\right)\left(\frac{dx_i(t)}{dt}\right) \tag{7}$$

vamos a probar que cada sumando

$$\left(\frac{dE(t)}{dx_i(t)}\right)\left(\frac{dx_i(t)}{dt}\right) \leq 0, i = 1,2,...,N. \tag{8}$$

En efecto, por una parte se tiene que

$$\frac{dE(t)}{dx_i(t)} = -\sum_{j=1}^{N} w_{ij} x_j(t) + \theta_i,$$

ya que $\frac{1}{2} w_{ij} x_i(t) x_j(t) + \frac{1}{2} w_{ji} x_j(t) x_i(t) = w_{ij} x_i(t) x_j(t)$, por ser la matriz de pesos simétrica.

Por otra parte, según la regla de actualización (6) resulta que:

a) Si $\frac{dx_i(t)}{dt} > 0$ entonces $f(\sum_{j=1}^{N} w_{ij} x_j(t) - \theta_i) > 0$

y como *f* es estrictamente creciente, con *f*(0)=0, esto es equivalente a que

$$\sum_{j=1}^{N} w_{ij} x_j(t) - \theta_i > 0 .$$

Por lo tanto, $\frac{dE(t)}{dt} < 0$ y así se verifica (8).

b) Si $\frac{dx_i(t)}{dt} < 0$ entonces $f(\sum_{j=1}^{N} w_{ij} x_j(t) - \theta_i) < 0$

y como *f* es estrictamente creciente, con *f*(0)=0, esto es equivalente a que

$$\sum_{j=1}^{N} w_{ij} x_j(t) - \theta_i < 0 .$$

Por lo tanto, $\frac{dE(t)}{dt} > 0$ y cumple (8).

c) Finalmente, si $\frac{dx_i(t)}{dt} = 0$ entonces el término correspondiente de la expresión (8) es cero.

La red alcanza un estado estable cuando $dx_i(t)/dt=0$, $\forall t \in [t_o,\infty)$, $i=1,2,\ldots,N$, es decir, ninguna de las unidades cambia su estado en una actualización. De la expresión (7) se deduce que entonces $dE(t)/dt=0$, $\forall t \in [t_o,\infty)$, es decir, dicho estado estable corresponde a un mínimo local de la función de energía. ■

Además, obsérvese que cuando la red alcanza un mínimo local de la función de energía, es decir, $dE(t)/dt=0$, es porque $dx_i(t)/dt=0$ (en cuyo caso $x_i(t)=1$ ó -1, o bien, $f(\sum_{i=1}^{N} w_{ij} x_j(t) - \theta_i) = 0$, es decir, $\sum_{i=1}^{N} w_{ij} x_j(t) - \theta_i = 0$) para algunas neuronas, y para la demás $dE(t)/dx_j=0$. En todas la neuronas se tiene que $dx_i(t)/dt = 0$, $i1,2,\ldots,N$. Es decir, si la red encuentra un mínimo local de la función de energía se estabiliza en él.

Asimismo, si no imponemos a las salidas de las unidades de proceso actualizaciones acotadas, es decir, dentro del intervalo [-1,1], la función de energía podría decrecer indefinidamente conforme evoluciona la red. Sin embargo, de esta forma la función de energía computacional tiene una cota inferior dada por la expresión:

$$-\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} |w_{ij}| - \sum_{i=1}^{N} |\theta_i| .$$

La red de Hopfield continua se puede considerar como una generalización de la discreta que incrementa considerablemente el espacio de estados y por tanto es más

probable que la función de energía computacional tenga muchos más mínimos locales que en el caso discreto. Sin embargo, los resultados experimentales muestran que el modelo continuo puede encontrar mejores soluciones que el discreto en problemas de optimización combinatoria. Aunque en problemas NP-difíciles, como el problema del viajante, el modelo continuo no presenta una clara ventaja frente al discreto. De todas formas, el modelo continuo se puede implementar más fácilmente con hardware analógico. Además, permite la computación paralela, es decir, que se actualicen simultáneamente las *N* unidades de proceso.

Por lo tanto, una red de Hopfield continua viene constituida por:

- Un conjunto de *N* unidades de proceso. Cada unidad de proceso tiene asociados *N-1* pesos sinápticos correspondientes a las conexiones con las demás unidades de proceso. Sean $w_{i1},w_{i2},...,w_{iN}$, los pesos sinápticos de la unidad de proceso *i*, siendo $w_{ii}=0$. Supondremos también que la matriz de pesos sinápticos es simétrica. Además, la entrada a dicha unidad, llamada *potencial de acción*, viene dada por la expresión:

$$u_i(t) = \sum_{j=1}^{N} w_{ij} x_j(t) - \theta_i$$

  donde $x_j(t)$ nos da la salida (estado) de la unidad de proceso *j* en el instante *t*, y θ_j es el umbral (sesgo) de dicha unidad.

- Una *función de transferencia* f que es una función continua estrictamente creciente cuyo rango es el intervalo [-1,1], y verifica $f(0)=0$. Por ejemplo, la función tangente hiperbólica:

$$f(u) = \frac{e^{\beta u} - e^{-\beta u}}{e^{\beta u} + e^{-\beta u}}$$

  $f(u_i(t))$ nos da la salida (estado) de la unidad de proceso *i* en el instante *t*.

- La *dinámica de la computación* que viene dada por la siguiente expresión:

$$\frac{dx_i(t)}{dt} = \begin{cases} 0 & \text{si} \quad x_i(t) = 1 \quad \text{y} \quad f(u_i(t)) > 0 \\ 0 & \text{si} \quad x_i(t) = -1 \ \text{y} \quad f(u_i(t)) < 0 \\ \eta f(u_i(t)) & \text{en otro caso} \end{cases} \tag{9}$$

  donde la tasa de aprendizaje η es una constante positiva pequeña cuya magnitud incide sobre la velocidad de convergencia. Esta regla nos indica cómo se actualiza la salida de la unidad de proceso en el instante *t*. Así, los cambios en la salida son graduales, es decir, en una cantidad de tiempo infinitesimal solamente pueden ocurrir cambios infinitesimales en las salidas de las unidades de proceso.

3.4 Aplicaciones a problemas de optimización combinatoria

En muchos problemas de optimización es difícil llegar a una solución óptima en un tiempo razonable de cómputo. Por ejemplo, consideremos el problema del viajante de comercio, que consiste en encontrar una ruta que pase por un cierto número de ciudades,

N, visitando cada día una de ellas de manera que cada ciudad sea visitada una sola vez, de forma que la distancia total recorrida sea mínima. Es decir, el problema consiste en encontrar el circuito Hamiltoniano de mínima longitud. En principio hay N! rutas posibles, pero como dada una ruta nos da igual el punto de partida, esto reduce el número de rutas a examinar en un factor N. Además, como no importa la dirección en que se desplace el viajante, el número de rutas a examinar se reduce también en un factor 2. Por lo tanto, hay que considerar (N-1)!/2 rutas posibles. Así, para un problema del viajante con 5 ciudades hay 12 rutas diferentes y no necesitamos un ordenador para encontrar la mejor ruta; para 10 ciudades hay 181.440 rutas diferentes y necesitaríamos ya un ordenador; para 30 ciudades hay más de 4×10^{31} rutas posibles, por lo que con un ordenador que calcule un millón de rutas por segundo necesitaría 10^{18} años. Si se hubiera comenzado a calcular al comienzo de la creación del universo, hace unos 13.400 millones de años (13.4×10^{9} años), todavía no se hubiera terminado. Puede comprobarse que si incorporamos una nueva ciudad el número de rutas se multiplica por el factor N. Es decir, el tiempo de cálculo que requiere el ordenador crece exponencialmente con el número de ciudades y, por ello, el problema pertenece a una clase de problemas que se conocen con el nombre de problemas NP-completos.

Ello nos lleva a que un método que encuentre una *buena* solución en un tiempo razonable será preferible al método que nos encuentre la *mejor* solución en un tiempo demasiado largo.

Por ello, la red de Hopfield es adecuada para este tipo de problemas, aunque nos suministre una solución que sólo sea mínimo local.

A continuación vamos a ver aplicaciones de la red de Hopfielp para encontrar soluciones factibles en problemas combinatorios y para encontrar soluciones mínimas locales en problemas de optimización combinatoria.

3.4.1 Redes biestables (o basculantes)

Vamos a comenzar con una red de Hopfield formada por dos neuronas y cuyos pesos son $w_{12} = w_{21} = -1$ y sus umbrales iguales a cero (ver figura 6). La función de energía es:

$$E(k)=s_1(k)s_2(k).$$

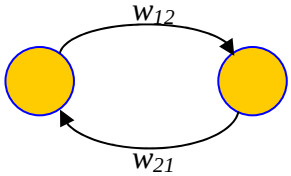


Figura 6. Red de Hopfield dipolar.

La red tiene cuatro configuraciones o estados posibles: (1,1),(1,-1),(-1,1) y (-1,-1), y posee dos estados estables, (1,-1) y (-1,1), que corresponden a mínimos globales de la función de energía (la energía total de las configuraciones estables es igual a -1, frente al valor de 1 correspondiente a los otros dos estados).

Por lo tanto, en esta red, cada neurona está forzando a la otra a que tome el valor opuesto para estabilizar la red; se comporta como una componente lógica con dos salidas que toman valores lógicos complementarios.

Por otro lado, esta red biestable se puede interpretar como una red capaz de almacenar uno de los estados (1,-1) ó (-1,1). Si tomamos como patrón a memorizar (1, -1) entonces los pesos de la red de Hopfield serán

$$w_{12} = 1(-1) = -1$$
$$w_{21} = (-1)1 = -1$$
$$w_{11} = w_{22} = 0.$$

Cada vez que la red comienza en un estado de partida la red le asigna el estado estable correspondiente. Este estado estable suele ser el patrón memorizado más próximo (con más componentes coincidentes) al estado de partida (patrón de entrada).

3.4.2. El problema de las ocho torres

Se trata de colocar ocho torres en un tablero de ajedrez de manera que sólo puede haber una torre en cada fila y en cada columna (ninguna torre debe estar en jaque) como se muestra en la figura 7. Para ello vamos a utilizar una red de Hopfield de 64 neuronas binarias {0,1} (en lugar de bipolares {-1,1}), una por cada casilla del tablero. La variable de estado s_{ij} representa el estado de la neurona que corresponde a la fila i y a la columna j del tablero de ajedrez,

$$s_{ij}(k) = \begin{cases} 1 & \text{si en la fila } i \text{ y columna } j \text{ hay una torre} \\ 0 & \text{si en la fila } i \text{ y columna } j \text{ no hay una torre} \end{cases}$$

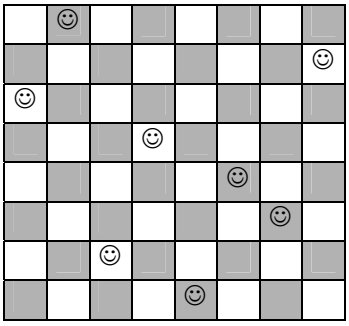


Figura 7. Ocho torres que no están en jaque.

Para modelar el problema vamos a establecer un conjunto de ecuaciones que recogen las exigencias impuestas en el problema:

a) Que sólo hay una torre en cada fila:

$$\left.\begin{array}{c} s_{11}+s_{12}+\ldots+s_{1N}=1 \\ s_{21}+s_{22}+\ldots+s_{2N}=1 \\ \vdots \\ s_{N1}+s_{N2}+\ldots+s_{NN}=1 \end{array}\right\} \quad \text{es decir,} \quad \sum_{j=1}^{N} s_{ij}=1\,;\, i=1,\ldots,N$$

b) Que sólo hay una torre en cada columna:

$$\left.\begin{array}{c} s_{11}+s_{21}+\ldots+s_{N1}=1 \\ s_{12}+s_{22}+\ldots+s_{N2}=1 \\ \vdots \\ s_{1N}+s_{2N}+\ldots+s_{NN}=1 \end{array}\right\} \quad \text{es decir,} \quad \sum_{i=1}^{N} s_{ij}=1\,;\; j=1,\ldots,N$$

Ahora tenemos que determinar una función de energía computacional que recoja dichas ecuaciones de manera que cuando se cumplan la energía alcance su valor mínimo. Así, la red neuronal partirá de una colocación arbitraria y siguiendo la dinámica de la computación llegará a una solución local de mínima energía.

Cuando tenemos un sistema de *m* ecuaciones de la forma

$$g_1(x_1,\ldots,x_n)=0$$

$$\ldots\ldots\ldots\ldots\ldots$$

$$g_m(x_1,\ldots,x_n)=0$$

que deseamos resolver, podemos resolverlo como un problema de minimización de la función

$$[g_1(x_1,\ldots,x_n)]^2+\ldots+[\,g_1(x_1,\ldots,x_n)]^2$$

Obsérvese que cada solución del sistema corresponde a un mínimo de dicha función y un mínimo de dicha función corresponde siempre a una solución del sistema de ecuaciones (suponiendo que exista) pues el valor mínimo de la función es igual a cero.

Por lo tanto, nuestra función de energía a minimizar va a ser:

$$E=\sum_{i=1}^{N}\left(\sum_{j=1}^{N} s_{ij}-1\right)^2+\sum_{j=1}^{N}\left(\sum_{i=1}^{N} s_{ij}-1\right)^2$$

Cualquier solución a la que corresponda el valor mínimo de la función de energía, E=0, será una solución factible para el problema de las ocho torres.

Si desarrollamos el primer término de la expresión anterior obtenemos:

$$\sum_{i=1}^{N}\left(\sum_{j=1}^{N} s_{ij}-1\right)^2=\sum_{i=1}^{N}\left[\left(\sum_{j=1}^{N} s_{ij}\right)\cdot\left(\sum_{k=1}^{N} s_{ik}\right)+1-2\cdot\sum_{j=1}^{N} s_{ij}\right]$$

$$=\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} s_{ij}\cdot s_{ik}+N-2\sum_{i=1}^{N}\sum_{j=1}^{N} s_{ij}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\substack{k=1\\k\neq j}}^{N} s_{ij}\cdot s_{ik} + \sum_{i=1}^{N}\sum_{j=1}^{N} s_{ij}^2 + N - 2\sum_{i=1}^{N}\sum_{j=1}^{N} s_{ij}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\substack{k=1\\k\neq j}}^{N} s_{ij}\cdot s_{ik} + N - \sum_{i=1}^{N}\sum_{j=1}^{N} s_{ij}.$$

pues $s_{ij}^2 = s_{ij}$ ya que . $s_{ij} \in \{0,1\}$

Desarrollando de manera similar el segundo término, resulta que la expresión de la energía queda:

$$E = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\substack{k=1\\k\neq j}}^{N} s_{ij}\cdot s_{ik} + N - \sum_{i=1}^{N}\sum_{j=1}^{N} s_{ij} + \sum_{j=1}^{N}\sum_{i=1}^{N}\sum_{\substack{r=1\\r\neq i}}^{N} s_{ij}\cdot s_{rj} + N - \sum_{j=1}^{N}\sum_{i=1}^{N} s_{ij}$$

Esta expresión se puede escribir también de la forma siguiente:

$$-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\substack{k=1\\k\neq j}}^{N}(-2)\cdot s_{ij}\cdot s_{ik} - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\substack{r=1\\r\neq i}}^{N}(-2)\cdot s_{ij}\cdot s_{rj} + \sum_{i=1}^{N}\sum_{j=1}^{N}(-2)\cdot s_{ij} + 2N$$

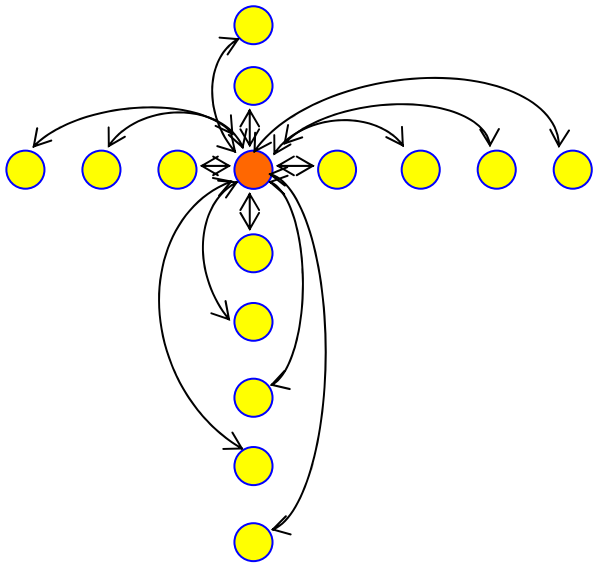


Figura 8. Conexiones de la neurona (3,4).

Dicha función va a ser la función de energía computacional que vamos a utilizar (salvo la constante *2N* que no influye a la hora de determinar el mínimo de dicha función). Es decir, identificando los coeficientes de los términos $s_{ij}s_{rk}$ como los pesos sinápticos $w_{ij,rs}$ obtenemos que

$w_{ij,ik}$ = -2 cuando $k \neq j$

$w_{ij,rj}$ = -2 cuando $r \neq i$

$w_{ij,ij}$ = 0

$$w_{ij,rk} = 0 \quad \text{en los demás casos.}$$

$$\theta_{ij} = -2.$$

Esto quiere decir que en la red cada unidad de proceso (*i*,*j*) solo está conectada con las unidades de proceso de su misma fila y de su misma columna (con peso igual a -2) y con las demás no está conectada (peso cero), como se puede ver en la figura 8.

3.4.3 El problema del recubrimiento minimal de los vértices de un grafo (servicios de vigilancia por vídeo)

Dado un grafo *G=(V,E),* siendo *V* el conjunto de vértices del grafo y *E* el conjunto de aristas, se trata de encontrar un subconjunto $X \subseteq V$ de forma que cada arista de *E* tenga al menos un vértice en dicho conjunto *X*, y además, no hay un subconjunto propio de X con dicha propiedad. Es decir, *X* es el subconjunto de vértices que recubre todas las aristas del grafo con menor número de elementos. Por ejemplo, en una ciudad donde las calles son las aristas del grafo y los puntos de concurrencia de calles son los vértices del grafo, podemos estar interesados en vigilar todas las calles con cámaras de vídeo desde los puntos de concurrencia. El número mínimo de cámaras de vídeo necesarias para vigilar todas las calles corresponde a la solución de este problema.

Consideremos una red neuronal de Hopfield con *N* (número de vértices del grafo) neuronas. La variable de estado de la neurona *i*, que representaremos por s_i, es binaria 0-1, con la siguiente interpretación:

$$s_i = \begin{cases} 1 & \text{si el vértice } i \text{ es uno de los seleccionados} \\ 0 & \text{en otro caso} \end{cases}$$

La función objetivo que deseamos minimizar viene dada por el número total de vértices seleccionados, es decir,

$$\sum_{i=1}^{N} s_i$$

Además, hay que asegurar que cada arista contenga al menos un vértice del conjunto del recubrimiento, *X*, o sea, que el número total de aristas cuyos dos vértices no están en *X* debe ser cero, es decir,

$$\sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{ij} (1 - s_i)(1 - s_j) = 0$$

donde $((a_{ij}))$ es la matriz de adyacencia del grafo,

$$a_{ij} = \begin{cases} 1 & \text{si existe la arista que une el vértice } i \text{ con el vértice } j \\ 0 & \text{en otro caso} \end{cases}$$

La restricción anterior se puede incorporar a la función objetivo de forma penalizada, de manera que el problema se reduce a *minimizar* la función:

$$\sum_{i=1}^{N} s_i + \lambda \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{ij} (1 - s_i)(1 - s_j)$$

El término de la derecha es siempre no negativo y por tanto alcanza su valor mínimo cuando es cero, es decir, cuando se satisface la restricción. Por ello, hay que tomar un valor de λ suficientemente grande que garantice que el mínimo de la función objetivo se alcance cuando su segundo término es nulo (solución factible).

Desarrollando la expresión anterior obtenemos:

$$\sum_{i=1}^{N} s_i + \lambda \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{ij} + \lambda \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{ij} s_i s_j - \lambda \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{ij} s_i - \lambda \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{ij} s_j$$

$$= \sum_{i=1}^{N} s_i + \lambda \sum_{i=1}^{N} n_i + \lambda \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{ij} s_i s_j - \lambda \sum_{i=1}^{N} n_i s_i - \lambda \sum_{j=1}^{N} n_j s_j$$

donde $n_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{ij}$ es el *grado de incidencia* del vértice *i*, es decir, el número de aristas que inciden en él y $a_{ii}=0$, $i=1,2,...,N$. Por lo tanto, la expresión se puede poner como:

$$= -\frac{1}{2} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} (-2\lambda) a_{ij} s_i s_j + \sum_{i=1}^{N} [1 - 2\lambda n_i] s_i + \text{constante}$$

Identificando el coeficiente del producto $s_i s_j$ como el peso sináptico w_{ij} en la función de energía computacional de Hopfield y el coeficiente del término en s_i como el umbral, se obtiene que:

$$\begin{aligned} w_{ij} &= -2\lambda a_{ij}, \ \forall i \neq j, \\ w_{ii} &= 0, \\ \theta_i &= 1 - 2\lambda n_i, \quad i = 1,2,...,N \end{aligned}$$

3.4.4 El problema de la bipartición de un grafo

Consideremos un grafo *G=(V,E)* cuyo número de vértices es *2N* (par). El problema consiste en descomponer el conjunto de vértices V en dos conjunto disjuntos V_1 y V_2, de *N* vértices cada uno, de manera que sea mínimo el número total de aristas que conectan un vértice de V_1 con un vértice de V_2.

Para formular este problema definimos la variables de decisión $s_1,...,s_{2N}$, de la siguiente manera:

$$s_i = \begin{cases} 1 & \text{si el vértice } i \text{ se asigna al conjunto } V_1 \\ -1 & \text{si el vértice } i \text{ se asigna al conjunto } V_2 \end{cases}$$

El número de conexiones (aristas) de los vértices de V_1 con los vértices de V_2 viene dado por la expresión:

$$\sum_{i=1}^{2N} \sum_{\substack{j=1 \\ j \neq i}}^{2N} a_{ij} \frac{(1 - s_i s_j)}{2}$$

donde $((a_{ij}))$ es la matriz de adyacencia del grafo,

$$a_{ij} = \begin{cases} 1 & \text{si existe la arista que une el vértice } i \text{ con el vértice } j \\ 0 & \text{en otro caso} \end{cases}$$

Obsérvese que la expresión $(1-s_i s_j)/2$ vale uno siempre y cuando uno de los vértices es del conjunto V_1 y el otro es del conjunto V_2.

Además, tenemos que asegurar que los dos conjuntos tengan el mismo número de vértices. Ello viene garantizado por la siguiente condición:

$$\sum_{i=1}^{2N} s_i = 0$$

Esta expresión nos dice que el número de variables con valor igual a 1 tiene que ser igual al número de variables con valor –1.

Por lo tanto, incorporamos la condición anterior en la función objetivo como un término de penalización y así el problema consiste en minimizar la función:

$$\sum_{i=1}^{2N} \sum_{\substack{j=1 \\ j \neq i}}^{2N} a_{ij} \frac{(1 - s_i s_j)}{2} + \lambda \left(\sum_{i=1}^{2N} s_i \right)^2$$

$$= \sum_{i=1}^{2N} \sum_{\substack{j=1 \\ j \neq i}}^{2N} a_{ij} / 2 - \sum_{i=1}^{2N} \sum_{\substack{j=1 \\ j \neq i}}^{2N} a_{ij} s_i s_j / 2 + \lambda \sum_{i=1}^{2N} \sum_{j=1}^{2N} s_i s_j$$

$$= \sum_{i=1}^{2N} \sum_{\substack{j=1 \\ j \neq i}}^{2N} a_{ij} / 2 - \sum_{i=1}^{2N} \sum_{\substack{j=1 \\ j \neq i}}^{2N} a_{ij} s_i s_j / 2 + \lambda \sum_{i=1}^{2N} \sum_{\substack{j=1 \\ j \neq i}}^{2N} s_i s_j + \lambda \sum_{i=1}^{2N} s_i^2$$

Como el primer término es constante (no depende de las variables) y el último término también lo es, pues $s_i^2 = 1$, entonces la función de energía computacional a minimizar viene dada por los dos términos restantes de la expresión anterior y se puede escribir de la forma siguiente:

$$E = -\frac{1}{2}\sum_{i=1}^{2N}\sum_{\substack{j=1 \\ j\neq i}}^{2N}(a_{ij} - 2\lambda)s_i s_j$$

Identificando el coeficiente del producto $s_i s_j$ como el peso sináptico w_{ij} en la función de energía computacional de Hopfield y el coeficiente del término en s_i como el umbral se obtiene que:

$$w_{ij} = (a_{ij} - 2\lambda),\ \forall\, i \neq j\ ,$$
$$w_{ii} = 0,$$
$$\theta_i = 0, \quad i = 1,2,...,2N$$

3.4.5 El problema del viajante de comercio

Un viajante de comercio tiene que hacer una ruta que pasa por N ciudades, visitando cada una de ellas una sola vez y volviendo al punto inicial. Se trata de determinar la secuencia, en la que hay que visitar dichas ciudades, que minimiza el coste total del recorrido. El número total de rutas posibles y diferentes es (n-1)!/2, pues para un viaje dado no importa cuál de las N ciudades sea el punto inicial, ya que la ruta es la misma al tener el mismo coste, y desde la ciudad de partida puede ir a una de las n-1 ciudades restantes; desde la segunda ciudad del trayecto se puede ir sólo a una de las n-2 ciudades restantes, y así sucesivamente, de manera que hay (n-1)! rutas posibles. Pero como no importa la dirección en la que se desplace el viajante, el número total de rutas diferentes se reduce a (n-1)!/2.

Para una ruta de cinco ciudades habrá 12 recorridos diferentes y no es preciso resolver el problema con el ordenador. Sin embargo, para 10 ciudades hay 181.440 rutas diferentes y hay que resolverlo por ordenador, pero para 30 ciudades hay más de 40×10^{30} rutas diferentes, es decir, que un ordenador que evaluara 10^{13} rutas por segundo y hubiese comenzado su cálculo cuando se creó el universo (hace más de 15.000 millones de año) no hubiera terminado aún de calcular todas la rutas posibles. Ello es porque el problema requiere un tiempo de cómputo que es una función exponencial del número de ciudades (y no polinomial). Por ello, una solución buena que se encuentre rápidamente será preferible a la mejor solución, que si llegamos a encontrarla puede ser ya demasiado tarde.

La red de Hopfield permite encontrar una buena solución para este problema, pues va mejorando la solución de partida y cuando se estabiliza alcanza un mínimo local de la función de energía (que va a ser la función objetivo que corresponde al coste total de la ruta).

Se trata ahora de desarrollar una representación de las soluciones posibles mediante unidades de proceso, y construir una función de energía cuyos mínimos locales favorezcan aquellos estados que contengan las distancias totales más cortas y aquellos estados que incluyan a cada ciudad una sola vez en el recorrido y que contengan una sola vez cada una de las posiciones del recorrido.

Para ello, vamos a utilizar una red de Hopfield con N^2 unidades de proceso, de manera que cada configuración de la red corresponda a una ruta. Así, se define la variable s_{ij} de la siguiente forma:

$$s_{ij} = \begin{cases} 1 & \text{si la ciudad i se encuentra en la posición j - ésima de la ruta} \\ 0 & \text{en otro caso.} \end{cases}$$

Como cada ciudad debe aparecer una sola en la ruta, se tiene que cumplir

$$\sum_{j=1}^{N} s_{ij} = 1, \quad i=1,2,...,N.$$

Además, sólo una ciudad se puede visitar en la posición j (tramo j-ésimo de la ruta). Es decir,

$$\sum_{i==1}^{N} s_{ij} = 1, \quad j=1,2,...,N.$$

Finalmente, la longitud total de la ruta viene dada por la expresión:

$$\frac{1}{2}\sum_{i,j,k}^{N} d_{ij} s_{ik} (s_{j(k+1)} + s_{j(k-1)})$$

pues el tramo entre las ciudades *i* y *j*, cuya distancia la representamos por d_{ij}, interviene en la ruta si se visita la ciudad *i* en la posición *k*, es decir, s_{ik} =1, y a continuación se visita la ciudad *j* (en la posición *k+1*, es decir, $s_{j(k+1)}=1$), o si se ha visitado la ciudad *j* en la posición *k-1* ($s_{j(k-1)}$=1) y de allí se visita la ciudad *i*. Se divide por dos porque una misma ruta aparece dos veces en la función objetivo, una en cada dirección.

Ahora sólo tendríamos que incorporar las restricciones a la función objetivo como término con penalización e identificar los pesos de la correspondiente función de Holpfield.

3.4.6 Diseño de un convertidor Analógico/Digital

Se trata de convertir (aproximar) una señal analógica continua $z(t) \in [0,3]$ en su representación binaria de 2 bits (x_0, x_1) que mejor la aproxima, donde x_0 corresponde al bit menos significativo, es decir, aproximar $z(t) \in [0,3]$ mediante $z^* = 2^0 x_0 + 2^1 x_1$, que según los valores de x_0 y $x_1 \in \{0,1\}$ dará lugar a los valores $z^* = 0$ (para x_0=0 y x_1=0), $z^* = 1$ (para x_0=1 y x_1=0), $z^* = 2$ (para x_0=0 y x_1=1) y $z^* = 3$ (para x_0=1 y x_1=1).

El objetivo es minimizar el error de representación que viene dado por la expresión:

$$E(t) = \left(z - (2^0 x_o + 2^1 x_1)\right)^2$$

Para ello utilizaremos una red de Hopfield continua con dos unidades de proceso y cuyos pesos sinápticos dependan del valor $z(t)$ dado, de manera que la red se estabilice en los estados x_o (la primera neurona) y x_1 (la segunda neurona) que conducen a la mejor representación digital de $z(t)$. Dicha red tendrá como función de energía la expresión $E(t)$ anterior.

Como los valores tienen que ser binarios (en lugar de bipolares) tomaremos como función de transferencia la función *logística*

$$f(x) = \frac{1}{1+e^{-ax}}$$

y además tenemos que imponer que

$$x_0(1-x_0) \quad \text{y} \quad x_1(1-x_1)$$

sean cero para asegurar que se estabilice en valores enteros cero o uno en lugar de valores del intervalo (0,1). Es decir, se trata de minimizar la función de energía computacional:

$$E(t) = \left(z - \sum_{i=o}^{1} 2^i x_i(t) \right)^2 + \lambda_0 x_0(t)(1-x_0(t)) + \lambda_1 x_1(t)(1-x_1(t))$$

$$= z^2 + 4x_0x_1 + (1-\lambda_0)x_0^2 + (4-\lambda_1)x_1^2 + (\lambda_0 - 2z)x_0 + (\lambda_1 - 4z)x_1$$

El término z^2 se puede quitar de la función de energía porque no depende de las variables a minimizar, su presencia sólo desplaza la función en un valor constante, y por tanto no afecta a los mínimos de la función de energía. Además, tomaremos λ_0=1 y λ_1=4, ya que los términos tercero y cuarto tienen que ser cero pues no aparecen en la función de energía de Hopfield. Por lo tanto, la función de energía será:

$$E(t) = -\frac{1}{2}\left[-4x_0x_1 - 4x_1x_0\right] + (1-2z)x_0 + 4(1-z)x_1$$

y por identificación de los coeficiente con la función de energía de Hopfield

$$E(t) = -\frac{1}{2}w_{01}x_0x_1 - \frac{1}{2}w_{10}x_1x_0 + \theta_0x_0 + \theta_1x_1$$

se obtiene

$$w_{01}=-4,\ w_{10}=-4,\ \theta_0=(1-2z),\ \theta_1=4(1-z).$$

Por lo tanto, las redes de Hopfield continuas también se pueden aplicar para resolver problemas de optimización combinatoria añadiendo a la función de energía el término de penalización

$$\sum_{i=1}^{n} \lambda_i x_i(t)\left[1-x_i(t)\right]$$

para asegurar que se estabilice en una solución binaria (no decimal) eligiendo valores de los coeficientes λ_i suficientemente grandes, ya que vale cero si y sólo si $x_i(t) \in \{0,1\}$, i=1,2,…,n. En el caso de unidades de proceso bipolares el término de penalización es

$$\sum_{i=1}^{n} (1-x_i(t)^2)$$

que vale cero si y sólo si $x_i(t) \in \{-1,1\}$, i=1,2,…,n.

En la prácticas, el modelo continuo se prefiere al discreto en muchos problemas, a pesar de ser este último más simple computacionalmente, ya que el modelo continuo puede evitar algunos de los numerosos mínimos locales pobres en los que suele quedar

atrapado el modelo discreto. Pero también el modelo continuo puede quedar atrapado en mínimos locales pobres como ocurre en el siguiente ejemplo:

Consideremos la red de Hopfield con dos unidades de proceso y con pesos sinápticos $w_{12}=w_{21}=-1$ (figura 9).

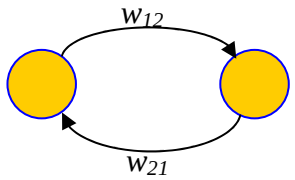


Figura 9. Red de Hopfield bipolar.

Si se parte del estado inicial $x_1(0)=1$ y $x_2(0)=1$, y tomamos umbrales iguales acero, entonces la red evoluciona actualizando sus estados y se estabiliza en el estado (0,0), es decir,

$$\lim_{t\to\infty} x_1(t)=0,\ \lim_{t\to\infty} x_2(t)=0\ .$$

Sin embargo, este estado no corresponde a un mínimo global de la función de energía que viene dada por la expresión $E(t)=x_1x_2$, pues los mínimos globales corresponden a los estados de la red (-1,1) y (1,-1). El modelo discreto sí se estabiliza en dichos valores.

3.5 Memorias Asociativas Dinámicas

En un ordenador convencional la información se almacena en dispositivos (discos duros, disquetes, CD-R, etc.) y para recuperarla es preciso conocer el lugar preciso donde se encuentra dicha información, es decir, para recuperar un conjunto de datos de la memoria hay que acceder a la dirección de memoria donde están esos datos. Sin embargo, la memoria humana no esta organizada de esta manera. Por ejemplo, si queremos recordar el nombre de una persona no nos sirve para nada saber que es la persona número 70 que conocemos. La información que se guarda en el cerebro no aparece como tal en ninguna lista. Sin embargo, sí nos puede ayudar a recordarlo saber que empieza por M. El ser humano recuerda sucesos cuando éstos están asociados a otros sucesos.

Se cree que la memoria humana (las redes neuronales biológicas) almacena la información en los puntos de contacto entre neuronas diferentes, la llamada sinapsis, y se sabe hace más de 100 años que las neuronas transmiten la información mediante señales eléctricas, y no utilizan para ello cables eléctricos (metálicos), sino membranas semipermeables e iones.

En este capítulo vamos a utilizar un dispositivo (una red neuronal recurrente), al que llamaremos **memoria asociativa**, para almacenar la información y poder recuperarla cuando sea necesario, es decir, una red retroalimentada, cuya salida se utiliza repetidamente como una nueva entrada hasta que el proceso converge. La información almacenada va a ser un conjunto de patrones, es decir, un conjunto de vectores binarios (o bipolares), que pueden representar firmas digitalizadas, caracteres, huellas dactilares, etc. Por lo tanto, el dispositivo deberá almacenar un conjunto dado de patrones, llamados **patrones de referencia**, memorias, patrones memorizados, patrones

principales o atractores. Partiendo de una entrada, llamada **patrón clave**, o simplemente clave, la red irá evolucionando, es decir, cambiando el estado de sus unidades de proceso, hasta alcanzar un estado estable que debe corresponder a un patrón de referencia almacenado. Por lo tanto, el dispositivo asociará a cada patrón clave su patrón de referencia. Para ello, el dispositivo utiliza una matriz de pesos sinápticos, que contiene la información que permite recuperar el patrón de referencia asociado al patrón de entrada. El proceso para la obtención de estos pesos sinápticos se llama **proceso de almacenamiento** o aprendizaje de los patrones de referencia (ver la figura 10). El número de patrones de referencia que tiene el dispositivo lo llamaremos tamaño de la memoria.

Una memoria asociativa recupera la información almacenada basándose en el conocimiento de parte de ésta (clave) y no en su posición en la memoria. El patrón clave puede ser una versión con ruido de un patrón memorizado, es decir, que difiere de él en pocas componentes. La memoria humana recuerda a una persona aunque vaya vestida de forma diferente o lleve gafas.

Figura 10. Memoria asociativa.

Hay dos tipos de tareas de asociación, la **autoasociación** que consiste en asociar el patrón de referencia a un patrón clave que es una versión con ruido del mismo (con la misma dimensión). Cuando el patrón de entrada es un patrón de referencia la red lo reconoce, es decir, le asocia a él mismo; y la **heteroasociación** que hace corresponder patrones de diferentes dimensiones, como, por ejemplo, asociar una firma digitar a un código (DNI).

Puede ocurrir que el patrón clave sea una versión de un patrón de referencia con demasiado ruido de manera que no se parezca a ninguno de los patrones memorizados, o que pueda haber varios patrones de referencia alternativos para un patrón de entrada, pues todos ellos se parecen lo mismo a dicho patrón.

Definición 3.1
Una **memoria autoasociativa** es un dispositivo para almacenar información que permite la recuperación de la misma basándose sólo en un conocimiento parcial de su contenido y no necesita conocer el lugar de su almacenamiento. La recuperación de la información se consigue según el grado de similitud entre el patrón de entrada y los patrones memorizados.

Consideremos los p patrones (vectores) $\mathbf{x}^k = (x^k_1, x^k_2, x^k_3,..., x^k_N)$, $k = 1,2, ... ,p$, con componentes x^k_i binarias (toman los valores cero o uno). Cada vector contiene N bits de información y al mismo tiempo se puede considerar como un *patrón binario* que representa las N características más destacadas de un determinado objeto. Supongamos

que deseamos almacenar en la memoria esta información suministrada por los *p* vectores de manera que cuando presentemos un nuevo patrón **(patrón de prueba)**

$$\mathbf{x}^* = (x^*_1, x^*_2, \ldots, x^*_N),$$

se recupere aquel patrón de memoria más parecido a este. El patrón de prueba puede ser una distorsión (ruido) de uno de los patrones de la memoria o alguna información parcial de los mismos. El parecido entre dos patrones, $\mathbf{x}^*$ y $\mathbf{x}^k$, se mide en términos de la desviación cuadrática media

$$D_k = \sum_{i=1}^{N} \left(x_i^* - x_i^k\right)^2$$

La desviación cuadrática media nos da el número de componentes diferentes (bits) que tienen diferentes los vectores $\mathbf{x}^k$ y $\mathbf{x}^*$, es decir, D_k es la distancia de Hamming entre los vectores $\mathbf{x}^k$ y $\mathbf{x}^*$. La distancia de Hamming también se puede expresar mediante la siguiente expresión:

$$D_k = \sum_{\mathrm{i}=1}^{\mathrm{N}} \left(x_i^k(1 - x_i^*) + (1\text{-}x_i^k)x_i^* \right)$$

Cuando los vectores son bipolares, es decir, sus componentes son –1 ó 1, entonces la distancia de Hamming viene dada por la expresión:

$$D_k = \sum_{\mathrm{i}=1}^{\mathrm{N}} \frac{1}{4}(x_i^k - x_i^*)^2$$

puesto que

$$(x_i^k - x_i^*)^2 = \begin{cases} 0 & \text{si } x_i^k = x_i^* \\ 4 & \text{si } x_i^k \neq x_i^* \end{cases}$$

Supongamos ahora que vamos a tener *p* pares entradas y salidas,

$$\{ (\mathbf{x}^1,\mathbf{y}^1),(\mathbf{x}^2,\mathbf{y}^2), \ldots ,(\mathbf{x}^\mathrm{p},\mathbf{y}^\mathrm{p}) \}, \quad \mathbf{x}^\mathrm{k} \in \Re^N, \ \mathbf{y}^\mathrm{k} \in \Re^M.$$

Es decir, al vector de entrada x^i le debe corresponder el vector de salida y^i. A dichos patrones los llamaremos memorias de referencia. Distinguiremos dos tipos de memorias asociativas:

-Memorias heteroasociativas
-Memorias autoasociativas

Una **memoria heteroasociativa e**s aquella que establece una correspondencia ϕ de $\Re^N$ en $\Re^M$ de tal manera que $\phi(\mathbf{x}^i) = \mathbf{y}^\mathrm{i}$, para i=1,2,..,p y además si **x** está más próximo a $\mathbf{x}^\mathrm{i}$ que a cualquier otro $\mathbf{x}^j$ entonces $\phi(\mathbf{x}) = \mathbf{y}^\mathrm{i}$.

Una **memoria autoasociativa** establece la misma correspondencia que la memoria heteroasociativa pero siendo los patrones de entrada y de salida los mismos, es decir. $\phi(\mathbf{x}^\mathbf{i}) = \mathbf{x}^\mathbf{i}$.

Por lo tanto, en una memoria autoasociativa vamos a tener *p* patrones de referencia a memorizar mientras que en una heteroasociativa vamos a tener *p* pares de patrones de referencia, puesto que hay que especificar las entradas (claves) y las salidas (memorias de referencia).

3.5.1 El Asociador Lineal

Supongamos que deseamos memorizar *p* pares de patrones (entradas y salidas),

$$\{ (\mathbf{x}^1,\mathbf{y}^1),(\mathbf{x}^2,\mathbf{y}^2), \ldots ,(\mathbf{x}^p,\mathbf{y}^p) \}, \quad \mathbf{x}^k \in \Re^N, \ \mathbf{y}^k \in \Re^M.$$

donde el vector $\mathbf{x}^k$ es un vector columna $N\times 1$ (entrada), el vector $\mathbf{y}^k$ es un vector columna $M\times 1$ (salida), y se supone que los *p* vectores de entrada (claves) son ortonormales, es decir, $(\mathbf{x}^k)^T\mathbf{x}^k = 1$ y $(\mathbf{x}^k)^T\mathbf{x}^i = 0$, $k \neq i$.

Un asociador lineal es una aplicación lineal de la forma

$$\mathbf{y}^i = \phi(\mathbf{x}^i) = W \cdot \mathbf{x}^i$$

donde $W = \{ w_{ij} \}$ es una matriz m×n, definida por la expresión:

$$W = \sum_{k=1}^{p} \mathbf{y}^k (\mathbf{x}^k)^T \tag{10}$$

Obsérvese que con esta elección de la matriz *W* se garantiza que

$$\phi(\mathbf{x}^i) = W \cdot \mathbf{x}^i = \sum_{K=1}^{p} \mathbf{y}^k (\mathbf{x}^k)^T \mathbf{x}^i = \mathbf{y}^i, \quad i = 1,2,..., p$$

Esta igualdad se cumple gracias a la ortonormalidad de los patrones.

Si tomamos como clave el vector $\mathbf{x} = \mathbf{x}^i + \mathbf{d}$, que es una pequeña modificación del vector clave $\mathbf{x}^i$ producida por el vector **d** que tiene la mayoría de sus componentes nulas**,** entonces el asociador lineal funcionaria de la siguiente manera:

$$\phi(\mathbf{x}) = \phi(\mathbf{x}^i + \mathbf{d}) = W \cdot (\mathbf{x}^i + \mathbf{d}) = \sum_{k=1}^{p} \mathbf{y}^k (\mathbf{x}^k)^T (\mathbf{x}^i + \mathbf{d})$$

$$= \mathbf{y}^i + \sum_{k=1}^{p} \mathbf{y}^k (\mathbf{x}^k)^T \mathbf{d} = \mathbf{y}^i + \phi(\mathbf{d})$$

Es decir, nos devuelve una versión también modificada del vector asociado $\mathbf{y}^i$ según una función de **d**. El asociador lineal actúa como una memoria interpolativa, pues a vectores próximos a $\mathbf{x}^i$ le asocia vectores próximos a $\mathbf{y}^i$ ya que ϕ es una función continua.

Sin embargo, la condición de que las claves tengan que ser ortogonales es muy fuerte ¿qué ocurre si las claves no fuesen ortogonales, sino solo normalizadas, $(\mathbf{x}^i)^T\mathbf{x}^i = 1, \forall i \in \{1,2, \ldots , p\}$? Tendríamos que

$$\phi(x^i) = W \cdot x^i = \sum_{K=1}^{p} y^k (x^k)^T x^i$$

$$= y^i (x^i)^T x^i + \sum_{\substack{k=1 \\ k\neq i}}^{p} y^k (x^k)^T x^i$$

$$= y^i + \sum_{\substack{k=1 \\ k\neq i}}^{p} y^k (x^k)^T x^i$$

No tenemos garantizada la correcta asociación a menos que el segundo miembro de la suma sea 0, cosa que es difícil mientras los patrones no sean ortogonales.

Como conclusión, podemos decir que el asociador lineal es la memoria asociativa más simple pero no es capaz de memorizar bien claves no ortogonales. Por ello, vamos a estudiar un nuevo modelo de memoria asociativa que incorpora la no linealidad pero de forma sencilla.

3.5.2 Memorias asociativas dinámicas no lineales (red de Hopfield)

A continuación vamos a estudiar un asociador no lineal simple (una red recurrente de Hopfield) que puede almacenar patrones no ortogonales. Supongamos que deseamos memorizar *p* patrones

$$\{\mathbf{s}^1, \mathbf{s}^2, \ldots, \mathbf{s}^p\}, \quad \mathbf{s}^k \in \Re^N, \ k=1,2,\ldots,p$$

Se trata de determinar los pesos sinápticos de una red de Hopfield con *n* unidades de proceso (tantas como componentes tienen los patrones) utilizando la información de los patrones a memorizar de manera que cuando la red comienza en una configuración, que corresponda a un patrón de entrada determinado, evolucione según la dinámica de la computación (1) hasta que se estabilice en una configuración que se corresponda con uno de los patrones memorizados (el más parecido con el patrón de entrada de la configuración inicial). Si tomamos el patrón de entrada de la red $\mathbf{s}=(s_1, s_2, \ldots, s_N)^T$, es decir, $S_i(0) = s_i, \ \forall\ i\in\{1,2,\ldots,N\}$ y $S_i(k) = S_i(k+1) = s_i \ \forall k\geq 1$, $i\in\{1,2,\ldots,N\}$, entonces diremos que la red ha memorizado dicho patrón. Por lo tanto, el patrón **s** está memorizado si la red se estabiliza en él mismo.

Supongamos que deseamos memorizar un solo patrón, el patrón

$$\mathbf{s}=(s_1, s_2, \ldots, s_N)^T$$

Vamos a determinar los pesos sinápticos de la siguiente manera:

$$w_{ij} = \frac{1}{N} s_i s_j \qquad (11)$$

Se suele conocer con el nombre de regla de Hebb, por su similitud con la hipótesis hecha por Hebb en 1949 acerca de la manera en que las fuerzas sinápticas en el cerebro cambian como respuesta a la experiencia (estímulos externos). Obsérvese que cuando las componentes s_i y s_j son iguales el peso es positivo y cuando son diferentes es negativo (w_{ij} una medida de la correlación entre los estados de las dos neuronas conectadas).

Con la elección de estos pesos, se memoriza el patrón de entrada pues cada vez que la red reciba este patrón de entrada (configuración inicial) va a dar como salida el mismo patrón, puesto que

$$S_i(1) = \text{sgn}\left[\sum_{j=1}^{N} w_{ij} S_j(0)\right] = \text{sgn}\left[\sum_{j=1}^{N} w_{ij} s_j\right] = \text{sgn}\left[\sum_{j=1}^{N} \frac{1}{N}(s_i s_j) s_j\right] = \text{sgn}[s_i] = s_i \qquad (12)$$

Vamos a ver cómo se comporta la red cuando se le presenta un patrón de prueba que difiere de los patrones memorizados. Supongamos que la red comienza en el estado determinado por los valores del patrón de prueba ($r_1,r_2,...,r_N$), que tiene las n primeras componentes diferentes con el patrón memorizado ($s_1,s_2,...,s_N$), y el resto iguales, es decir,

$$r_i = \begin{cases} -s_i & i = 1,2,...,n \\ s_i & i = n+1, n+2,..., N \end{cases}$$

Si actualizamos la unidad de proceso i, en la primera iteración,

$$\begin{aligned} S_i(1) &= \text{sgn}\left[\sum_{j=1}^{N} w_{ij} S_j(0)\right] = \text{sgn}\left[\sum_{j=1}^{n} w_{ij}(-s_j) + \sum_{j=n+1}^{N} w_{ij} s_j\right] \\ &= \text{sgn}\left[\sum_{j=1}^{n} (\frac{1}{N} s_i s_j)(-s_j) + \sum_{j=n+1}^{N} (\frac{1}{N} s_i s_j) s_j\right] \\ &= \text{sgn}\left[(1 - \frac{2n}{N}) s_i\right] \end{aligned}$$

$$= \begin{cases} -s_i & \text{si} \quad n > N/2 \\ s_i & \text{si} \quad n \le N/2 \end{cases}$$

Por lo tanto, si el patrón de entrada tiene mayoría de componentes iguales al memorizado se estabilizará la red en el patrón memorizado mientras que si tiene mayoría de componentes diferentes se estabiliza en el patrón opuesto del memorizado. Es decir, la red de Hopfield cada vez que memoriza un patrón también memoriza su opuesto, puesto que si $n=N$ entonces ($r_1,r_2,...,r_N$)= ($-s_1,-s_2,...,-s_N$) es el patrón opuesto y se estabiliza en él mismo.

Supongamos ahora que deseamos memorizar p patrones fundamentales

$$(s_1^k, s_2^k, ..., s_N^k) \quad \text{para} \quad k \in \{1,2,...,p\}$$

En este caso, la regla de Hebb viene dada por la siguiente expresión:

$$w_{ij} = \frac{1}{N} \sum_{k=1}^{p} s_i^k s_j^k \qquad (13)$$

Se puede considerar como una regla de aprendizaje puesto que el valor del peso sináptico para p+1 patrones se puede considerar como una modificación del valor que tenía para los p primeros patrones añadiéndoles $s_i^{p+1} s_j^{p+1} / N$, es decir,

$$w_{ij}(p+1) = w_{ik}(p) + \frac{1}{N} s_i^{p+1} s_j^{p+1}$$

ya que según la regla de Hebb (1949) las conexiones sinápticas del cerebro modifican su respuesta con cambios que son proporcionales a la correlación entre las activaciones de las neuronas pre y postsinápticas.

Obsérvese que $w_{ii} = p/N$, i=1,2,…,p, pues $s_i^k s_i^k = 1$. Sin embargo, se suele tomar $w_{ii} = 0$, i=1,2,…,p, puesto que conduce a resultados similares, ya que en la expresión (12) tendríamos:

$$S_i(1) = \operatorname{sgn}\left[\sum_{\substack{j=1 \\ j\neq i}}^{N} w_{ij} S_j(0)\right] = \operatorname{sgn}\left[\sum_{\substack{j=1 \\ j\neq i}}^{N} w_{ij} s_j\right] = \operatorname{sgn}\left[\sum_{\substack{j=1 \\ j\neq i}}^{N} \frac{1}{N}\left(s_i s_j\right) s_j\right] = \operatorname{sgn}\left[\frac{N-1}{N} s_i\right] = s_i$$

Además, en algunos casos, se pueden evitar estados espurios, es decir, estados en los que se estabiliza la red y no corresponden a patrones fundamentales (o a sus opuestos), pues, si w_{ii} es mayor que $\sum_{\substack{j=1 \\ j\neq i}}^{N} w_{ij} S_j(k)$ (en alguna etapa k), como

$$S_i(k+1) = \operatorname{sgn}\left(\sum_{\substack{j=1 \\ j\neq i}}^{N} w_{ij} S_j(k) + w_{ii} S_i(k)\right),$$

entonces la unidad de proceso i se estabilizaría en el estado $S_i(k)$, es decir, $S_i(k+1) = S_i(k)$, favoreciendo los estados espurios en la vecindad de un patrón fundamental.

El potencial sináptico asociado a la neurona i en la iteración inicial k=1 cuando la red se encuentra en la configuración que corresponde al patrón, $(S_1(1), S_2(1), ..., S_N(1)) = (s_1^r, s_2^r, ..., s_N^r)$ viene dado por la siguiente expresión, después de sustituir los pesos w_{ij} por los valores de la expresión (13):

$$h_i(1) = \frac{1}{N}\sum_{j=1}^{N}\sum_{k=1}^{p} s_i^k \, s_j^k \, S_j(1) = \frac{1}{N}\sum_{\substack{k=1 \\ k\neq r}}^{p}\sum_{j=1}^{N} s_i^k \, s_j^k \, s_j^r + \frac{1}{N}\sum_{j=1}^{N} s_i^r \, s_j^r \, s_j^r$$

$$= \frac{1}{N}\sum_{\substack{k=1 \\ k\neq r}}^{p} s_i^k \sum_{j=1}^{N} s_j^k \, s_j^r + s_i^r$$

Para que la salida, $S_i(2) = sgn[h_i(1)]$, coincida con el patrón de entrada, s_i^r, i=1,2,...,N, basta que los p patrones de entrada sean ortogonales dos a dos, es decir,

$$\sum_{j=1}^{N} s_j^k \, s_j^r = 0, \quad \forall k \neq r$$

También ocurrirá eso cuando N es mucho más grande que p, de manera que la cantidad

$$\frac{1}{N}\sum_{\substack{k=1 \\ k\neq r}}^{p} s_i^k \sum_{j=1}^{N} s_j^k \, s_j^r$$

sea lo suficientemente pequeña en comparación con s_i^r, de forma que $sgn[h_i(0)]$ coincidirá con s_i^r.

Sin embargo, en la práctica los patrones a memorizar seguramente no serán ortogonales, por lo que para garantizar la memorización correcta *N* tendrá que ser lo suficientemente grande en comparación con p. En general, la capacidad de almacenamiento de la red se define como el número máximo de patrones que puede memorizar la red con un error "aceptable".

Veamos cómo se comporta la red si utilizamos como patrón de prueba $(-s_1^r, s_2^r, ..., s_N^r)$ que es una versión con ruido del patrón $(s_1^r, s_2^r, ..., s_N^r)$.

El potencial sináptico es:

$$h_i(0) = \frac{1}{N}\sum_{\substack{k=1 \\ k\neq r}}^{p} s_i^k \left(s_1^k(-s_1^r) + \sum_{j=2}^{N} s_j^k s_j^r \right) + \frac{1}{N} s_i^r \left[s_1^r(-s_1^r) + \sum_{j=2}^{N} s_j^r \, s_j^r \right]$$

$$= \frac{1}{N}\sum_{\substack{k=1 \\ k\neq r}}^{p} s_i^k \left(-s_1^k s_1^r + \sum_{j=2}^{N} s_j^k s_j^r \right) + \left(1 - \frac{2}{N}\right) s_i^r$$

Así, cuanto mayor sea *N* con respecto a *p*, (de manera que se el signo de la expresión anterior sea el signo del segundo término, es decir, de s_i^r (suponiendo *N*>2)), más fiable será la red, es decir, cometerá menos asignaciones incorrectas. En este caso la red le asignaría dicha versión con ruido al patrón memorizado $(s_1^r, s_2^r, ..., s_N^r)$.

De manera similar, si el patrón utilizado difiere del patrón almacenado $(s_1^r, s_2^r, ..., s_N^r)$ en *n* componentes, se llega a la expresión:

$$h_i(t) = \frac{1}{N}\sum_{\substack{k=1 \\ k\neq r}}^{p} s_i^k \left(-\sum_{j=1}^{n} s_j^k s_j^r + \sum_{j=n+1}^{N} s_j^k s_j^r \right) + \left[1 - \frac{2n}{N}\right] s_i^r$$

Así, conforme menores sean *n* y *p* con respecto a *N*, más fiables será dicha red como memoria asociativa.

3.5.3 Capacidad de almacenaje de una memoria asociativa no lineal

¿Cuántos patrones de referencia (atractores) se pueden almacenar en una red de Hopfield con *N* unidades de proceso de manera que se puedan recuperar sin error? El comportamiento de las redes de Hopfield depende en gran medida del número de patrones de referencia. La *capacidad de almacenamiento* de una red neuronal se refiere a la cantidad de información que se puede almacenar en la red de tal manera que se recupere sin error. Una medida de la capacidad de almacenamiento es la siguiente:

$$C = \frac{p}{N}$$

donde p es el número de patrones almacenados y N es el número de unidades de proceso de la red neuronal.

Si la red no está completamente conectada entonces otra medida de su capacidad de almacenamiento puede ser

$$C_w = \frac{p}{N_w}$$

donde N_w es el número de conexiones de la red.

La capacidad de almacenamiento realmente depende de varios factores, como los pesos sinápticos, la similitud entre los patrones almacenados, y la diferencia entre los patrones estímulo y los de referencia. Amit (1989) estableció una cota sobre el número de patrones de manera que se garantice que cada patrón de referencia se recupere sin error.

Teorema

La capacidad máxima de una red de Hopfield está acotada por $c = \frac{1}{4\ln N}$. Es decir,

$\lim_{N\to\infty} P$(todas las componente de todos los patrones almacenados sean recuperada correctamente) = 1,

siempre que $p < \frac{N}{4\ln N}$.

<u>Ejemplo</u>:

Supongamos que deseamos diseñar un **asociador no lineal** (red de Hopfield) que memorice los patrones (1 -1 1) y (-1 1 -1) que representan a las imágenes siguientes:

Los ocho patrones posibles se corresponden con los vértices del cubo de la figura 11.

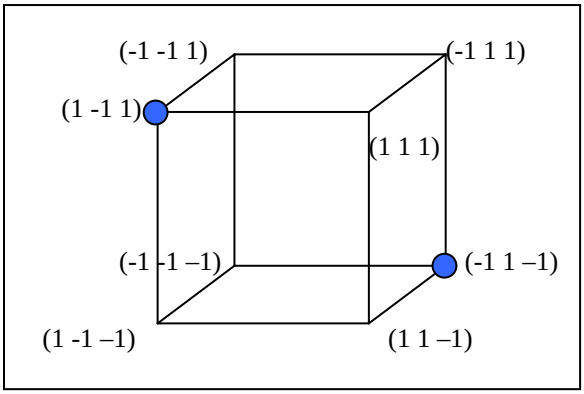


Figura 11. Patrones memorizados.

Vamos a diseñar una red de Hopfield que tiene tres unidades de proceso y cuyos pesos sinápticos vienen dados por la expresión (1),

$$w_{12} = \frac{1}{3}(-1-1) = -\frac{2}{3},\ w_{13} = \frac{1}{3}(1+1) = \frac{2}{3},\ \ w_{23} = \frac{1}{3}(-1-1) = -\frac{2}{3}$$

Es decir,

$$\mathbf{w} = \begin{pmatrix} 0 & -2/3 & 2/3 \\ -2/3 & 0 & -2/3 \\ 2/3 & -2/3 & 0 \end{pmatrix}$$

Obsérvese que los valores w_{ij} $(i \neq j)$ de dicha matriz de pesos también se puede obtener mediante los productos matriciales:

$$w = \frac{1}{N} \cdot \sum_{k=1}^{p} s^k \cdot \left(s^k\right)^T = \frac{1}{3} \cdot \left[\begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & -1 & 1 \end{pmatrix} + \begin{pmatrix} -1 \\ 1 \\ -1 \end{pmatrix} \cdot \begin{pmatrix} -1 & 1 & -1 \end{pmatrix} \right].$$

$$= \frac{1}{3} \begin{pmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{pmatrix} + \frac{1}{3} \begin{pmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 2/3 & -2/3 & 2/3 \\ -2/3 & 2/3 & -2/3 \\ 2/3 & -2/3 & 2/3 \end{pmatrix}.$$

La red resultante se muestra en la figura 12.

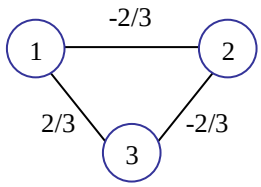


Figura 12. Red de Hopfield.

Supongamos que desear ver el patrón que le asocia al patrón de prueba (1 1 1). La red parte de la configuración inicial, $S_1(0)=1$, $S_2(0)=1$ y $S_3(0)=1$. A continuación se selecciona una unidad de proceso aleatoriamente, por ejemplo, la unidad 1. Su potencial sináptico viene dado por la expresión:

$$h_1 = \left(\frac{-2}{3}\right) \cdot 1 + \frac{2}{3} \cdot 1 = 0.$$

Por lo tanto, la regla de actualización dice que la unidad se queda en el estado que estaba.

Elegimos otra unidad de proceso para actualizar (segunda iteración), por ejemplo, la unidad 2. Su potencial sináptico

$$h_2 = \left(\frac{-2}{3}\right)\cdot 1 + \left(\frac{-2}{3}\right)\cdot 1 = \left(\frac{-4}{3}\right) < 0\,.$$

y así la unidad 2 cambia al estado $S_2(2) = -1$ (ver figura 3.4).

En este momento el estado de las neuronas es (configuración de la red):

$$S_1(2)= 1,\ S_2(2)= -1,\ S_3(2)= 1.$$

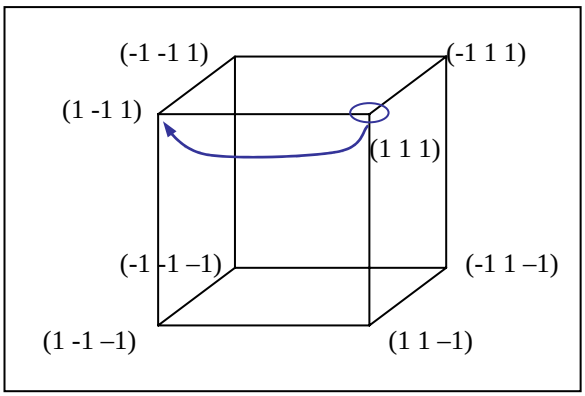


Figura 13. Cambio de la configuración (1 1 1) a la (1 -1 1).

En la iteración tres elegimos otra unidad de proceso, por ejemplo, la 3, y calculamos su potencial sináptico,

$$h_3 = \left(\frac{2}{3}\right)\cdot 1 + \left(\frac{-2}{3}\right)\cdot(-1) = \left(\frac{4}{3}\right) > 0$$

Como es positivo el estado que presenta es el 1 (igual al que presentaba)

Si en la iteración 4 elegimos la unidad 1, como su potencial sináptico

$$h_1 = \left(\frac{-2}{3}\right)\cdot(-1) + \left(\frac{2}{3}\right)\cdot 1 = \left(\frac{4}{3}\right) > 0$$

es positivo continua presentando el estado 1.

Si en la iteración 5 elegimos la unidad 2, como su potencial sináptico

$$h_2 = \left(\frac{-2}{3}\right)\cdot 1 + \left(\frac{-2}{3}\right)\cdot 1 = \left(\frac{-4}{3}\right) < 0$$

es negativo, sigue presentando el mismo valor -1.

Como no ha cambiado ninguna de las tres unidades de proceso, la red se ha estabilizado. Al patrón de prueba (1 1 1) le ha asociado el patrón memorizado (1 –1 1). Así, la red considera que el patrón de prueba (1 1 1) es una versión con ruido del patrón (1 –1 1).

Si hubiera memorizado solamente un patrón de los dos, por ejemplo, el (1 –1 1), hubiera salido el mismo resultado. Porque uno es el opuesto del otro y la red por defecto si memoriza un patrón también memoriza su opuesto.

$$\mathbf{w} = \frac{1}{3}\cdot\begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}\cdot\begin{pmatrix} 1 & -1 & 1 \end{pmatrix} - \frac{1}{3}\cdot\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -1/3 & 1/3 \\ -1/3 & 0 & -1/3 \\ 1/3 & -1/3 & 0 \end{pmatrix}$$

3.6 La Red BSB (Brain-State-in-a-Box)

Esta red fue propuesta por Anderson et al. (1972) y es similar a la red de Hopfield. Se utiliza como autoasociador de tareas, aunque también se extiende a una red con dos o más capas para la heteroasociación.

Está constituida por

- *N* unidades de proceso (tantas como la dimensión del espacio de los patrones a memorizar), todas ellas conectadas entre sí. La actualización se hace en unidades de tiempo discretas.

- Una *función de transferencia* f que es la función rampa dada por la expresión (ver la figura 14):

$$f(u_i(k)) = \begin{cases} +1 & \text{si} \quad u_i(k) \geq 1 \\ u_i(k) & \text{si} \quad -1 < u_i(k) < +1 \\ -1 & \text{si} \quad u_i(k) < -1 \end{cases}$$

  es decir,

$$f(u_i(k)) = \min(1, \max(-1, u_i(k)))$$

- La *dinámica de la computación* viene establecida por la ecuación:

$$x_i(k+1) = f\left(\sum_{j=1}^{N} w_{ij} x_j(k)\right),$$

que nos da el estado (salida) siguiente de la unidad de proceso *i*.

Se suele fijar el peso sináptico w_{ii}=1, i=1,2,...,*N*. La evolución de la red es la siguiente: comienza con un estado inicial de activación que es amplificado regularmente por retroacción positiva y sujeto a la condición de que las unidades de proceso se saturan en los valores –1 y 1. Así, los estados de la red estarán siempre en el hipercubo $[-1,1]^N$. La función rampa lleva a la red a un interesante comportamiento, en el que la red se mueve regularmente desde un punto arbitrario de dentro del hipercubo (caja)

hacia una cara del mismo y entonces se desplaza sobre la cara hasta alcanzar un vértice de la misma.

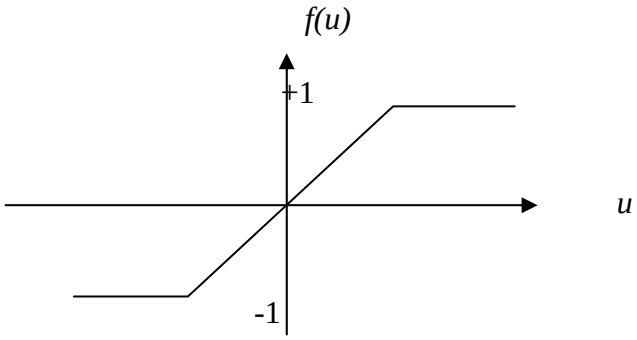


Figura 14. Función rampa con valores entre –1 y +1.

Los pesos sinápticos de las conexiones de la red se determinan a partir de un conjunto $s^r \in [-1,1]^N$, $r=1,2,...,p$, de p patrones bipolares llamados *memorias fundamentales*, mediante la regla de Hebb. Es decir, si el conjunto de patrones es fijo, se pueden determinar los pesos sinápticos mediante un proceso no iterativo, de manera que el peso w_{ij} viene dado por la ecuación:

$$w_{ij} = \frac{1}{p}\sum_{r=1}^{p} s_i^r s_j^r$$

donde s_j^r es la componente j del patrón r. También se pueden determinar de una forma iterativa mediante la siguiente expresión:

$$\Delta w_{ij} = \eta s_j^r \left(s_i^r - \sum_{k=1}^{N} w_{ik} s_k^r \right), \quad \text{r=1,2,...,}p. \tag{14}$$

donde $\eta \geq 0$ es una constante positiva prefijada. Esta regla corresponde a la minimización del error

$$\sum_{r=1}^{p}\sum_{j=1}^{N}\left(s_j^r - \sum_{k=1}^{N} w_{jk} s_k^r \right)^2$$

con respecto a w_{jk} siguiendo la técnica del descenso del gradiente. La regla de actualización (14) se aplica sucesivamente hasta que el error llegue a ser despreciable. Una vez que se ha terminado el entrenamiento se espera que

$$\sum_{r=1}^{p} \Delta w_{ij}^r = 0$$

es decir, que

$$\sum_{r=1}^{p} s_j^r \left(s_i^r - \sum_{k=1}^{N} w_{ik} s_k^r \right) = 0$$

Dicha ecuación se verifica si

$$s_i^r = \sum_{k=1}^{N} w_{ik} s_k^r \ .$$

De esta manera la red es estable para los patrones memorizados, es decir, cuando la red parte de un patrón memorizado se estabiliza en él y no cambia de estado.

Como en la red de Hopfield puede haber estados espurios, es decir, estabilizarse en estados que no corresponden a ninguno de los patrones memorizados ni sus opuestos.

3.7 Memoria asociativa bidireccional (BAM)

En 1987, Bart Kosko introduce un nuevo tipo de memoria asociativa: la memoria asociativa bidireccional (B.A.M). Se trata de una memoria heteroasociativa que asocia vectores bipolares (binarios) de distinta dimensión. Es decir, puede asociar a un código binario de 10 bits una firma digitalizada de 10.000 bits, o a una imagen 140.000 bits una imagen comprimida de 7.000 bits.
La BAM consta de dos capas de unidades de proceso, *n* unidades en la primera capa y *m* en la segunda, estando conectadas entre sí solamente las unidades de la primera capa con las unidades de la segunda (figura 15). Representaremos por w_{ij} el valor del peso sináptico de la conexión de la unidad *i* de la primera capa con la unidad *j* de la segunda, i=1,2,..,n, j=1,2,...,m, siendo estas conexiones bidireccionales.

La red bidireccional comienza con una configuración inicial y va actualizando simultáneamente en cada iteración todas las unidades de proceso de una capa y a continuación las de la otra capa, y así sucesivamente hasta que la red se estabilice (alcance una configuración de equilibrio). La dinámica de la computación de la red se define de forma similar a como se hace en la red de Hopfield, teniendo en cuenta ahora que las entradas de un unidad de proceso son los estados de las unidades de proceso de la otra capa. Por lo tanto, la regla de actualización de la red (dinámica de la computación) para las unidades de la primera capa viene dada por la siguiente expresión:

$$x_i(k+1) = \begin{cases} 1 & \text{si} \quad \sum_{j=1}^{m} w_{ij} y_j(k) > \theta_i \\ x_i(k) & \text{si} \quad \sum_{j=1}^{m} w_{ij} y_j(k) = \theta_i \\ -1 & \text{si} \quad \sum_{j=1}^{m} w_{ij} y_j(k) < \theta_i \end{cases} \quad , i = 1,2,\ldots,n \qquad (15)$$

y para las unidades de la segunda capa por:

$$y_j(k+1) = \begin{cases} 1 & \text{si} \quad \sum_{i=1}^{n} w_{ij} x_i(k) > \eta_j \\ y_j(k) & \text{si} \quad \sum_{i=1}^{n} w_{ij} x_i(k) = \eta_j \\ -1 & \text{si} \quad \sum_{i=1}^{n} w_{ij} x_i(k) < \eta_j \end{cases} \quad , j = 1,2,..,m \qquad (16)$$

Por lo tanto, el estado de las unidades de la segunda capa viene determinado por el estado de las unidades de la primera capa, y viceversa. Como las unidades de proceso son bipolares podemos tomar los valores umbral iguales a cero.

A continuación vamos a estudiar dicha dinámica de la computación. Para ello vamos a introducir una función de energía computacional que va a regir la evolución de la red, como con el modelo de Hopfield. La función de energía computacional de la BAM en la iteración *k* viene dada por la siguiente expresión:

$$E(k) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} x_i(k) y_j(k) + \sum_{i=1}^{n} \theta_i x_i(k) + \sum_{j=1}^{m} \eta_j y_j(k)$$

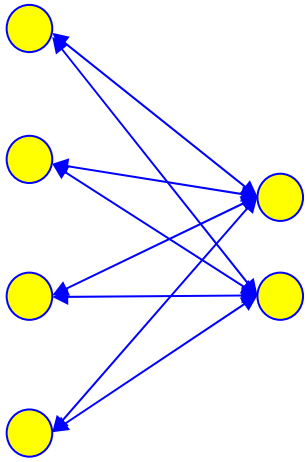


Figura 15. Arquitectura de una red BAM.

Al igual que en el modelo de Hopfield el modelo BAM evoluciona de forma que decrece la función de energía, o no cambia, en cada actualización. Por ello, se puede utilizar también esta red para resolver problemas de optimización combinatoria, sin más que identificar los valores de los pesos sinápticos con los coeficientes correspondientes de la función objetivo del problema de optimización.

Cuando en cada iteración sólo se actualiza una unidad de proceso de una capa y en la siguiente iteración se actualiza otra de la otra capa diremos que estamos siguiendo una actualización secuencial, mientras que si en una iteración actualizamos todas las unidades de una capa y en la siguiente iteración todas las unidades de la otra capa, diremos que estamos haciendo una actualización en paralelo. Vamos a ver que la red comienza en una configuración cualquiera y va evolucionando hacia estados de equilibrio en los que la red se estabiliza. Estos estados de equilibrio van a ser los atractores de la red.

Teorema 1. Una memoria asociativa bidireccional con una matriz de pesos sinápticos arbitraria la función de energía computacional decrece, o no cambia, en cada actualización, y la red alcanza un estado estable (estado de equilibrio) después de un número finito de actualizaciones, tanto en modo secuencial como en modo paralelo.

Demostración: Supongamos que en la iteración *k* vamos a actualizar la unidades de proceso de la segunda capa. Entonces,

$$
\begin{aligned}
E(k+1)-E(k) &= -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}x_i(k)y_j(k+1) + \sum_{i=1}^{n}\theta_i x_i(k) + \sum_{j=1}^{m}\eta_j y_j(k+1) \\
&\quad + \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}x_i(k)y_j(k) - \sum_{i=1}^{n}\theta_i x_i(k) - \sum_{j=1}^{m}\eta_j y_j(k) \\
&= -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}x_i(k)\left[y_j(k+1)-y_j(k)\right] + \sum_{j=1}^{m}\eta_j\left[y_j(k+1)-y_j(k)\right] \\
&= -\sum_{j=1}^{m}\left[y_j(k+1)-y_j(k)\right]\left[\sum_{i=1}^{n} w_{ij}x_i(k) - \eta_j\right] \\
&\le 0
\end{aligned}
$$

pues si $\sum_{i=1}^{n} w_{ij}x_i(k) - \eta_j > 0$ entonces $y_i(k+1)=1 \ge y_i(k)$,

y si $\sum_{i=1}^{n} w_{ij}x_i(k) - \eta_j < 0$ entonces $y_i(k+1)= -1 \le y_i(k)$.

Por lo tanto, como la red sólo cambia de configuración cuando $\sum_{i=1}^{n} w_{ij}x_i(k) - \eta_j \neq 0$, para algún *j*, en cuyo caso alcanza un menor valor de la función de energía, y el número de configuraciones posibles de la red es finito ($2^m \cdot 2^n$), entonces la red se tiene que estabilizar en un número finito de iteraciones. ■

La red BAM se puede utilizar también como una memoria asociativa en cuyo caso los pesos de la red se determinan a partir de *p* pares de patrones, llamados memorias fundamentales, cada par viene dado por un patrón y su código, es decir, se pretende que la red memorice los *p* pares de patrones siguientes:

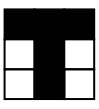
$$\left\{\mathbf{x}^k = \left(x_1^k, x_2^k, ..., x_n^k\right),\ \mathbf{y}^k = \left(y_1^k, y_2^k, ..., y_m^k\right);\ k = 1,2,..., p\right\}$$

Dichos patrones fundamentales actuarán de atractores, es decir, cuando la red parte de un patrón de prueba que no sea fundamental la red se va a estabilizar en uno de los patrones fundamentales (o sus opuestos), el más parecido al patrón de prueba, que considerará como una versión con ruido del correspondiente patrón memorizado. Para ello se determinan los pesos sinápticos de la red mediante la regla de Hebb:

$$W = \sum_{k=1}^{p} (\mathbf{x}^k)^T \mathbf{y}^k \qquad (17)$$

donde el elemento w_{ij} de la matriz *W* nos da el peso sináptico de la conexión entre la unidad *i* de la primera capa y la unidad *j* de la segunda capa.

Ejemplo: Supongamos que se desea memorizar los patrones y códigos siguientes:

 ↔ (1 -1 -1)

 ↔ (-1 -1 1)

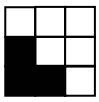
Es decir, tenemos como memorias fundamentales los pares,

$$(1\ 1\ 1\ {-1}\ 1\ {-1}\ {-1}\ 1\ {-1}) \leftrightarrow (1\ {-1}\ {-1})$$
$$(1\ {-1}\ {-1}\ 1\ {-1}\ {-1}\ 1\ 1\ 1) \leftrightarrow ({-1}\ {-1}\ 1)$$

Según la expresión (17), la matriz de pesos sinápticos es:

$$W = \begin{pmatrix} 0 & -2 & 0 \\ 2 & 0 & -2 \\ 2 & 0 & -2 \\ -2 & 0 & 2 \\ 2 & 0 & -2 \\ 0 & 2 & 0 \\ -2 & 0 & 2 \\ 0 & -2 & 0 \\ -2 & 0 & 2 \end{pmatrix}$$

Si le damos a la red como entrada en la primera capa el patrón de prueba



que es una versión con ruido del segundo patrón (L), entonces las unidades de la segunda capa se actualizan según su potencial sináptico

$$(-1\ -1\ -1\ 1\ -1\ -1\ 1\ 1\ -1) \begin{pmatrix} 0 & -2 & 0 \\ 2 & 0 & -2 \\ 2 & 0 & -2 \\ -2 & 0 & 2 \\ 2 & 0 & -2 \\ 0 & 2 & 0 \\ -2 & 0 & 2 \\ 0 & -2 & 0 \\ -2 & 0 & 2 \end{pmatrix} = (-8\ \ -2\ \ 8)$$

Como el potencial sináptico de la primera unidad de la segunda capa vale -8 entonces dicha unidad presenta el estado -1; análogamente, como la segunda unidad de la segunda capa tiene un potencial sináptico igual a -2 presenta también el estado -1 y la tercera capa presenta el estado 1 puesto que su potencial sináptico es positivo (igual a 8). Por lo tanto el código asignado es (-1 -1 1). A continuación se actualizan las unidades de la primera capa según el estado que presentan las de la segunda. Su potencial sináptico es:

$$\begin{pmatrix} 0 & -2 & 0 \\ 2 & 0 & -2 \\ 2 & 0 & -2 \\ -2 & 0 & 2 \\ 2 & 0 & -2 \\ 0 & 2 & 0 \\ -2 & 0 & 2 \\ 0 & -2 & 0 \\ -2 & 0 & 2 \end{pmatrix} \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ -4 \\ -4 \\ 4 \\ -4 \\ -2 \\ 4 \\ 2 \\ 4 \end{pmatrix}$$

Por lo tanto, las unidades de la primera capa presentan la configuración dada por el vector $(1\,-1\,-1\;1\,-1\,-1\;1\;1\;1)$ que corresponde al segundo patrón fundamental. Si actualizamos de nuevo las unidades de la segunda capa

$$(1\,-1\,-1\;1\,-1\,-1\;1\;1\;1)\begin{pmatrix} 0 & -2 & 0 \\ 2 & 0 & -2 \\ 2 & 0 & -2 \\ -2 & 0 & 2 \\ 2 & 0 & -2 \\ 0 & 2 & 0 \\ -2 & 0 & 2 \\ 0 & -2 & 0 \\ -2 & 0 & 2 \end{pmatrix} = (-12\;\; -6\;\; 12)$$

las unidades de proceso presentan la configuración dada por el vector (-1 -1 1) que es el mismo código que presentaban, correspondiente a al segundo patrón memorizado. La red ya no cambia, se ha estabilizado. Por lo tanto, la red le ha asociado al patrón de prueba el segundo patrón memorizado y su correspondiente código. Considera que el patrón de prueba es una versión con ruido de la letra L memorizada.