# Red de Hopfield y Crecimiento de redes

23 de junio de 2009

# Índice general



[1] *Growing Cell Strutures.*
[2] *Growing Neural Gas.*

# Capítulo 1

# Red de Hopfield

Una de las mayores contribuciones al área de las redes neuronales fue realizada en los años 1980 por John Hopfield, quien estudió modelos aoutoasociativos que presentaban algunas similaridades con los perceptrones, pero que también incluían grandes diferencias.

Debido a la arquitectura y al funcionamiento, la red de Hopfield se puede incluir dentro de las redes de neuronas recurrentes, pues cada neurona está conectada con todas las demás y además se produce un procesamiento temporal de los patrones. Lo que diferencia a esta red de las demás redes recurrentes es que actúa a la manera de una memoria asociativa.

El concepto de memoria asociativa es bastante intuitivo: se trata simplemente de asociar dos patrones. Dentro de este concepto definiremos diferentes tipos de memorias asociativas:

- **Memoria heteroasociativa:** Establece una correspondencia $F$ entre dos vectores $\vec{x}$ e $\vec{y}$ de tal manera que $F(\vec{x}_i) = \vec{y}_i$, y si un $\vec{x}$ arbitrario está más próximo a $\vec{x}_i$ que a cualquier otro $\vec{x}_j$, entonces $F(\vec{x}) = \vec{y}_i$. En esta definición, el estar más próximo quiere decir con respecto a la *Distancia de Hamming*.

- **Memoria asociativa interpoladora:** Establece una correspondencia $F$ entre $\vec{x}$ e $\vec{y}$ de tal manera que $F(\vec{x}_i) = \vec{y}_i$, pero si el vector de entrada difiere de uno de los ejemplares en el vector $\vec{d}$, de tal modo que $\vec{x} = \vec{x}_i + \vec{d}$, entonces $F(\vec{x}) = \vec{y}_i + \vec{d}$.

- **Memoria autoasociativa:** Presupone que $\vec{y} = \vec{x}$ y establece una correspondencia $F$ de $\vec{x}$ en $\vec{x}$ tal que $F(\vec{x}) = \vec{x}$, y si algún $\vec{x}_i$ arbitrario está más próximo a $\vec{x}$ que a cualquier otro vector $\vec{x}_j$, entonces $F(\vec{x}_i) = \vec{x}$.

Hemos realizado las anteriores definiciones para centrarnos ahora en el modelo de Hopfield. Primero diremos que la estructura de red neuronal que propone viene ha ser una memoria autoasociativa de una sóla capa, totalmente conectada y recurrente.

La red de Hopfield está formada por $n$ neuronas, cada una conectada a todas las demás salvo a ella misma. La matriz de pesos de la red de Hopfield es una matriz $W = (w_{ij})$ de orden $n \times n$, donde $w_{ij}$ representa el peso de la conexión de la neurona $i$ a la neurona $j$. Dicha matriz posee las siguientes particularidades:

- Es una matriz simétrica, es decir, $w_{ij} = w_{ji}\ \forall i, j = 1, 2, \ldots, n$. Esto implica que el peso de la conexión entre dos neuronas tiene el mismo valor en ambos sentidos.
- Los elementos de la diagonal de la matriz son iguales a cero, es decir, $w_{ii} = 0\ \forall i = 1, 2, \ldots, n$, puesto que no existen conexiones de una neurona a sí misma.

Las neuronas de la red Hopfield poseen dos estados, generalmente $-1$ y 1, que vienen determinados por el nivel o potencial de activación que recibe la neurona. De este modo, el estado de la neurona $i$ en un instante de tiempo $t + 1$ viene determinado por su función de transferencia, que denotaremos por $s_i(t + 1)$. Esta función es, generalmente, una *hardlimiter* y viene dada por:

$$s_i(t+1) = sgn\big(v_i(t+1)\big) \quad \text{para } i = 1, 2, \ldots, n \tag{1.1}$$

donde $sgn$ es la función signo definida como:

$$sng(x) = \begin{cases} +1 & \text{si } x > 0 \\ -1 & \text{si } x < 0 \end{cases} \tag{1.2}$$

y $v_i(t+1)$ es el nivel de activación, resultado de la función de activación (que en esta red es igual al valor de entrada neto del la neurona), que actúa sobre la neurona $i$ y que se calcula como:

$$v_i(t+1) = \sum_{j=1}^{n} w_{ji} s_j(t) - u_i \quad \text{para } i = 1, 2, \ldots, n \tag{1.3}$$

donde $u_j$ es un umbral fijo aplicado a la neurona $i$ y $s_j(t)$ es el estado de la neurona $j$ en el instante anterior de tiempo, $t$.

En el caso de que el nivel de activación que recibe la neurona, $v_i(t + 1)$, sea igual a cero se considera que el estado de la neurona no varía respecto al instante de tiempo anterior, es decir, $s_i(t + 1) = s_i(t)$.

Incluyendo todas sus características podríamos expresar $s_i(t+1)$ también de la siguiente forma:

$$s_i(t+1) = \begin{cases} +1 & \text{si } v_i(t) > 0 \\ s_i(t) & \text{si } v_i(t) = 0 \\ -1 & \text{si } v_i(t) < 0 \end{cases} \tag{1.4}$$

De las definiciones anteriores podemos deducir que para la red de Hopfield no tiene sentido hablar de neuronas de entrada o salida de la red, sino del estado de la red en cada instante de tiempo. Para una red de Hopfield con $n$ neuronas el estado viene dado por:

$$\vec{s}(t+1) = [s_1(t+1), s_2(t+1), \ldots, s_n(t+1)]^t \tag{1.5}$$

donde $^t$ denota la matriz traspuesta. Dicho estado $\vec{s}$ representa una palabra binaria de $n$ bits de información.

## 1.1. Aprendizaje y funcionamiento de la red de Hopfield

La operación de la red de Hopfield es totalmente diferente al sistema del perceptrón. En el modelo de Hopfield, la primera salida es tomada como entrada en el ciclo siguiente, produciendo una nueva salida. Por tanto el aprendizaje es también diferente; en este sistema no se trata de ajustar pesos, ya que éstos se mantienen constantes desde el principio, si no de encontrar dichos pesos en función del problema. De tal manera que el conjunto total del sistema puede venir representado por una función denominada Función de Energía.

En la red de Hopfield se distinguen dos fases de operación llamadas: *fase de almacenamiento* y *fase de recuperación*. Durante la fase de almacenamiento se van a determinar los valores que deben tomar los pesos de la red para almacenar un conjunto de patrones, y la fase de recuperación describe el mecanismo para recuperar la información almacenada a partir de información incompleta.

### 1.1.1. Fase de almacenamiento

Sea $\left\{\vec{x}(k) = \big(x_1(k), x_2(k), \ldots, x_n(k)\big)\right\}_{k=1,\ldots,p}$ el conjunto de $p$ patrones que se desea almacenar, donde cada patrón $\vec{x}(k)$ es un vector $n$-dimensional cuyas componentes toman **valores binarios**, es decir $-1$ ó 1. Para almacenar patrones, el peso de la conexión de la neurona $j$ a la $i$ viene dado por:

$$w_{ij} = \sum_{k=1}^{p} x_j(k) x_i(k) \quad \forall i \neq j \tag{1.6}$$

Esta ecuación cumple la regla de Hebb, pues si $x_i(k)$ y $x_j(k)$ son iguales el peso de la conexión de $i$ a $j$ se incrementa en una unidad, mientras que si son distintas se reducen en la misma cantidad.

### 1.1.2. Fase de recuperación

Sea $\vec{x} = (x_1, x_2, \ldots, x_n)$ un patrón de prueba diferente de los patrones almacenados en la fase anterior. Dicho patrón representa generalmente una versión de uno de los vectores $\vec{x}(k)$ almacenados pero con información incompleta o al que se le a incluido o ruido. En esta fase la red de Hopfield recuperará el patrón almacenado más próximo a $\vec{x}$. El procedimiento seguido será el siguiente:

Paso 1. Se inicializan los estados de las $n$ neuronas de la red utilizando el patrón $\vec{x}$:

$$s_i(0) = x_i \quad \text{para } i = 1, 2, \ldots, n$$

Paso 2. Se espera a que la red alcance un estado *estable* o punto fijo, entendiendo por tal aquel en el que el estado de los elementos de la red permanezca invariante en el tiempo:

$$s_i(t+1) = s_i(t) \quad \forall i = 1, 2, \ldots, n$$

Este estado estable de la red representará el patrón recuperado a partir del patrón de prueba $\vec{x}$.

## 1.2. Función de energía

En la red de Hopfield, los pesos se calculan por adelantado y no forman parte de un sistema dinánimo. En esta situación los vectores de salida cambian en función del tiempo y si forman parte de un sistema dinámico. En este sistema nos interesará contestar a las preguntas siguientes: ¿Existe solución?, y en caso afirmativo ¿Convergerá el sistema en un tiempo finito?.

En el caso de la Red de Hopfield podemos demostrar que esa función existe:

Para otros tipos de redes se describía un proceso iterativo para hallar los pesos adecuados para resolver un problema concreto. En dichas descripciones

cada punto del *espacio de pesos* tenía asociado un cierto valor de error. Para algunos casos sencillos, utilizando los valores posibles de los pesos y la función de error, podíamos obtener una representación gráfica de la *superfície de error*. La tarea de obtener unos pesos óptimos para nuestro problema se identificaba con la obtención de los pesos que se correspondiesen con un valle de dicha superficie de error. Durante el proceso de entrenamiento los **pesos cambian a lo largo del tiempo** (iteraciones) formando un **sistema dinámico**.

Para la red de Hopfield se produce un situación ligeramente distinta. Los pesos se calculan por adelantado y no forman parte de un sistema dinámico mientras los que si lo forman son los vectores de salidas de los elementos de procesado de la red, que cambian en función del tiempo. De este modo una trama cualquiera que se presente a la red puede hacer necesarias varias iteraciones hasta que la red se estabilice. En esta situación son los vectores $\vec{s}$ los que cambian en función del tiempo formando un sistema dinámico.

Con respecto a los sistemas dinámicos de los que hablamos[1] nos interesaban varios aspectos de su conducta: ¿Existe una solución? De ser así, ¿convergerá en un tiempo finito? ¿Cuál es la solución? Hasta el momento nos centrábamos en esta última cuestión. Ahora, para el caso de la red de Hopfield, examinaremos las otras dos.

Hemos visto que en otras redes existe una función error o *función de energía* que describe el comportamiento y permite entender su funcionamiento. En el caso de la red de Hopfield también existe dicha función, la cual nos disponemos a analizar... pero no si antes hablar un poco de la *Memoria Bidireccional Asociativa*[2] y de su propia función de error.

En la teoría de sitemas dinámicos se puede probar un teorma acerca de la existencia de estados estables que emplea el concepto de una función llamada *Función de Lyapuov* o **Función de Energía**: "Si se puede hallar una función acotada de las variables de estado de un sistema dinámico, tal que todos los cambios de estado den lugar a una disminución del valor de la función, entonces el sistema posee una solución estable".

### 1.2.1. La BAM

La BAM es similar a una red de Hopfield[3] con unas pequeñas modificaciones: no hay umbrales y las neuronas están organizadas en dos capas.

[1] Referidos a la evolución de los pesos y, ahora, a la evolución de las activaciones de los elementos de la red.

[2] BAM para los amigos.

[3] Por no decir que una red de Hopfield es una variación de la BAM.

Llamaremos a estas capas capa $X$ y capa $Y$, y al vector estado de las neuronas cada capa $\vec{x}$ y $\vec{y}$ respectivamente.

Sean $\vec{x}_i \in \mathbb{R}^n$ y $\vec{y}_i \in \mathbb{R}^m$ con $i = 1, 2, \ldots, L$, o mejor dicho: $\{(\vec{x}_i, \vec{y}_i)\}_{i=1,2,\ldots,L}$ el conjunto de $L$ vectores que queremos almacenar, de forma que el vector $\vec{x}_i$ tenga asociado el vector $\vec{y}_i$ y viceversa (de ahí lo de bidireccional). La matriz de pesos para dicho conjunto de vectores a almacenar se construirá como:

$$W = \vec{y}_1\vec{x}_1^t + \vec{y}_2\vec{x}_2^t + \ldots + \vec{y}_L\vec{x}_L^t = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{pmatrix} \tag{1.7}$$

Esta ecuación da los pesos de las conexiones *procedentes* de la capa $x$ con *destino* a la capa $Y$. $w_{ij}$ será el peso de la conexión entre el elemento $i$ de la capa $X$ al elemento $j$ de la capa $Y$. Si deseamos obtener la matriz que nos indique los pesos de la capa $Y$ a la $X$ sólo tendremos que trasponer al matriz $W$.

### Función de energía de la BAM

Como indicábamos al comienzo de esta sección, las preguntas para las que buscamos respuestas son ¿Existe una solución? y ¿Convergerá el sistema (a ella) en un tiempo finito?

Puesto que hemos encontrado respuestas a nuestros problemas, daremos por sentado que la respuesta existe[4]. Examinaremos ahora si es posible determinar si la BAM converge *siempre* a una solución estable. En la teoría de sistemas dinámicos se puede probar un teorema acerca de la existencia de estados estables que emplea el concepto de una función llamada *función de Lyapunov* o **función energía**. Si se puede hallar una función acotada de las variables de un sistema dinámico, tal que todos los cambios de estado den lugar a una disminución del valor de la función, entonces el sistema posee una solución estable. Esta función es la llamada función de Lyapunov o función energía. En el caso de la BAM, esta función energía tiene la siguiente forma:

$$E = -\vec{y}^t\vec{w}\vec{x} = -\sum_{i=1}^{m}\sum_{j=1}^{n} y_i w_{ij} x_j \tag{1.8}$$

[4]si, con un par.

**Teorema sobre la función energía de la BAM**

Ahora enunciaremos un teorema sobre la función energía de la BAM:

1. Todo cambio de $\vec{x}$ o de $\vec{y}$ durante el proceso de la BAM dará lugar a una disminución de $E$.
2. $E$ posee un límite inferior dado por $E_{\min} = -\sum_{i=1,j=1}^{m,n} |w_{ij}|$.
3. Cuando $E$ cambia, debe hacerlo en una cantidad finita.

Los puntos 1 y 2 demostrarán que $E$ es una función de Lyapunov, y que el sistema dinámico posee un estado estable. El punto 2 muestra que $E$ puede disminuir sólo hasta cierto valor. El punto 3 previene la situación en que las variaciones de $E$ se volviesen infinitamente pequeñas, de forma que $E$ no pudiese alcanzar un valor mínimo en un tiempo finito.

**Demostración del teorema sobre la función energía de la BAM**

Por desgracia, demostraremos únicamente el primer punto del teorema.
Partimos pues de la ecuación de la función energía o de Lyapunov:

$$E = -\sum_{i=1}^{m}\sum_{j=1}^{n} y_i w_{ij} x_j$$

De acuerdo con el teorema, todo cambio en $\vec{x}$ o en $\vec{y}$ debe dar lugar a una disminución del valor de $E$. Consideraremos en primer lugar un cambio en una sola componente de $\vec{y}$, concretamente su componente $y_k$.

Reescribiendo la Ecuación 1.8 separando el término que incluye a $y_k$ tendremos:

$$E = -\sum_{\substack{i=1\\ i\neq k}}^{n}\sum_{j=1}^{m} y_i w_{ij} x_j - \sum_{j=1}^{n} y_k w_{kj} x_j \tag{1.9}$$

ahora aremos el cambio $y_k \leftarrow y_k^{\mathsf{nueva}}$, de modo que el nuevo valor de $E$ será:

$$E = -\sum_{\substack{i=1\\ i\neq k}}^{m}\sum_{j=1}^{n} y_i w_{ij} x_j - \sum_{j=1}^{n} y_k^{\mathsf{nueva}} w_{kj} x_j \tag{1.10}$$

Expresaremos ahora la variación de $E$, $\Delta E$ en función de la variación de $\vec{y}_k$:

$$\begin{aligned}
\Delta E &= E^{\mathsf{nueva}} - E = -\sum_{j=1}^{n} y_k^{\mathsf{nueva}} w_{kj} x_j + \sum_{j=1}^{n} y_k w_{kj} x_j = \\
&= -y_k^{\mathsf{nueva}} \sum_{j=1}^{n} w_{kj} x_j + y_k \sum_{j=1}^{n} w_{kj} x_j \\
\Delta E &= (y_k - y_k^{\mathsf{nueva}}) \sum_{j=1}^{n} w_{kj} x_j \qquad (1.11)
\end{aligned}$$

Teniendo en cuenta que la expresión de $y_k$ es la siguiente:

$$y_k^{\mathsf{nueva}} (\equiv s_k) = \begin{cases} +1 & \text{si } v_k^y > 0 \\ y_k & \text{si } v_k^y = 0 \\ -1 & \text{si } v_k^y < 0 \end{cases}$$

donde $v_k^y = \sum_{j=1}^{n} w_{kj} x_j$.

Se pueden considerar dos posibles cambios de $y_k$:

- $y_k$ pasa de valer $+1$ a $-1$. Entonces $(y_k - y_k^{\mathsf{nueva}}) > 0$. Pero este cambio sólo puede darse cuando $\sum_{j=1}^{n} w_{kj} x_j < 0$, por lo que tendremos que:

$$\Delta E = \overbrace{(y_k - y_k^{\mathsf{nueva}})}^{>0} \overbrace{\sum_{j=1}^{n} w_{kj} x_j}^{<0} < 0$$

- $y_k$ pase de $-1$ a $+1$. En cuyo caso tendremos:

$$\Delta E = \overbrace{(y_k - y_k^{\mathsf{nueva}})}^{<0} \overbrace{\sum_{j=1}^{n} w_{kj} x_j}^{>0} < 0$$

En el caso de que $y_k$ no varíe tendremos que $(y_k - y_k^{\mathsf{nueva}}) = 0$ con lo que $E$ no varía.

De forma análoga podrá demostrase que se cumple la misma situación para cambios de más de una componente de $\vec{y}$ y de más de una componente de $\vec{x}$.

**Retomemos ahora el caso de la red de Hopfield.**

Mantengamos presentes los resultados obtenidos sobre la función de energía de la BAM.

Sea una red de Hopfield con $n$ neuronas y con conexiones $W = (w_{ij})$, siendo $W$ una matriz simétrica con ceros en su diagonal. La función de energía asociada a dicha red viene dada por la expresión:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j\neq i}^{n} w_{ij}s_i s_j + \sum_{i=1}^{n} u_i s_i \tag{1.12}$$

Esta es una función válida pues contiene todos los elementos variables en el tiempo de la red.

Separando la contribución de una neurona $k$ a la función energía (Ecuación 1.12), dada por la expresión anterior, tendremos:

$$E = -\frac{1}{2}\sum_{\substack{i=1\\ i\neq k}}^{n}\sum_{\substack{j=1\\ j\neq k}}^{n} s_i w_{ij} s_j + \sum_{\substack{i=1\\ i\neq k}}^{n} u_i s_i \overbrace{-\frac{1}{2}s_k\sum_{i=1}^{n} s_i w_{ik} - \frac{1}{2}s_k\sum_{j=1}^{n} w_{kj}s_j + u_k s_k^{\text{nueva}}}^{\text{Contribución del elemento } k.} \tag{1.13}$$

Supongamos ahora que el elemento $k$, y sólo él, sufre una modificación. La variación de $E$ ($\Delta E$) dependerá solo de la variación del elemento $k$, los términos estables se anularán pues $\Delta E = E(t+1) - E(t)$:

$$\Delta E = -\frac{1}{2}\left(\Delta s_k\sum_{i=1}^{n} s_i w_{ik} + \Delta s_k\sum_{j=1}^{n} w_{kj}s_j + \Delta s_k u_k\right) \tag{1.14}$$

Teniendo en cuenta que la matriz de pesos es simétrica y (ojo que ahora vamos a hacer *magia*) suponiendo que el umbral $u_k$ es cero porque: *"al tratar de minimizar el valor de E en algunos casos el término umbral puede incrementar dicho valor y en otros disminuirlos, de tal manera que para que no exista distinta influencia sobre diferentes entradas se le asigna un valor cero."*
Tendremos :

$$\begin{aligned}
\Delta E &= -\frac{1}{2}\left(\Delta s_k\sum_{i=1}^{n} s_i w_{ik} + \Delta s_k\sum_{j=1}^{n} w_{kj}s_j + \Delta s_k u_k\right) \approx \\
&\approx -\frac{1}{2}\left(\Delta s_k\sum_{i=1}^{n} s_i w_{ik} + \Delta s_k\sum_{j=1}^{n} w_{kj}x_j\right) \\
\Delta E &= -\Delta s_k\sum_{i=1}^{n} w_i k s_i
\end{aligned} \tag{1.15}$$

Los posibles cambios del término $s_k$ son:

- De $+1$ a $-1$, en cuyo caso $\Delta s_k < 0$ y por las equaciones (1.1), (1.2) y (1.3) sabemos que $s_k$ si cambia a $-1$ el término $\sum_i w_{ik} s_i < 0$ por lo que $\Delta E < 0$ y $E$ decrece.
- De $-1$ a $+1$, lo que nos deja con $\Delta s_k > 0$ y $\sum_i w_{ik} s_i > 0$ con lo que $\Delta E < 0$ y $E$ decrece también.

# Capítulo 2

# Crecimiento de redes

## 2.1. Introducción

Al trabajar con mapas autoorganizativos tenemos que definir *a priori* la estructura de la red. Determinar la definición adecuada suele ser una tarea bastante compleja debido a la poca información de la que se dispone sobre el espacio de entrada (su probabilidad de distribución). Para complicarnos aún más las cosas, una vez decidida su estructura, ésta es inalterable.

Para solucionar este problema surge una nueva filosofía a la hora de aplicar redes de neuronas a problemas de clasificación: el *crecimiento de redes*. Se trata de que sea la propia red la que determine de forma autónoma su estructura, es decir, el número de elementos de procesado y conexiones entre ellos.

Estas redes se construyen partiendo de estructuras básicas que son hipertetraedros de dimensión $k$. Para llegar a la estructura final de la red se realiza un proceso donde se van añadiendo y borrando elementos al tiempo que se sigue un proceso de autoorganización similar al que se produce en el modelo propuesto por Kohonen.

### 2.1.1. Inserción de neuronas

Para poder determinar dónde y cómo hemos de incluir una nueva neurona se introduce un nuevo concepto, el *valor resource*. El *valor resource* es un valor asignado a cada neurona que medirá lo lejos que nos encontramos de la estructura ideal. Este valor irá variando a medida que la red vaya evolucionando.

Siempre después de un número constante de adaptaciones sobre la red actual, $\lambda$, se analizará la necesidad o no de añadir nuevas neuronas. Añadir un

nuevo elemento a la red conlleva el adecuar las conexiones entre los elementos vecinos para que la nueva estructura sea consistente.

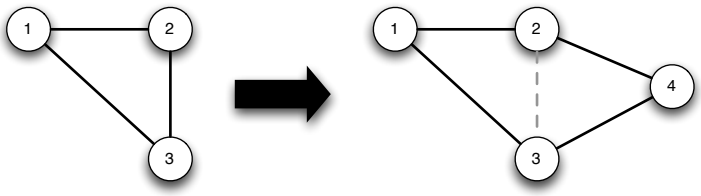


Figura 2.1: Tanto el vector de referencia de la nueva neurona como el valor de su resource se calcularán a partir de los parámetros de sus neuronas vecinas.

### 2.1.2. Borrado de neuronas

Cuando un conjunto de vectores de entrada que pueden activar una determinada neurona tiene una probabilidad de aparición muy baja, podemos estimar que el coste de mantenerla dentro de la estructura de la red no nos compensa y por lo tanto decidir eliminarla.

### 2.1.3. Propiedades comunes y ventajas del crecimiento

Propiedades comunes:

- La estructura de la red es un grafo constituido por un número de nodos y un conjunto de aristas que los conectan.
- Cada nodo tiene un vector de referencia en el espacio de entradas (prototipo).
- La adaptación de los vectores de referencia se realiza generando una señal de entrada y cambiando el vector de la ganadora y sus vecinas.
- En cada paso de adaptación la información del error local se almacena en la ganadora.
- El error acumulado se usa para determinar dónde insertar nuevas unidades en la red.
- Normalmente todos los parámetros del modelo son constantes en el tiempo.

Ventajas del crecimiento:

- Posibilidad de usar problemas dependientes de medidas de error para determinar dónde insertar nuevas unidades.
- Posibilidad de interrumpir la autoorganización o continuar con una interrumpida previamente.
- Pocos *números mágicos* a definir. El tamaño de la red no se define de antemano pero puede definirse en base a ciertos criterios y obtenerse en el proceso de ajuste.

## 2.2. Descripción del problema

Sea $V = \mathbb{R}^n$ el espacio de los vectores de entrada $n$-dimensionales. Las señales de entrada siguen la distribución de probabilidad $P(\xi)$ **desconocida**. Nuestro objetivo es obtener un mapeo desde $V$ a una estructura topológica discreta y $k$-dimensional $A$, es decir, realizar una **clasificación** de $V$ sobre la estructura $A$. Se deberán cumplir las siguientes propiedades:

- Vectores similares han de ser clasificados en elementos de $A$ que sean topológicamente próximos.
- Elementos de $A$ que sean vecinos tendrán que clasificar vectores similares del espacio de entrada.
- Regiones de $V$ que posean una densidad alta serán representados por un número elevado de elementos de $A$.

Los dos primeros puntos se refieren a que el mapeo debe preservar las relaciones de similitud y cercanía, lo que permite que la complejidad de los datos sea reducida sin un pérdida de información. La tercera propiedad se refiere a que es posible obtener alguna información sobre la densidad de los vectores de entrada, que en un principio es desconocida.

## 2.3. Crecimiento de estructuras de células (GCS)[1]

La red GCS proporciona una estructura flexible que inicialmente consta un número variable de elementos de procesado y una topología dimensional, donde ese valor puede ser escogido aleatoriamente manteniendo una estructura compacta de la red.

[1] *Growing Cell Strutures.*

### 2.3.1. Definición del problema

Partimos de un conjunto de señales de entrada de dimensión $n$ y con una probabilidad de distribución desconocida $V = P(E)$.

El objetivo es realizar una clasificación de $V$ sobre una estructura $k$-dimensional $A$ con las siguientes propiedades:

- Vectores similares han de ser clasificados en elementos de $A$ que se encuentren topológicamente próximos.
- Elementos de $A$ que sean vecinos tendrán que clasificar vectores similares del espacio de entrada.
- Regiones de $V$ que posean una densidad alta serán representadas por un número grande de elementos de $A$ y viceversa.

### 2.3.2. Estructura de la red

La topología inicial de la red será una estructura $k$-dimensional (para $k = 1$ es un segmento, para $k = 2$ se trata de un triángulo y para $k > 3$ se trata de un hipertetraedro) donde existirán $k + 1$ vértices y $k(k + 1)/2$ enlaces.

Cada neurona poseerá un vector de referencia o vector de pesos ($\vec{w}_c$) de dimensión $n$ que se puede interpretar como la posición que ocupa la neurona sobre el espacio de entradas.

Se considera que para cada vector de entrada $\xi$ existe un elemento de procesado ganador, $c$, tal que:

$$\forall i \neq c, i \in A, \|\vec{w}_i - \xi\| > \|\vec{w}_c - \xi\|$$

donde $\|\|$ representa la **distancia euclídea**.

### 2.3.3. Funcionamiento de la estructura

Ecuaciones de actualización de pesos de la célula ganadora $s$ y sus vecinas:

$$\Delta\vec{w}_s = \beta_s(\xi - \vec{w}_s) \quad \text{para } s \text{ la célula ganadora.} \tag{2.1}$$

$$\Delta\vec{w}_c = \beta_c(\xi - \vec{w}_c) \quad \forall v \in N_s \tag{2.2}$$

donde $N_s$ representa al conjunto de vecinas directas de $s$.

Regla para la actualización de los contadores:

$$\Delta\tau_s = 1 \quad \text{para } s \text{ la célula ganadora.} \tag{2.3}$$
$$\Delta\tau_c = -\alpha\tau_c \quad \forall c \in A \tag{2.4}$$

Sea $N_c$ el conjunto de vecinos directos del elemento de procesado $c$. Definiremos un contador $\tau_c$ que contiene el número de patrones de entrada para los que el elemento $c$ ha resultado ganador. Puesto que las señales de entrada actuales deben tener un peso mayor que las anteriores en el tiempo, durante el proceso de aprendizaje se decrementará esta variable.

El algoritmo de aprendizaje será como sigue:

Paso 1. Se escoge un vector de entrada $\xi$.

Paso 2. Se determina la unidad ganadora $s$.

Paso 3. Se actualizan los vectores de pesos de la neurona ganadora $s$ y sus vecinos directos:

$$\Delta\vec{w}_s = \beta_s(\xi - \vec{w}_s)$$
$$\Delta\vec{w}_c = \beta_c(\xi - \vec{w}_c)\forall c \in N_s$$

donde $N_s$ representa al conjunto de todos los vecinos directos del elemento $s$.

Paso 4. Se incrementa el contador de la neurona ganadora siguiendo la Ecuación (2.3) con lo que $\tau_s = \tau_s + 1$.

Paso 5. Se decrementan **todos los contadores** una fracción $\alpha$ siguiendo la Ecuación (2.4), de modo que $\tau_c = \tau_c - \alpha\tau_c, \quad \forall c \in A$.

Si se escogen valores pequeños para $\beta_s$ y $\beta_c$ las neuronas se moverán desde sus posiciones iniciales aleatorias con un cierto equilibrio en todas las direcciones. El movimiento no cesará hasta que los parámetros de adaptación no disminuyan. Siempre se debe cumplir que $\beta_s \gg \beta_c$, ya que de lo contrario serían los vecinos, y no el elemento ganador, los que se moverían más rápido hacia el vector de entrada y la topología resultante no reflejaría correctamente la distribución del espacio de entrada.

### 2.3.4. Crecimiento de la estructura

El objetivo perseguido es conseguir una estructura en la que los vectores de pesos $\vec{w}_c$ se distribuyan de acuerdo a $P(\xi)$. Este punto es alcanzado cuando cada neurona tiene la misma probabilidad de ser la ganadora para el vector de entrada actual. La probabilidad de distribución no se conoce explícitamente, pero con los contadores locales antes mencionados, es posible calcular una estimación de $P(\xi)$, denominada *frecuencia relativa de activación* de la neurona ($h_c$), reflejada en la siguiente ecuación:

$$h_c = \frac{\tau_c}{\sum_{i=1}^{N_A} \tau_i} \tag{2.5}$$

donde $N_A$ es el número de elementos de la estructura $A$.

Después de un cierto tiempo, el valor de la frecuencia relativa de activación debería ser igual para todas las neuronas. Un alto valor de $h_c$ señala una buena posición para insertar una nueva neurona, ya que lo que se busca es reducir ese valor hasta un cierto nivel uniforme. Siempre después de un número fijo de pasos de adaptación, $\lambda$, se evalúa la posibilidad de crecimiento de la estructura. Los pasos que se siguen son:

Paso 1. Se busca la neurona $q$ con la **mayor frecuencia relativa de activación**, es decir, que cumpla que:

$$h_q \geq h_c \quad \forall c \in A$$

Paso 2. Se busca el **vecino directo** de $q$ que con la **mayor distancia** al vector de entrada. Esta será la neurona $f$ que cumpla:

$$\|\vec{w}_f - \vec{w}_q\| \geq \|\vec{w}_c - \vec{w}_q\| \quad \forall c \in N_q$$

donde $N_q$ son los vecinos directos de la neurona $q$.

Paso 3. Se **inserta una nueva neurona** $r$ entre $q$ y $f$. Este nuevo elemento se conecta a las demás neuronas de la estructura de modo que estas nuevas conexiones mantengan una estructura consistente $k$-dimensional. El vector de pesos inicial para la neurona $r$ la situará a medio camino entre $q$ y $f$, es decir:

$$\vec{w}_r = \frac{1}{2}(\vec{w}_q + \vec{w}_f)$$

Paso 4. La inserción de $r$ dará lugar a una nueva región de Voronoi $F_r$ en el espacio de entradas. Al mismo tiempo las regiones de Voronoi de los vecinos topológicos de $r$ se verán reducidas. Este cambio se refleja

de acuerdo a una redistribución de las variables contador $\tau_c$ que se calcularán siguiendo la siguiente ecuación:

$$\Delta\tau_c = \frac{|F_c^{\text{nuevo}}| - |F_c^{\text{viejo}}|}{|F_c^{\text{viejo}}|}\tau_c \quad \forall c \in N_r \tag{2.6}$$

donde $N_r$ es el conjunto de todos los vecinos directos de $r$ y $|F_c|$ es el volumen $n$-dimensional de $F_c$.

Paso 5. El valor inicial del contador de la nueva neurona será:

$$\tau_r = -\sum_{c \in N_r} \Delta\tau_c$$

Esto se puede interpretar como asignarle al nuevo elemento tantas señales de entrada como podría haber tenido en el caso de haber existido desde el momento de la construcción de la red. Este mismo concepto se aplica a la reducción de los contadores de sus células vecinas.

Una representación gráfica de este proceso se puede observar en las Figura 2.2)

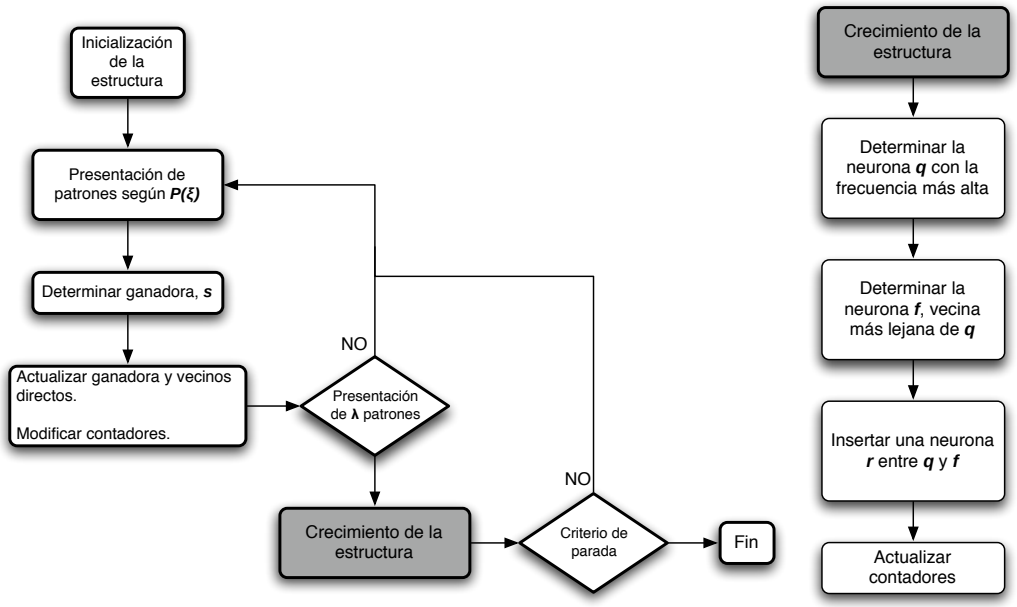


Figura 2.2: Algoritmo de aprendizaje y modificación de las estructura de una red GCS.

### 2.3.5. Características

La principal característica del modelo propuesto es que para cada $\lambda$ pasos de adaptación se produce una inserción.

También es conveniente hacer notar que el único cambio se que produce en la estructura a lo largo del tiempo consiste en la inserción de neuronas.

- Cada paso de adaptación incrementa el contador de la unidad ganadora incrementando la probabilidad de que se inserte una neurona cerca de ella.

- La inserción de una neurona $r$, cerca de otra $c$, decrementa el tamaño de la región de Voronoi $F_c$ y el valor del contador $\tau_c$. La reducción de su región de Voronoi hace que sea menos probable que $c$ sea ganadora en una próxima ocasión.

Dentro de los parámetros manejados por este tipos de redes, las GCS, es importante resaltar la influencia de los parámetros $\beta_s$ y $\beta_c$. Estos determinan la velocidad de movimiento de los elementos de procesado. Valores altos implicarán un movimiento muy rápido que podría llevar a un funcionamiento incorrecto de la red, valores bajos provocarán movimientos muy lentos que tendrán como consecuencia la necesidad de un mayor número de pasos de entrenamiento para adaptarse con precisión a la función de distribución buscada.

## 2.4. Gas neuronal creciente (GNG)[2]

Los mapas autoorganizativos de Kohonen o las redes GCS permiten la proyección en subespacios de muestras discretas no lineales de una **dimensionalidad escogida a priori**. Dependiendo de las relaciones inherentes entre la dimensionalidad de los datos y la dimensionalidad del espacio buscado, alguna información de la disposición topológica de los datos de entrada se pierde en el proceso. Se puede decir que, generalmente, no existe un mapeo reversible de una dimensionalidad alta a una baja. La búsqueda de estructuras que permitiesen estos mapeos reversibles, permitió alcanzar un nuevo objetivo dentro del aprendizaje no supervisado, el *aprendizaje topológico*.

Dada una distribución de datos de alta dimensionalidad, $P(\xi)$, se tiene que encontrar una estructura topológica que refleje lo más fielmente posible la distribución topológica de los datos. Una de las formas más elegantes de construir este tipo de estructuras es mediante *aprendizaje competitivo de*

[2] *Growing Neural Gas.*

*la regla de Hebb*(CHL). Este aprendizaje requiere el uso de un método de quantización de vectores.

A este respecto se propuso el método *neural-gas* o *gas neuronal*. Basándose en éste se presentó un nuevo método llamado *Gas Neuronal Creciente*, que tiene las propiedades de crecer y de que los parámetros introducidos son constantes en el aprendizaje.

El modelo **Gas Neuronal Creciente** es una estructura autoorganizativa, con aprendizaje no supervisado, en donde los fundamentos para preservar la topología se basan en la obtención de la llamada *triangulación inducida de Delaunay* (TID). Se introduce los conceptos de:

- **Error local** para cada elemento.
- **Edad** para cada conexión.

Este modelo surgió principalmente para mejorar algunas de las limitaciones del modelo básico de Kohonen. Mientras un SOM[3] necesita la definición de una topología fija en el momento de su creación, una GNG inicia su entrenamiento con apenas dos neuronas y se van añadiendo nuevas unidades gradualmente para mejorar el desarrollo de la red.

Otra diferencia respecto al modelo de Kohonen es la forma en que se conectan las neuronas. En los mapas de Kohonen, las conexiones son laterales formando una cruz en cada unidad. En el modelo GNG, una unidad puede tener mucho más de cuatro vecinos, generando diversas figuras geométricas. Se trata por tanto de una red con mayor capacidad de aprendizaje.

La inserción de aristas entre las unidades más cercanas al patrón de entrada genera una conexión simple de TID. La eliminación de aristas es necesaria para deshacerse de aquellas que ya no formarán parte de la TID. Este proceso es conocido como *envejecimiento de aristas*.

### 2.4.1. Algoritmo de las GNG

Se tendrán en cuenta las siguientes consideraciones:

- Sea un conjunto $A$ de unidades o neuronas. Cada unidad $c \in A$ tiene asociado un vector de referencia $\vec{w}_c \in \mathbb{R}^n$. Los vectores de referencia se pueden ver como las posiciones en el espacio de entrada de las correspondientes unidades.
- Sea un conjunto $N$ de conexiones (o arcos) entre pares de unidades. Estas conexiones no tienen asociado un peso. Su único objetivo es la definición de la estructura topológica.

[3]Mapa Autoorganizativo, en ingles *Self-Organizative Map*

La idea principal del método es añadir sucesivamente nuevas neuronas a una red inicialmente reducida, evaluando medidas estadísticas locales realizadas durante pasos previos de adaptación. Es la misma aproximación que se realizó en el caso de las redes GCS, sin embargo esta tenían una topología con dimensionalidad fija.

En la aproximación que explicamos a continuación, la topología de la red es generada incrementalmente por el modelo CHL, y tiene una dimensionalidad que dependerá de los datos de entrada y que además puede variar localmente. El algoritmo propuesto es el siguiente:

Paso 1. Se empieza con dos unidades $a$ y $b$ en posiciones aleatorias $\vec{w}_a$ y $\vec{w}_b$ en $\mathbb{R}^n$. También es necesario inicializar el conjunto de conexiones (a un conjunto vacío) $C$, $C \subset A \times A$ a $\varnothing$.

Paso 2. Se obtiene una señal de entrada $\xi$ acorde con $P(\xi)$.

Paso 3. Se determina la unidad más próxima, $s$, unidad ganadora, y la segunda más próxima, $p$, de acuerdo con las siguientes ecuaciones:

$$s = min\|\xi - \vec{w}_c\| \quad \forall c \in A \tag{2.7}$$
$$p = min\|\xi - \vec{w}_c\| \quad \forall c \in A - \{s\} \tag{2.8}$$

donde $\|\xi - w_c\|$ representa la distancia entre los vectores $\xi$ y $\vec{w}_c$ (en este caso **la distancia euclídea**, para variar).

Paso 4. Si no existe una conexión entre $s$ y $p$, se crea y se establece la edad de la nueva conexión a cero:

$$C = C \cup \{s, p\}$$
$$edad_{(s,p)} = 0$$

Paso 5. Calcular el *error local* para la neurona ganadora, como la suma del cuadrado de la distancia euclídea entre la neurona ganadora y el patrón presentado:

$$\Delta E_s = \|\xi - \vec{w}_s\|^2$$

Paso 6. Se actualiza el vector de pesos asociado a la célula ganadora y los asociados a sus vecinas. Siendo $\mu_s$ y $\mu_n$ las tasas de aprendizaje para la neurona ganadora y sus vecinas respectivamente tendremos:

$$\Delta\vec{w}_s = \mu_s(\xi - \vec{w}_s) \quad (2.9)$$
$$\Delta\vec{w}_c = \mu_n(\xi - \vec{w}_c) \quad \forall c \in N_s \quad (2.10)$$

donde $N_s$ representa al conjunto de los vecinos topológicos directos de $s$.

Paso 7. Incrementar la edad de todas las conexiones de $s$ según la regla:

$$edad_{(s,c)} = edad_{(s,c)} + 1 \quad \forall c \in N_s \quad (2.11)$$

donde $N_s$ es el conjunto de neuronas vecinas de $s$.

Paso 8. Eliminar las conexiones cuya edad supere el valor $a_{\mathsf{max}}$. Si tras este proceso aparecen elementos aislados, también deben ser eliminados de la estructura.

Paso 9. Si el número de señales de entrada suministradas es múltiplo de $\delta$, se inserta una nueva unidad de acuerdo a los siguientes pasos:

Paso 9.1. Determinar la uinidad $q$ que tenga el máximo error acumulado.

Paso 9.2. Determinar, de entre los vecinos de $q$, la unidad $f$ con mayor error acumulado.

Paso 9.3. Insertar una nueva neurona $r$, a medio camino entre $q$ y $f$. Su vector de pesos se inicializará a:

$$\vec{w}_r = \frac{1}{2}(\vec{w}_q + \vec{w}_f)$$

Paso 9.4. Establecer las conexiones entre la nueva unidad $r$ y las unidades $q$ y $f$, y eliminar el arco original entre $q$ y $f$.

Paso 9.5. Decrfementar las variables de error asociadas a $q$ y $f$ multiplicándolas por un término constante $\alpha$.

$$\Delta E_q = -\alpha E_q$$
$$\Delta E_f = -\alpha E_f$$

Paso 9.6. La variable de error para la unidad $r$ será:

$$E_r = \frac{1}{2}(E_q + E_f)$$

Paso 10. Disminuir la variable de error de todas las unidades:

$$\Delta E_c = -\beta E_c \quad \forall c \in A$$

Paso 11. Si se cumple el criterio de parada (por ejemplo, el tamaño de la red) se finaliza, si no se vuelve al paso 2.

Una representación gráfica de este algoritmo se puede observar en la Figura 2.3.

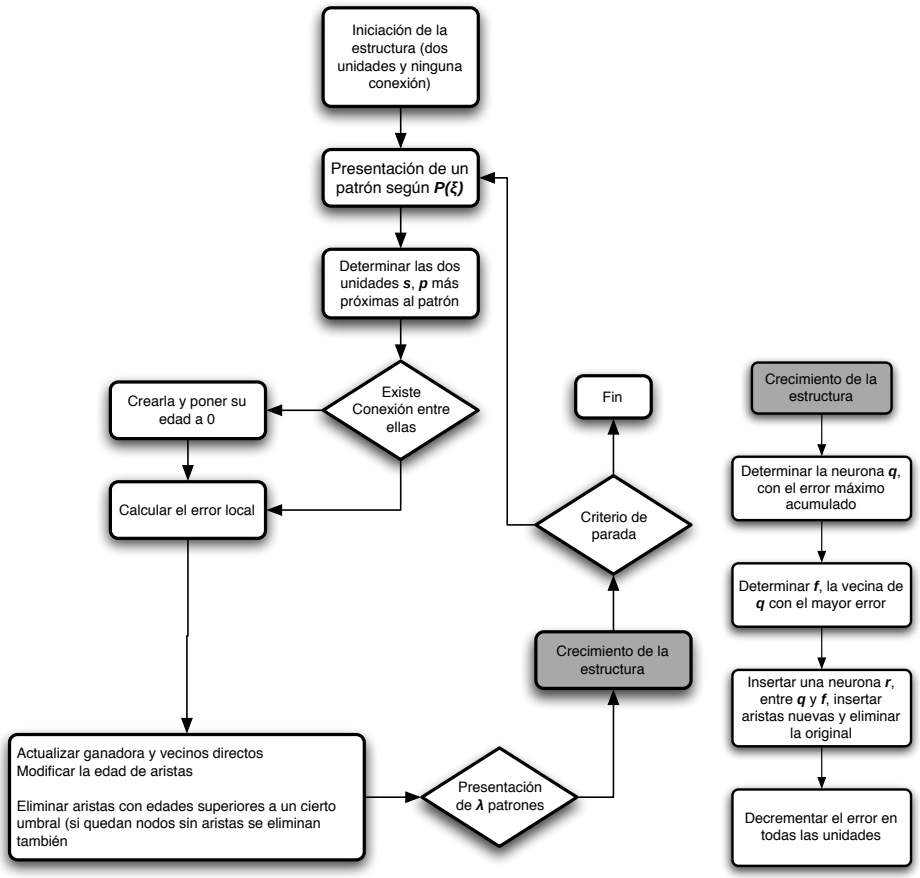


Figura 2.3: Modificación y crecimiento de la estructura de una red GNG.

### 2.4.2. Funcionamiento del método

A continuación explicaremos el resultado de aplicar el algoritmo anterior y las consecuencias derivadas de cada uno de sus pasos.

El actualizar el error local de la unidad ganadora es una forma de detectar nodos que cubren grandes regiones del espacio de entrada. En estos nodos, su error local crecerá mucho más rápido que en otros nodos que cubran menores regiones, estadísticamente hablando. Como lo que buscamos es minimizar el error, será en torno a estos nodos donde insertaremos nuevas neuronas.

Cuando decrementamos la variable de error de todas las unidades (penúltimo paso del algoritmo), buscamos aumentar la influencia de los errores recientes y evitar así el crecimiento incontrolado de los errores locales.

Las ecuaciones (eq:GNG.cambio.pesos.ganadora) y (2.10) determinan el movimiento de los nodos de la red, mediante la adaptación de los centros (vectores de pesos). Tanto la célula ganadora como sus vecinas, estas en menor medida, acercan su posición a la del patrón que la ha hecho ganar. El objetivo es permitir un movimiento general de todas las unidades hacia las áreas de donde proceden las señales de entrada, es decir aquéllas donde $P(\xi) > 0$.

La inserción de conexiones entre la neurona más próxima y la siguiente más próxima a la señal de entrada, da lugar a una conexión simple denominada *triangulación inducida de Delaunay*(TID) respecto a la posición actual de todas las unidades. La eliminación de arcos, es necesaria para librarse de aquellas conexiones que no volverán a formar parte de la triangulación inducida de Delaunay, ya que sus puntos finales se han movido. Esto se logra mediante el uso de la variable de *edad local* de las conexiones con las unidades más próximas.

Con las inserciones y borrado de arcos, el modelo trata de construir y encontrar la triangulación inducida de Delaunay meidante movimientos pequeños a través de la adaptación de los vectores de referencia.

La acumulación de las distancias (paso 5), durante la adaptación, ayuda a identificar las unidades que se encuentran en áreas del espacio de entrada donde el mapeo de señales tiene un alto índice de error. Para reducir éste se introducirán nuevas neuronas en estas regiones.