Capítulo II . El Perceptrón Multicapa.

2.1. El Perceptrón Multicapa.

En 1958 Rosenblatt publica sus primeros trabajos acerca de su modelo neuronal denominado *perceptrón* junto a su algoritmo de convergencia. Este modelo, como veremos, está formado por tres elementos o niveles de organización: las unidades sensoriales, las unidades asociativas y las unidades de respuesta. Las conexiones entre las dos primeras se definen en forma de pesos fijos y entre las dos últimas mediante pesos variables. Las unidades asociativas actúan como pequeños preprocesadores diseñados para extraer información a partir de los ejemplos presentados.

En la Fig. 2.1. se ilustra un perceptrón monocapa -o simplemente perceptrón- con una única neurona[1], estando por ello limitado su uso a la clasificación de patrones en dos clases o categorías. Si expandimos esta capa de salida incluyendo más de una neurona en ella podremos clasificar correctamente más de dos clases con la limitación demostrada por Minsky y Papert (1969) de que estas clases deben ser separables linealmente.

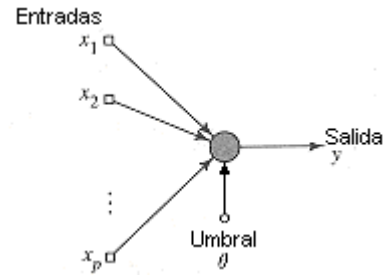

Figura 2.1. Perceptrón de Rosenblatt.

2.1.1. Modelos de neurona. Nomenclatura.

Una neurona es una unidad de proceso de la información, constituyente fundamental de las redes neuronales que puede seguir varios modelos matemáticos, pero que deberá contar esencialmente con las siguientes partes:

1. Una colección de *sinapsis* o *conexiones pesadas.*
Una señal x_j a la entrada de la sinapsis *j* conectada a la neurona *k* será multiplicada por el peso w_{kj}. Este peso será positivo si la sinapsis asociada es *excitadora,* y negativo si la sinapsis es *inhibitoria.*
2. Un *sumador* para las señales de entrada pesadas.

[1] A partir de ahora utilizaremos indistintamente los términos *neurona, unidad* y *nodo.*

3. Una *función de activación.*

Su función es la de limitar la amplitud de la salida de la neurona.

4. Una entrada fija que multiplica a un peso adicional llamado *umbral* q_k y que tiene el efecto de rebajar la entrada de la red al rango de la función de activación[2].

En la Fig. 2.2 vemos el modelo típico de una neurona propuesto por McCulloch-Pitts y formado por los bloques descritos arriba.

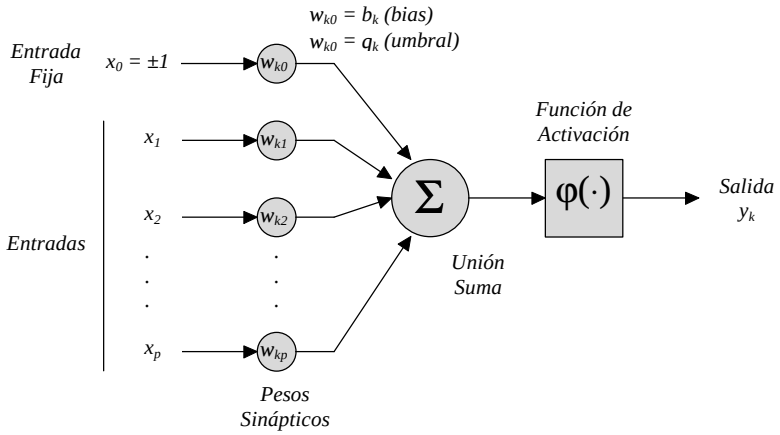

Figura 2.2. Modelo no lineal de una neurona.

A partir de las definiciones anteriores podemos describir matemáticamente una neurona *k* como sigue:

$$u_k = \sum_{j=1}^{p} w_{kj} x_j \tag{2.1}$$

$$v_k = j\,(u_k - q_k) \tag{2.2}$$

Redefiniendo $v_k = u_k - q_k$, obtenemos $y_k = j\,(v_k)$ y si, además, añadimos una nueva sinapsis, cuya entrada sea de valor fijo ± 1 y cuyo peso sea $w_{k0} = q_k$, obtendremos los dos modelos de neurona no lineal representados en la figura anterior.

■ Funciones de activación [2].

Se denotan por $j\,(v)$ y tienen las siguientes variantes.

1. Función Umbral o Escalón. (Fig. 2.3a)

$$j\,(v) = \begin{cases} 1 & \text{si } v \geq 0 \\ 0 & \text{si } v < 0 \end{cases} \tag{2.3}$$

[2] Dependiendo de que el valor de la entrada fija sea +1 o -1 recibirá los nombres de *bias* o *umbral* respectivamente.

2. Función rampa lineal. (Fig. 2.3b)

$$j(v) = \begin{cases} 1 & si\ v \geq \frac{1}{2} \\ v & si\ \frac{1}{2} > v \geq -\frac{1}{2} \\ 0 & si\ v \leq -\frac{1}{2} \end{cases} \tag{2.4}$$

3. Función Sigmoide o Tangente hiperbólica. (Fig. 2.3c)

$$j(a \cdot v) = \frac{1}{1+\exp(-a \cdot v)} \quad \text{o} \quad j(v) = tanh(a \cdot v / 2) = \frac{1-\exp(-a \cdot v)}{1+\exp(-a \cdot v)} \tag{2.5}$$

En la Fig. 2.3c observamos diferentes funciones sigmoide superpuestas, indicando una dependencia con la pendiente *a* de las ecuaciones anteriores: cuanto mayor sea esta más abrupta será la función de activación y más se asemejará a la función de activación escalón (Fig. 2.3a).

Aunque las funciones de activación sigmoidales son las más empleadas, las posibilidades son más amplias. Por ejemplo, la posibilidad de descomponer cualquier función en series de Fourier lleva a la conclusión que podría emplearse una función de activación sinusoidal [1].

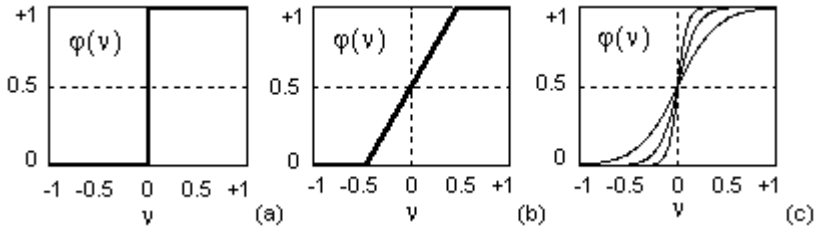

Figura 2.3. Funciones de Activación. (a) Función Umbral. (b) Función lineal a trozos. (c) Función Sigmoide o Tangente hiperbólica.

■ Preliminares. Nomenclatura.

Consideremos una RNA totalmente conectada (*fully-connected*) como en la Fig. 2.4, en la que una neurona en una capa de la red está conectada a todas las neuronas de la capa anterior. Esta es la estructura típica de un *perceptrón multicapa* donde identificamos dos tipos de señales (Fig. 2.5):

1. *Señal de Función.* Es la señal que se propaga de la entrada (izquierda en la figura) hacia la capa de salida (derecha).

2. *Señal de Error*. Son las generadas por las neuronas de salida y que se retropropagarán en forma de ajuste de las conexiones sinápticas hacia la entrada con el fin de ajustar la salida obtenida lo más fielmente a la salida deseada.

Cada neurona oculta o de salida de un perceptrón multicapa está pues diseñada para realizar dos cálculos: la propagación de la señal hacia adelante *(forward pass)* y el cálculo del vector gradiente necesario para la retro-propagación *(backward pass).*

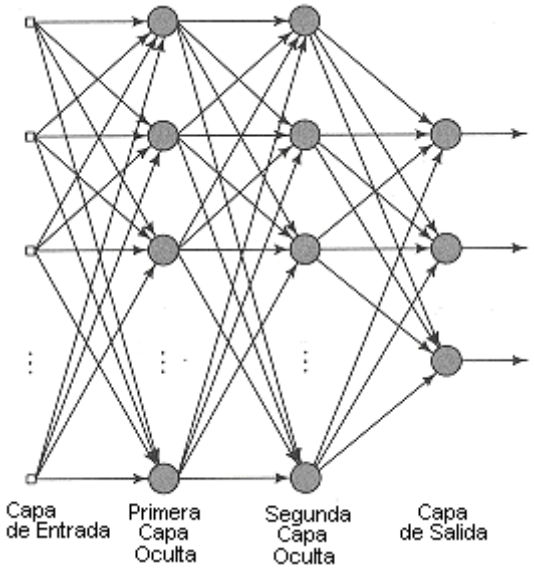

Figura 2.4. Perceptrón Multicapa con dos capas ocultas.

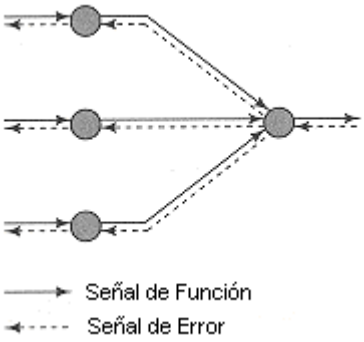

Figura 2.5. Sentidos de propagación de las señales en un perceptrón multicapa.

La nomenclatura utilizada [2] en la descripción de las redes, y según las Fig. 2.6 donde se muestra la conexión entre dos neuronas *j* y *k,* es la siguiente:

1. Los índices *i, j, k* denotan las neuronas ubicadas en las capas *i, j, k* respectivamente en el sentido de la propagación.

2. La iteración *n* se refiere al *n-ésimo* patrón presentado.

3. El símbolo *E(n)* se refiere a la suma cuadrática de los errores en la iteración *n-ésima*. La media sobre todas las muestras será E_{av}.

4. El símbolo $e_j(n)$ se refiere al error a la salida de la neurona *j*.

5. El símbolo $d_j(n)$ se refiere a la respuesta deseada del nodo *j*.

6. El símbolo $y_j(n)$ se refiere a la señal de salida de la neurona *j*.

7. El símbolo $w_{ji}(n)$ se refiere al peso de la conexión sináptica entre las neuronas *j* e *i* en la iteración *n.* La corrección aplicada a ese peso se denota por $\Delta w_{ji}(n)$.

8. El símbolo $x_i(n)$ se refiere al elemento *i-ésimo* del vector muestra.

9. El elemento *k-ésimo* de la salida total de la red se denota por $o_k(n)$.

10. El parámetro de aprendizaje será η y el momento α.

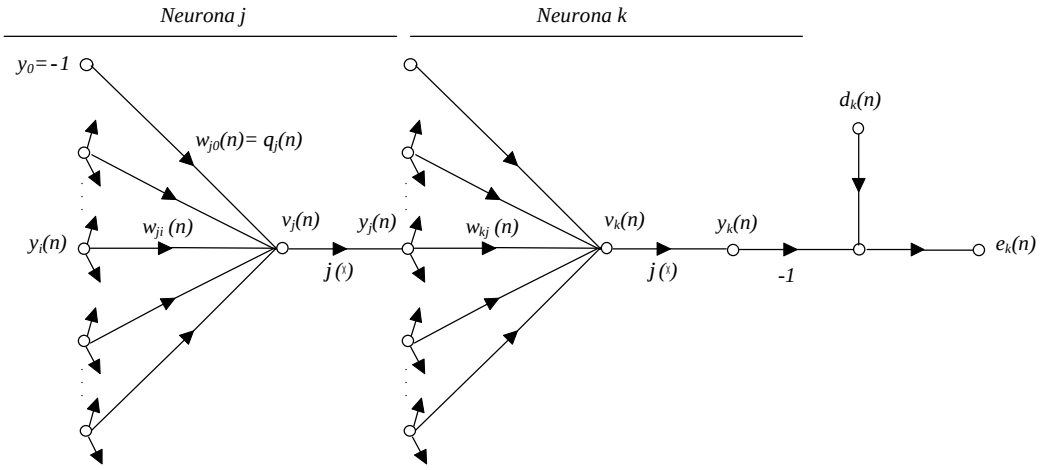

Figura 2.6. Gráfico de señales en una neurona de salida *k* conectada a una neurona oculta *j.*

2.1.2. Algoritmo de back-propagation.

La idea central de esta regla o algoritmo se encuentra en calcular los errores para las unidades de las capas ocultas a partir de los errores de las unidades de la capa de salida siendo propagados capa tras capa hacia la entrada.

La señal de error en la neurona de salida *j* en la iteración *n* (presentación del *n-ésimo* patrón de entrenamiento) viene definida así:

$$e_j(n) = d_j(n) - y_j(n) \tag{2.6}$$

donde *j* es un nodo de salida. Además, como vimos en el Cap. 1.2.2., Aptdo. Reglas Básicas de Aprendizaje (1), la suma de los errores cuadráticos de la red se puede definir como:

$$E(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n) \tag{2.7}$$

donde el conjunto *C* incluye todos los nodos en la capa de salida de la red.

La Fig. 2.7 muestra la neurona *j* alimentada por un conjunto de señales procedentes de una capa de neuronas a su izquierda.

El número de patrones o ejemplos forman el conjunto de entrenamiento o *Training set* y lo denotamos con *N*. Así, el Error Cuadrático Medio (E_{av}) es obtenido sumando *E(n)* para todas las iteraciones y normalizando respecto *N*:

$$E_{av} = \frac{1}{N}\sum_{n=1}^{N} E(n) \tag{2.8}$$

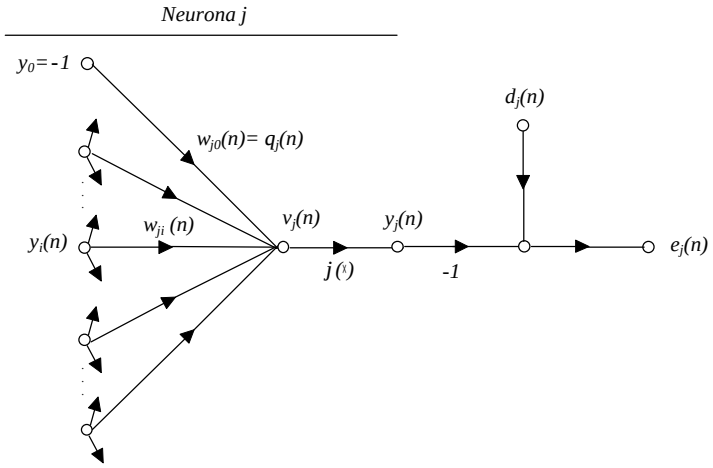

Figura 2.7. Gráfico de señales en una neurona de salida.

Para un determinado conjunto de entrenamiento, E_{av} representa una medida del aprendizaje de la RNA denominada *Función Coste*.

El objetivo de este proceso de aprendizaje es minimizar esta función de error ajustando los parámetros libres que tenemos (pesos, constante de adaptación, ...).

■ Neurona j como nodo de salida de la red.

El nivel de activación $v_j(n)$ producido en la entrada de la no linealidad asociada con la neurona *j* es, por tanto:

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n) y_i(n) \tag{2.9}$$

donde *p* es el número total de entradas (excluyendo la umbral) aplicadas al nodo *j*. Por ello, la señal que nos aparece a la salida de la neurona *j* en la iteración *n* será:

$$y_j = j_j(v_j(n)) \tag{2.10}$$

Aplicando la regla de la cadena, podremos expresar el gradiente de la suma cuadrática de los errores respecto de los pesos asociados de esta forma:

$$\frac{\P E(N)}{\P w_{ji}(n)} = \frac{\P E(N)}{\P e_j(n)} \frac{\P e_j(n)}{\P y_j(n)} \frac{\P y_j(n)}{\P v_j(n)} \frac{\P v_j(n)}{\P w_{ji}(n)} \tag{2.11}$$

Los multiplicandos de esta última ecuación se obtienen derivando las Ecs. (2.7), (2.6), (2.10) y (2.9) respectivamente:

$$\frac{\partial E(N)}{\partial e_j(n)} = e_j(n), \quad (2.12) \qquad \frac{\partial e_j(n)}{\partial y_j(n)} = -1, \quad (2.13) \qquad \frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n)), \quad (2.14) \qquad \frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \quad (2.15)$$

Por lo que agrupando todos los términos calculados, nos queda:

$$\frac{\partial E(N)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j'(v_j(n))y_i(n) \tag{2.16}$$

La corrección $\Delta w_{ji}(n)$ aplicada al peso $w_{ji}(n)$ vendrá dado por:

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(N)}{\partial w_{ji}(n)} \tag{2.17}$$

donde η es una constante que determina la velocidad de aprendizaje, y se llama parámetro de aprendizaje del algoritmo de *back-propagation*.

De acuerdo con las Ecs. (2.16) y (2.17), obtenemos:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{2.18}$$

donde el gradiente local $\delta_j(n)$ viene dado por

$$\delta_j(n) = -\frac{\partial E(N)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n)\varphi_j'(v_j(n)) \tag{2.19}$$

Hasta aquí, hemos analizado el caso en que la neurona *j* es un nodo de salida; veamos ahora la derivación de las señales y ecuaciones de propagación y de errores en el caso en que la neurona *j* sea un nodo oculto.

■ Neurona j como nodo oculto de la red.

De acuerdo a la Ec. (2.19), podemos volver a definir el gradiente local $\delta_j(n)$ para la neurona oculta *j* de la siguiente manera:

$$\delta_j(n) = \frac{\partial E(N)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial E(N)}{\partial y_j(n)} \varphi_j'(v_j(n)) \tag{2.20}$$

Para calcular la derivada parcial $\frac{\partial E(N)}{\partial y_j(n)}$, debemos proceder considerando la Fig. 2.6, por lo que la Ec. (2.2) la podríamos reescribir así

$$E(n) = \frac{1}{2} \sum_{k \in C} e_k^2(n) \tag{2.21}$$

donde el índice *k* indica que la neurona *k* es un nodo de salida.

Ahora bien, al diferenciar la Ec. (2.21), obtenemos

$$\frac{\P E(N)}{\P y_j(n)} = \sum_{k \in C} e_k(n) \frac{\P e_k(n)}{\P y_j(n)} = \sum_{k \in C} e_k(n) \frac{\P e_k(n)}{\P v_k(n)} \frac{\P v_k(n)}{\P y_j(n)} \tag{2.22}$$

Para resolver esta ecuación, echaremos mano de la Fig. 2.6, haciendo notar que:

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - j_k(v_k(n)) \tag{2.23}$$

donde recordemos que el índice *k* nos indica el nodo de salida.

Las derivadas parciales, tendrán como expresión

$$\frac{\P e_k(n)}{\P v_k(n)} = -j_k'(v_k(n)) \tag{2.24}$$

De la Fig. 2.6, también advertimos que el nivel de activación vendrá dado para la neurona *k* de la siguiente manera:

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n) \tag{2.25}$$

donde *q* nos indica el número total de entradas (excluyendo la umbral) aplicadas a la neurona *k.*

De la expresión anterior obtenemos la segunda ecuación necesaria

$$\frac{\P v_k(n)}{\P y_j(n)} = w_{kj}(n) \tag{2.26}$$

Por tanto, haciendo uso de las Ecs. (2.22), (2.24) y (2.26), nos acercamos a la expresión final de la derivada parcial deseada:

$$\frac{\P E(N)}{\P y_j(n)} = -\sum_{k \in C} e_k(n) j_k'(v_k(n)) w_{kj}(n) = -\sum_{k \in C} d_k(n) w_{kj}(n) \tag{2.27}$$

Finalmente, y usando las Ec. (2.27) y (2.20), obtenemos el gradiente local $d_j(n)$ para una neurona oculta j:

$$d_j(n) = j_j'(v_j(n)) \sum_{k \in C} d_k(n) w_{kj}(n) \tag{2.28}$$

2.1.3. Características del Algoritmo de Aprendizaje de back-propagation.

Analizaremos los aspectos más importantes a la hora de trabajar con este algoritmo de aprendizaje.

■ Fases de cálculo.

Como hemos comentado y calculado en el Cap. 2.1.1. Preliminares y en el Cap. 2.1.2, tenemos dos funciones a realizar por cada neurona, y por tanto, en la aplicación del Algoritmo de BP también distinguimos dos pasos:

1. Cálculo hacia adelante. Consiste en la propagación de las señales de entrada hacia la salida y en el cálculo del error cometido comparando la salida obtenida con la deseada.

2. Cálculo hacia atrás. Este paso se inicia en la capa de salida pasando las señales de error hacia la capa de entrada capa a capa, y calculando recursivamente los gradientes locales para cada neurona.

■ Derivación de las funciones de activación.

1. Función de activación sigmoidal.

$$y_j = j_j(v_j(n)) = \frac{1}{1+\exp(-v_j(n))}, \qquad -\infty < v_j(n) < +\infty \tag{2.29}$$

Diferenciando los dos lados de esta expresión, obtenemos

$$\frac{\P y_j(n)}{\P v_j(n)} = j_j'(v_j(n)) = \frac{\exp(-v_j(n))}{\left[1+\exp(-v_j(n))\right]^2} \tag{2.30}$$

Usando la Ec. (2.29) para simplificar la Ec. (2.30), obtenemos:

$$j_j'(v_j(n)) = y_j(n)\left[1-y_j(n)\right] \tag{2.31}$$

Para una neurona *j* de la capa de salida, sabemos que $y_j(n) = o_j(n)$, por lo que podemos expresar el gradiente local para la neurona *j* de la siguiente manera:

$$d_j(n) = e_j(n) j_j'(v_j(n)) = [d_j(n) - o_j(n)] o_j(n)[1-o_j(n)] \tag{2.32}$$

donde la neurona *j* es un nodo de salida, $o_j(n)$ es la señal a la salida de la neurona *j*, y $d_j(n)$ es la respuesta deseada para esa neurona.

En este momento podemos introducir el concepto de *saturación* de una neurona. En la Ec. (2.32) observamos que si el valor de salida está próximo a los valores extremos de la función de activación empleada (en la sigmoidal +1 o 0), el ajuste aplicado a los pesos sinápticos de la neurona será pequeño, incluso a pesar de que la magnitud del error asociado sea grande (Ec. (2.6)) y por tanto la red puede necesitar mucho tiempo para escapar de esa situación. Esta situación

se da porque la derivada de la función de activación en estos valores extremos se aproxima a cero (Ec. 2.31).

Por otra parte, para una neurona *j* oculta, podemos expresar el gradiente local como

$$d_j(n) = j'_j(v_j(n)) \sum_{k \in C} d_k(n) w_{kj}(n) = y_j(n)\left[1 - y_j(n)\right] \sum_{k \in C} d_k(n) w_{kj}(n) \quad (2.33)$$

2. Función de activación hiperbólica.

Se define así:

$$y_j = j_j(v_j(n)) = a \cdot tanh[b \cdot v_j(n)]$$

$$= \frac{1 - \exp(-b \cdot v_j(n))}{1 + \exp(-b \cdot v_j(n))} = \frac{2}{1 + \exp(-b \cdot v_j(n))} - a \quad (2.34)$$

donde los rangos de funcionamiento son:

$$-\infty < v_j(n) < +\infty \qquad \text{y} \qquad -a < y_j(n) < +a$$

Procediendo de igual manera que para la función de activación sigmoidal, obtenemos:

i. Para una neurona *j* de la capa de salida,

$$d_j(n) = e_j(n) j'_j(v_j(n)) = \frac{b}{2a}[d_j(n) - o_j(n)][1 - o_j^2(n)] \quad (2.35)$$

ii. Para una neurona *j* oculta,

$$d_j(n) = j'_j(v_j(n)) \sum_{k \in C} d_k(n) w_{kj}(n) = \frac{b}{2a}[1 - y_j^2(n)] \sum_{k \in C} d_k(n) w_{kj}(n) \quad (2.36)$$

■ Velocidad de aprendizaje y término momento.

Recordando la Ec. (2.18) vemos que cuanto más pequeño hagamos el parámetro de aprendizaje h más pequeños serán los cambios -o correcciones- de los pesos de la red de una iteración a la siguiente, pero por otra parte, si hacemos este parámetro demasiado grande con el fin de acelerar la convergencia hacia un error mínimo podemos provocar una inestabilidad en el funcionamiento de la red que se traduce en un comportamiento oscilante en torno al mínimo buscado ω_0 (Fig. 2.8).

Un método sencillo para evitar el peligro de esta inestabilidad consiste en modificar la *Regla Delta* de la Ec. (2.18) incluyendo un término momento α, dando lugar a la denominada *Regla Delta Generalizada:*

$$\Delta w_{ji}(n) = a \Delta w_{ji}(n-1) + h d_j(n) y_i(n) \quad (2.37)$$

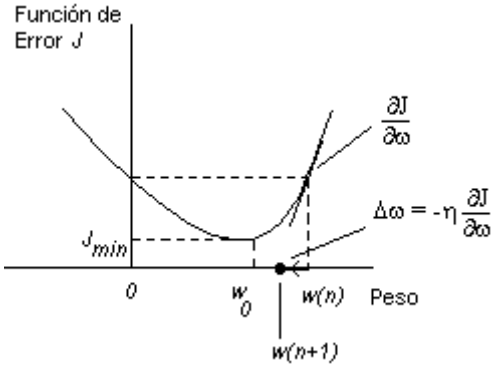

Figura 2.8. Convergencia en el espacio de los pesos: dirección del mínimo de la función coste mediante el Método de Descenso por Gradiente.

Por otra parte, para asegurar la convergencia, deberemos mantener el rango del momento estrictamente entre $0 \leq |a| < 1$.

De la Ecs. (2.37) y (2.17) se infiere que los cambios en los valores de los pesos sea proporcional a $\partial E(n)/\partial w_{ji}(n)$, con lo que podemos realizar otra observación importante respecto al signo de esta derivada. Cuando éste sea igual en dos iteraciones sucesivas, la corrección de los pesos crecerá en magnitud - recordar Ec. (2.18) -, y por tanto ese ajuste se realizará con valores elevados, contribuyendo α a acelerar la búsqueda
del mínimo. Por contra, si de una iteración a la siguiente se obtienen signos opuestos de esta derivada, la corrección de los pesos decrecerá en magnitud y por tanto la inclusión del término momento tendrá un efecto estabilizante en la búsqueda del mínimo.

Además, el momento tiene la propiedad de prevenir que el proceso de aprendizaje termine en un mínimo local en el espacio o superficie de error (Fig. 2.9).

■ Modos de entrenamientos.

Definimos *época* como el proceso de presentación de un conjunto entero de ejemplos o muestras de entrenamiento. Es recomendable la presentación de estos patrones de forma aleatoria de una época a otra.

Podemos proceder de dos formas diferentes en el entrenamiento:

1. Modo "on-line". Donde la actualización de los pesos se realiza tras la presentación de cada ejemplo de entrenamiento.

2. Modo "batch". Donde la actualización de los pesos de la red se efectúa tras la presentación de todos los ejemplos de entrenamiento que constituyen una época.

■ Inicialización de pesos.

Con el fin de evitar la saturación[3] de las neuronas los pesos suelen ser inicializados aleatoriamente dentro de un rango pequeño (valores típicos son [-0.05, +0.05] o [-0.5,+0.5]). Una elección errónea de este rango puede provocar una saturación rápida hacia los valores límite de las funciones de activación. La precaución esencial para resolver este problema está en intentar trabajar en las regiones lineales de las funciones de activación.

[3] Tal y como explicamos en el Aptdo. 2.1.3. Derivación de las funciones de Activación, con valores elevados de los pesos, se corre el riesgo de una saturación de las neuronas ya que a la salida suma pesada se dan valores grandes que derivan en valores marginales tras la aplicación de las funciones de activación (Fig. 2.2 y Fig. 2.3c).

■ Deficiencias.

El algoritmo *back-propagation* se ha convertido en el estándar para el entrenamiento de perceptrones multicapa, siendo el objeto de comparación a la hora de introducir nuevos algoritmos de aprendizaje. A pesar del aparente éxito de este algoritmo, hay aspectos que no lo hacen efectivo y robusto.

Algunos problemas se derivan de una inicialización de los pesos errónea y de la elección deficiente de los parámetros α y η. Esto incidirá en los siguientes aspectos negativos [1], [2]:

1. Paralización de la red. Se trata de la saturación de la red antes de llegar al sistema óptimo provocada por un valor excesivo de la suma pesada de las entradas. Esta saturación se explica considerando que la entrada total de una neurona oculta o de salida alcanzará grandes magnitudes, y por tanto, las funciones de activación tenderán a sus valores extremos tal y como se observa en las Ecs. (2.31) y (2.32).

2. Mínimo local. El algoritmo de *back-propagation* se basa en la búsqueda del mínimo valor en la superficie de error ajustando los pesos mediante el *método del descenso por gradiente*. Es por tanto evidente que ese ajuste se realice en la dirección inversa al gradiente de la curva o superficie de error. Ahora bien, esta superficie está formada normalmente por crestas y valles, siendo posible que el sucesivo ajuste de los pesos se esté produciendo alrededor de un mínimo local. Ésto se debe al cambio sucesivo del signo del gradiente como consecuencia de una elección errónea en la inicialización de los pesos de la red (Fig. 2.9).

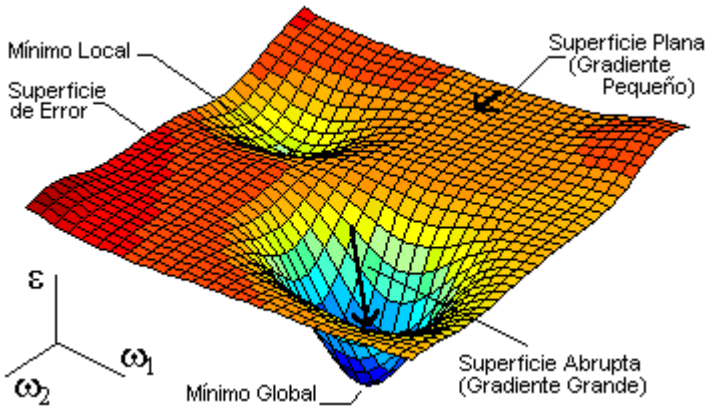

Figura 2.9. Superficie de Error típica. La búsqueda del mínimo mediante métodos de descenso por gradiente puede provocar la caída en regiones de mínimo local.