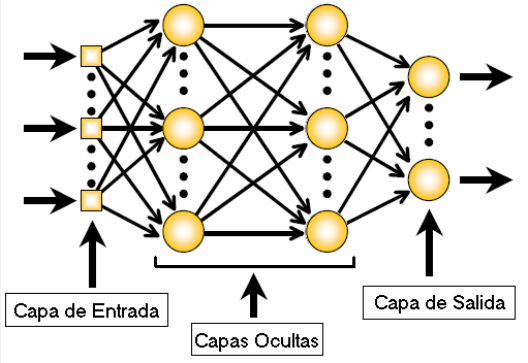
REDES PERCEPTRON MULTICAPAS

El Perceptrón Multicapa (*MLP*, *Multi Layer Perceptron*) es el exponente más típico de las redes neuronales artificiales con aprendizaje supervisado. Se basa en la unidad más simple, el perceptrón simple.



Cada perceptrón está conectado a todos los demás

de la siguiente capa. No hay conexión entre las

neuronas de la misma capa.

En estas redes podemos diferenciar tres tipos de capas, la capa de entrada, las capas intermedias u ocultas y la capa de salida. Las capas ocultas no están conectadas al exterior. El número de capas ocultas generalmente es 1 o 2.

El entrenamiento de estas redes, se basa en la presentación sucesiva y de forma reiterada, de vectores en las capas de entrada y salida (vectores entrada y su correspondiente vector salida deseada).

La red crea un modelo a base de ajustar sus pesos en función de los vectores de entrenamiento, de forma que a medida que se pasan estos patrones, para cada vector de entrada la red producirá un valor de salida más similar al vector de salida esperado.

Estas redes también se llaman de retropropagación (*backpropagation*), nombre que viene dado por el tipo de aprendizaje que utilizan.

MÉTODO DE APRENDIZAJE BACKPROPAGATION

El algoritmo de Backpropagation está basado en el descenso por el gradiente e intenta encontrar un mínimo en una cierta función de error.

Utilizando este método nos encontramos con dos tipos de señales:

- ✓ Señal de avance (*feedforward pass*): es la señal que recorre la red desde su entrada (señal de entrada), pasa por las capas intermedias y emerge en la capa de salida (señal de salida).

- ✓ Señal de error: se origina en la capa de salida y se propaga a través de la red (capa por capa) en el sentido de retroceso.

La señal de error se define como el promedio del error cuadrático:

$$E_{av} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}\sum e_k^2(n)$$

Donde:

$$e_k(n) = d_k(n) - y_k(n)$$

La primer sumatoria se realiza sobre el conjunto de entradas (número de patrones).

La segunda sumatoria se realiza sobre el conjunto de neuronas de la capa de salida.

El error es función de los parámetros libres de la red (pesos) y para un conjunto de patrones de entrada representa el parámetro de medida del proceso de aprendizaje (función de costo).

El objetivo del proceso de aprendizaje es ajustar los parámetros libres de la red con el objetivo de minimizar esta función error.

Se utiliza el método aproximado del gradiente descendente, donde se toma el gradiente instantáneo del error cuadrático.

En una primera simplificación los pesos se adaptan con el error generado por la presentación de cada patrón. Este procedimiento resulta en una estimación del cambio de pesos verdadero que debería resultar al minimizar la función de costo.

Aún cuando las neuronas intermedias no están directamente accesibles, comparten la responsabilidad del error que se genera en las neuronas de salida.

Cuando una neurona pertenece a una capa intermedia, no tiene predefinido el valor de la salida deseada para cada presentación de un patrón de entrada.

Consecuentemente la señal de error de una neurona intermedia se debe calcular recursivamente en términos de las señales de error de todas las neuronas a las que la neurona intermedia está directamente conectada.

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 ***k*** en la iteración ***n*** (presentación del ***n-ésimo*** patrón de entrenamiento) viene definida así:

$$e_k(n) = d_k(n) - y_k(n)$$

Donde ***k*** es un nodo de salida.

La suma de los errores cuadráticos de la red se puede definir como:

$$\xi(n) = \frac{1}{2}\sum e_k^2(n)$$

Donde la sumatoria se realiza sobre todos los nodos de la capa de salida de la red.

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

$$\Delta w_{kj}(n) = -\eta \frac{\partial \xi(n)}{\partial w_{kj}(n)}$$

Neurona k como nodo de salida de la red

El nivel de activación $v_k(n)$ producido en la entrada de la no linealidad asociada con la neurona ***k*** es:

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

Donde ***q*** es el número total de entradas aplicadas al nodo ***k***.

Por ello, la señal que nos aparece a la salida de la neurona ***k*** en la iteración ***n*** será:

$$y_k = \varphi_k(v_k(n))$$

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:

$$\frac{\partial \xi(n)}{\partial w_{kj}(n)} = \frac{\partial \xi(n)}{\partial e_k(n)} \frac{\partial e_k(n)}{\partial y_k(n)} \frac{\partial y_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial w_{kj}(n)}$$

Donde:

$$\frac{\partial \xi(n)}{\partial w_{kj}(n)} = e_k(n), \quad \frac{\partial e_k(n)}{\partial y_k(n)} = -1, \quad \frac{\partial y_k(n)}{\partial v_k(n)} = \varphi_k'(v_k(n)), \quad \frac{\partial v_k(n)}{\partial w_{kj}(n)} = y_j(n),$$

Reemplazando:

$$\frac{\partial \xi(n)}{\partial w_{kj}(n)} = -e_k(n)\varphi_k'(v_k(n))y_j(n)$$

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

$$\Delta w_{kj}(n) = -\eta \frac{\partial \xi(n)}{\partial w_{kj}(n)}$$

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

Luego:

$$\Delta w_{kj}(n) = \eta\, e_k(n)\varphi_k'(v_k(n))y_j(n)$$

Se define el gradiente local $\delta_k(n)$ como:

$$\delta_k(n) = -\frac{\partial \xi(n)}{\partial e_k(n)} \frac{\partial e_k(n)}{\partial y_k(n)} \frac{\partial y_k(n)}{\partial v_k(n)} = e_k(n)\varphi_k'(v_k(n))$$

Entonces:

$$\Delta w_{kj}(n) = \eta \delta_k(n) y_j(n)$$

Hasta aquí, hemos analizado el caso en que la neurona ***k*** es un nodo de salida.

Veamos ahora las 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

Podemos volver a definir el gradiente local $\delta_j(n)$ para la neurona oculta ***j*** de la siguiente manera:

$$\delta_j(n) = \frac{\partial \xi(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial \xi(n)}{\partial y_j(n)} \varphi_j'(v_j(n))$$

Como:

$$\xi(n) = \frac{1}{2} \sum e_k^2(n)$$

Entonces:

$$\frac{\partial \xi(n)}{\partial y_j(n)} = \sum e_k(n) \frac{\partial e_k(n)}{\partial y_j(n)} = \sum e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)}$$

Como:

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_k(v_k(n))$$

Las derivadas parciales, tendrán como expresión:

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n))$$

Donde:

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

q nos indica el número total de entradas aplicadas a la neurona ***k***.

Luego:

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n)$$

Finalmente, obtenemos el gradiente local $\delta_j(n)$ para una neurona oculta ***j***:

$$\frac{\partial \xi(n)}{\partial y_j(n)} = -\sum e_k(n)\varphi_k'(v_k(n)) w_{kj}(n) = -\sum \delta_k(n) w_{kj}(n)$$

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum \delta_k(n) w_{kj}(n)$$

Conclusiones

La modificación de los pesos mediante la aplicación del método de *Backpropagation* es:

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

Donde:

- **η es el parámetro de aprendizaje.**
- **$\delta_j(n)$ es el gradiente local de la neurona *j*.**
- **$y_i(n)$ son las señales de entrada de la neurona *j*.**

El cálculo de $\delta_j(n)$ depende de la ubicación de la neurona en la red (capa de salida o capa intermedia).

En la aplicación del método de BP se pueden distinguir dos procesos de cálculo:

1.-Proceso hacia adelante: los pesos se mantienen inalterables y la red implementa cálculos en cada una de las neuronas del siguiente tipo:

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

$$y_j = \varphi_j(v_j(n))$$

En el caso de una neurona de la capa de salida, se calcula el error para la correspondiente neurona, por lo tanto el proceso de cálculo hacia adelante comienza con la presentación del patrón de entrada y termina con el cálculo del error para cada neurona de la capa de salida.

2.-Proceso hacia atrás: comienza en la capa de salida propagando la señal de error hacia atrás a través de las distintas capas, para realizar el cálculo recursivo de los gradientes locales $\delta_j(n)$ para cada neurona.

Los pesos se van modificando utilizando los distintos gradientes locales.

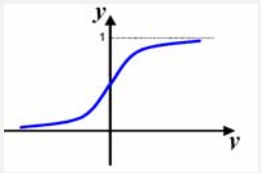
Para cada presentación de un patrón de entrenamiento, las entradas se mantienen fijas durante los dos procesos de cálculo.

FUNCIÓN DE TRANSFERENCIA

Para el cálculo de los gradientes locales $\delta_j(n)$ es necesario contar con la derivada de la función de transferencia.

En consecuencia es necesario que la función sea diferenciable (continua).

En redes Perceptrón Multicapas generalmente se utiliza una función sigmoidea no-lineal, del tipo:

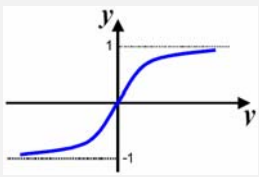


$$y_j = \varphi_j(v_j(n)) = \frac{1}{1+\exp(-v_j(n))}, \qquad -\infty < v_j(n) < +\infty$$

Donde $0 < y_j(n) < 1$.

Otra función con las mismas características es la función tangente hiperbólica:

$$y_j = \varphi_j(v_j(n)) = \frac{\exp(v_j(n)) - \exp(-v_j(n))}{\exp(v_j(n)) + \exp(-v_j(n))} \qquad -\infty < v_j(n) < +\infty$$



Donde $-1 < y_j(n) < 1$.

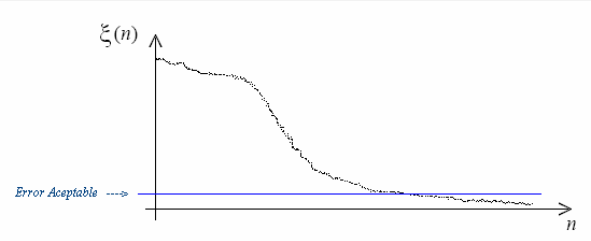
CRITERIO DE CONVERGENCIA

El método de *Backpropagation* no converge naturalmente por lo tanto es necesario establecer criterios de convergencia.

El criterio de convergencia más utilizado establece un valor mínimo del error medio cuadrático aceptable, para la aplicación con la cual se trabaja (estos valores son generalmente muy pequeños, ej.: menores del 1%).

Se define que el método converge cuando logra un error cuadrático menor al valor mínimo adoptado.

Otro criterio que puede utilizarse es definir un cierto número de iteraciones.



INICIALIZACION

El primer paso en el proceso de *Backpropagation* es la inicialización de la red, es decir, definir los valores iniciales para los parámetros libres de la red.

Una mala elección de los valores iniciales puede generar los siguientes problemas para la convergencia de la red:

Saturación prematura, el problema de saturación ocurre cuando la salida lineal de una neurona $v_j(n)$ es grande y por lo tanto la salida no-lineal de la neurona $y_j = \varphi(v_j(n))$ es cercana a ± 1.

Aún cuando el error es grande, los ajustes se producen en forma muy lenta. Esto se debe a que las variaciones en la salida lineal no se reflejan en la salida de la neurona, es decir que la derivada $\varphi'(v_j(n))$ es pequeña y en consecuencia el ajuste de los pesos se realiza de una forma muy lenta.

Para evitar la condición de saturación prematura se indican las siguientes recomendaciones prácticas:

- ✓ Valores iniciales de los parámetros libres de la red uniformemente distribuidos dentro de un rango pequeño de valores.
- ✓ La saturación es menos probable cuando el número de neuronas intermedias se puede mantener bajo, siempre dentro de un funcionamiento eficiente de la red.
- ✓ La saturación prácticamente no ocurre cuando las neuronas trabajan en su región lineal.

MÍNIMOS GLOBAL Y LOCALES

Al igual que en el algoritmo LMS, el algoritmo de Backpropagation es un procedimiento del tipo de gradiente descendiente aproximado.

Este proceso no genera problemas cuando se trata de un único elemento, ya que en este caso la superficie del error siempre tiene un único mínimo.

Sin embargo para redes multicapas, la superficie del error es mas compleja, existiendo la posibilidad de que tenga varios mínimos.

La solución óptima generalmente se relaciona con un grupo reducido de estos mínimos, correspondiendo al resto de los mínimos una solución no óptima, muchas veces inaceptable. Porque generan errores no aceptables.

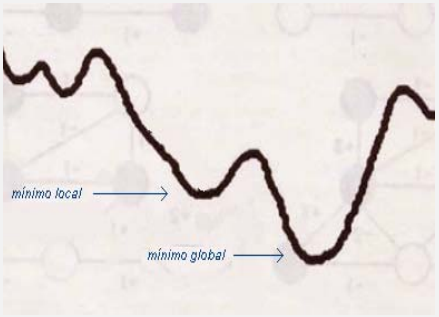
Es importante tratar de modificar las condiciones, para las cuales el sistema tenga más probabilidad de caer en uno de los mínimos locales.

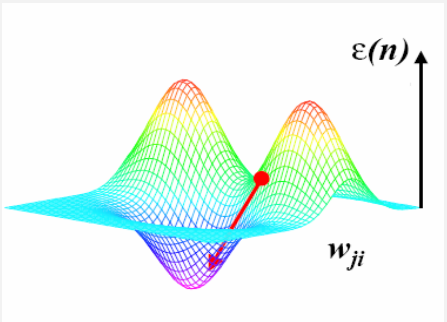
En redes con muchas neuronas intermedias la ocurrencia de un mínimo local es menos probable.

En caso que el proceso de entrenamiento caiga en un mínimo local, es posible salir de dicha situación mediante:

- ✓ El cambio de los valores iniciales de los parámetros libres de la red, puede modificar una situación de mínimo local.
- ✓Adicionar una señal de ruido a los pesos.

En ocasiones la solución generada por un mínimo local puede ser aceptable, considerándose a la red entrenada.





MOMENTO

El algoritmo de aprendizaje provee una aproximación a la trayectoria en el espacio de los pesos, calculada por el método del gradiente descendiente.

Cuanto más pequeño se toma el parámetro de aprendizaje **η** más pequeños son los cambios de pesos entre una iteración y la otra, y más suave es trayectoria en el espacio de los pesos.

En consecuencia el proceso es más estable, pero más lento.

En cambio si se aumenta el valor del parámetro η, aumenta la velocidad de convergencia del método pero la red puede resultar inestable (ej.: oscilatoria).

Una manera de aumentar la velocidad de convergencia, sin poner en peligro la estabilidad de la red, es agregar un término a la ecuación de modificación de los pesos.

El nuevo término se denomina momento:

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n)$$

Donde α es la constante de momento.

Para determinar el efecto que el término momento tiene sobre el proceso de adaptación, se transforma la ecuación anterior en una serie temporal del tipo:

$$\Delta w_{ji}(n) = \eta \sum \alpha^{n-t} \delta_j(t) y_i(t)$$

Para que sea una serie convergente $0 \leq \alpha < 1$.

Cuando el factor de sensibilidad tiene el mismo signo durante varias iteraciones la variación de los pesos aumenta en magnitud. Por lo tanto, en estas condiciones, la incorporación del momento acelera el descenso en el espacio de pesos.

Cuando el factor de sensibilidad tiene signos opuestos en iteraciones consecutivas, la variación de los pesos disminuye, por lo tanto la incorporación del momento genera un factor estabilizante cuando las direcciones oscilan en el signo.

En una aplicación del algoritmo de BP, el entrenamiento es la consecuencia de varias presentaciones de todo el conjunto de patrones de entrada.

Una presentación completa de todo el conjunto de patrones se denomina época (*epoch*).