Redes Neuronales

Jamie Areli-Toral Barrera

Resumen

En este proyecto se presenta como se conforman las redes neuronales, en especial se analizara un algoritmo de aprendizaje supervisado para el entrenamiento de redes neuronales llamado propagación hacia atrás de errores o retropropagación (mejor conocido en inglés como backpropagation).

Palabras claves

Redes neuronales, algoritmo, multicapas, entrenamiento, retropropagación.

Metodología

Las redes de neuronas artificiales son una herramienta atractiva para solucionar problemas de clasificación como el reconocimiento de caracteres manuscritos, el reconocimiento de palabras habladas, y el diagnóstico de diferentes enfermedades.

Entre las más utilizadas se encuentra el algoritmo de retropropagación debido a la naturaleza de su proceso de aprendizaje, que solamente necesita de dos ecuaciones para propagar las señales de error hacia atrás, para obtener esas ecuaciones se utilizaran tres técnicas importantes en matemáticas que es la función de error, gradiente descendiente y la regla de la cadena.

Originalidad

Se desarrollara un documento nuevo explicando detalladamente el algoritmo de retropropagación.

Introducción

Las redes neuronales artificiales (RNA) son modelos matemáticos que intentan reproducir el funcionamiento del sistema nervioso, constituidos por un conjunto de unidades llamadas neuronas o nodos conectados unos con otros.

El primer modelo de red neuronal fue propuesto por McCulloch y Pitts (1943) en términos de un modelo computacional de actividad nerviosa. Este modelo era un modelo binario, donde cada neurona tenía un escalón o umbral prefijado, y sirvió de base para los modelos posteriores.

Las redes neuronales permiten obtener un modelo no explícito que relaciona un conjunto de variables de salida con un conjunto de variables de entrada. Así, estos modelos permiten predecir cuál es el valor de salida, dados unos valores de entrada del modelo. Para estimar el modelo es necesario disponer de un conjunto de observaciones de las variables. Estas observaciones son usadas como patrones de entrenamiento para que la red aprenda y sea capaz de predecir una salida del modelo, ante nuevas observaciones. Por tanto, las

capacidades de la red van a depender en gran medida de esta fase de entrenamiento. En la fase de entrenamiento es necesario controlar muchos parámetros y distintos algoritmos de optimización.

1. Modelo biológico de una neurona

La neurona es la célula fundamental y básica del sistema nervioso especializada en conducir impulsos nerviosos.
Las neuronas tienen características propias que le permiten comunicarse entre ellas, lo que las diferencia del resto de las células biológicas.
Se estima que el cerebro humano contiene más de cien mil millones de neuronas cada una con un promedio de 7.000 conexiones sinápticas con otras neuronas.

En las neuronas se pueden distinguir tres partes fundamentales: Dentrinas, soma o cuerpo celular y axón.

Las dendritas actúan como un canal de entrada de señales provenientes desde el exterior hacia el soma de la neurona, mientras que el axón actúa como un canal de salida.
La figura 1 muestra una neurona y las distintas partes que la componen.

El espacio entre dos neuronas vecinas se denomina sinapsis. Su funcionamiento es el siguiente, en el soma de las neuronas transmisoras o presinápticas genera un pulso eléctrico llamado potencial de acción. El pulso eléctrico se propaga a través del axón en dirección a las sinapsis, que es la zona de contacto entre otras neuronas (u otro tipo de células, como las receptoras). La sinapsis recoge información electro-química procedente de las células adyacentes que están conectadas a la neurona en cuestión. Esta información llega al núcleo que se encuentra dentro del soma de la neurona, a través de las dendritas, que la procesa hasta generar una respuesta, la cual es posteriormente propagada por el axón.
La sinapsis está compuesta de un espacio líquido donde existe una cierta concentración de iones. Este espacio tiene determinadas características eléctricas que permiten inhibir o potenciar la señal eléctrica a conveniencia.

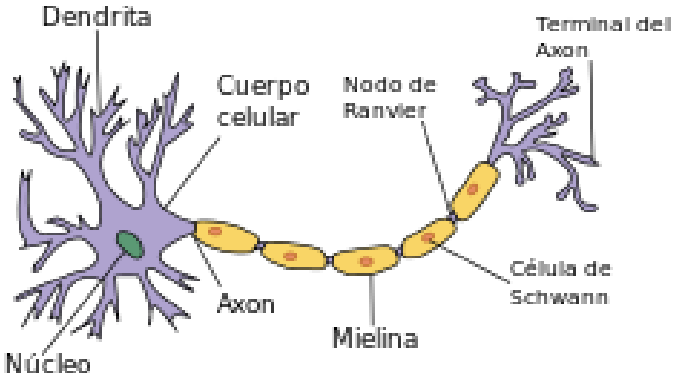


Figura 1.- Modelo fisiológico de una neurona.

2. Modelo de una red neuronal

Las redes neuronales son modelos matemáticos que intentan reproducir el comportamiento del cerebro humano. El principal objetivo de este modelo es la construcción de sistemas capaces de presentar un cierto comportamiento inteligente. Esto implica la capacidad de aprender a realizar una determinada tarea.

Una red neuronal está compuesta de tres partes: Entrada, núcleo y salidas.

La figura 2 muestra un esquema de una red neuronal artificial.

Las entradas reciben los datos o parámetros que le permiten decidir a la neurona se estará activa o no, normalmente se presentan como $x_1, x_2, \dots, x_n$.
Entre la entrada y el núcleo se tienen los pesos ($w_1, w_2, \dots, w_n$), que representan la memoria de la red.
En el núcleo se realizan todas las operaciones necesarias para determinar la salida de la neurona; el proceso que se realiza en el núcleo varía dependiendo de la red neuronal que se esté trabajando.
Las salidas devuelven la respuesta de la neurona, es decir se está activa o no, representadas comúnmente como $y_1, y_2, \dots, y_n$.

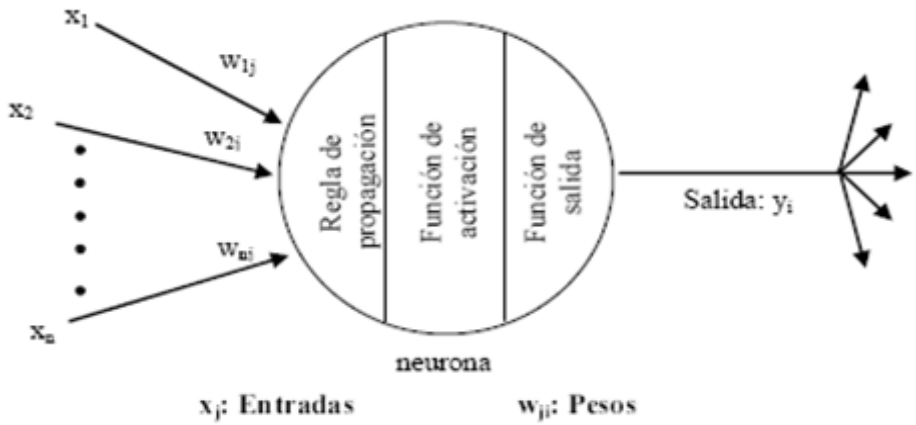


Figura 2.- Modelo de una red neuronal artificial

Veamos que en el núcleo se realizan 3 tipos de operaciones para determinar la salida de la neurona que son: Regla de propagación, Función de activación y Función de salida.

La regla de propagación, integra la información proveniente de las distintas neuronas artificiales y proporciona el valor del potencial postsináptico de la neurona i.
La función de activación, provee el estado de activación actual de la neurona i.
La función de salida, representa la salida actual de la neurona i.

Estudiando más a fondo los puntos anteriores tenemos que:

Entrada y salida

Las entradas y salidas de una neurona pueden ser clasificadas en dos grandes grupos, binarias o continuas. Las neuronas binarias, sólo admiten dos valores posibles. En general en este tipo de neurona se utilizan los siguientes dos alfabetos {0,1} o {-1,1}. Por su parte, las neuronas continuas admiten valores dentro de un determinado rango, que en general suele definirse como [-1, 1].
La selección del tipo de neurona a utilizar depende de la aplicación y del modelo a construir.

Pesos

El peso sináptico w_{ij} define la fuerza de una conexión sináptica entre dos neuronas, la neurona presináptica i y la neurona postsináptica j. Los pesos sinápticos pueden tomar valores positivos, negativos o cero. En caso de una entrada positiva, un peso positivo actúa como excitador, mientras que un peso negativo actúa como inhibidor. En caso de que el peso sea cero, no existe comunicación entre el par de neuronas.
Mediante el ajuste de los pesos sinápticos la red es capaz de adaptarse cualquier entorno y realizar una determinada tarea.

Regla de propagación

La regla de propagación determina el potencial resultante de la interacción de la neurona i con las N neuronas vecinas.

La regla de propagación más simple y utilizada consiste en realizar una suma de las entradas ponderadas con sus pesos correspondientes:

$$net_i(t) = \sum_{j=1}^{N} w_{ij} * x_j(t)$$

Función de activación

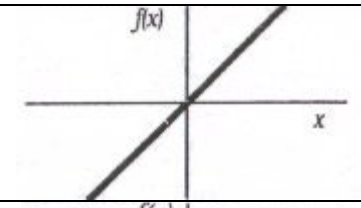
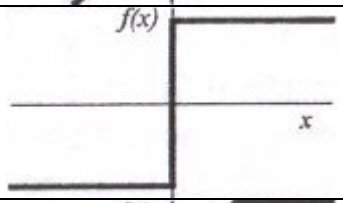
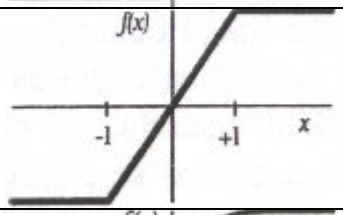
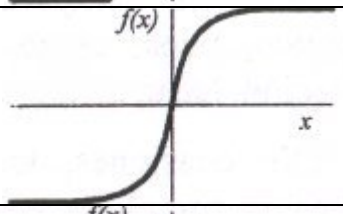
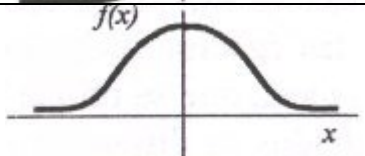
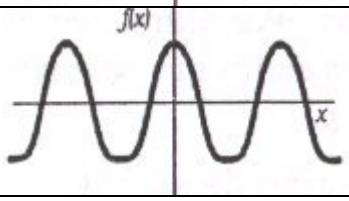
La función de activación determina el estado de activación actual de la neurona en base al potencial resultante net_i y al estado de activación anterior de la neurona $a_i(t-1)$. El estado de activación de la neurona para un determinado instante de tiempo t puede ser expresado de la siguiente manera:

$$a_i(t) = f(a_i(t-1), net_i(t))$$

Sin embargo, en la mayoría de los modelos se suele ignorar el estado anterior de la neurona, definiéndose el estado de activación en función del potencial resultante h_i:

$$a_i(t) = f(\, net_i(t))$$

Algunas de las funciones de activación más utilizadas en los distintos modelos de redes neuronales son:

	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, \infty]$	
Escalón	$y = \begin{cases} 1, & si\ x \geq 0 \\ 0, & si\ x < 0 \end{cases}$ $y = \begin{cases} 1, & si\ x \geq 0 \\ -1, & si\ x < 0 \end{cases}$	$[0, 1]$ $[-1, 1]$	
Lineal a tramos	$y = \begin{cases} 1, & si\ x > 1 \\ x, & si\ -1 \leq x \leq 1 \\ -1, & x < -1 \end{cases}$	$[-1, 1]$	
Sigmoidea	$y = \frac{1}{1 + e^{-x}}$ $y = \tanh(x)$	$[0, 1]$ $[-1, 1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0, 1]$	
Sinusoidal	$y = A\sin(wx + \varphi)$	$[-1, 1]$	

Función de salida

La función de salida proporciona el valor de salida de la neurona, en base al estado de activación de la neurona. Es decir:

$$y_i(t) = f(\, net_i(t))$$

3. Estructura de una red neuronal

Las partes principales de una red neuronal son: neuronas de entradas, ocultas, salidas y las interconexiones entre las neuronas.

La figura 3 muestra la estructura de una red neuronal artificial.

Las neuronas de entrada son transparentes, es decir no realizan ningún proceso, sólo dejan pasar la información que se quiere manejar en la red.

Las neuronas ocultas reciben las entradas y tienen la función de proporcionar un mejor aprendizaje. Las neuronas ocultas pueden o no estar presentes en una red y su incorporación depende de dos factores: Primero la topología con la que se esté trabajando y segundo de la complejidad de los patrones que deben ser aprendidos por la red.
Cuando la red está formada por una única capa de neuronas, se le llama redes monocapa, y las neuronas que conforman dicha capa cumplen la función de neuronas de entrada y salida simultáneamente. Cuando la red está compuesta por dos o más capas hablamos de redes multicapa.

La figura 4 muestra las regiones de decisión de una red neuronal de 1, 2 y 3 capas.

Las neuronas de salida se encargan de proporcionar la salida del sistema indicado, según su aprendizaje, una respuesta correcta o incorrecta.

Las interconexiones son las sinapsis de la red, estas tienen asociadas un peso sináptico, y son direccionales.
Cuando la conexión se establece entre dos neuronas de una misma capa hablamos de conexiones laterales o conexiones intra-capa. Por el contrario, si la conexión se establece entre neuronas de distintas capas se la denomina conexión inter-capa. Si la conexión se produce en el sentido inverso al de entrada-salida la conexión se llama recurrente o realimentada.

Una vez definida el tipo de neurona que se utilizará en un modelo de redes neuronales artificiales es necesario definir la topología de la misma.
La organización y disposición de las neuronas dentro de una red neuronal se denomina topología, y viene dada por el número de capas, la cantidad de neuronas por capa, el grado de conectividad, y el tipo de conexión entre neuronas.
A su vez, hablamos de redes neuronales con conexión hacia delante (redes feedforward) cuando las interconexiones entre las distintas neuronas de la red siguen un único sentido, desde la entrada de la red hacia la salida de la misma. Cuando las interconexiones pueden ser tanto hacia delante como hacia atrás hablamos de redes recurrentes (redes feedback).

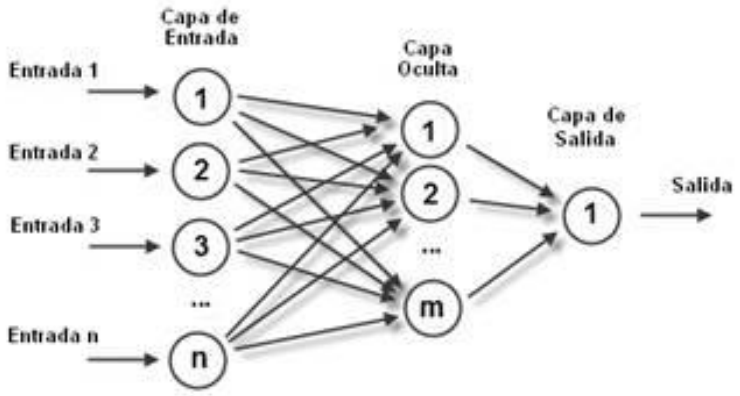


Figura 3.- Estructura de una red neuronal artificial

Estructura	Regiones de Desición	Problema de la XOR	Clases con Regiones Mezcladas	Formas de Regiones más Generales
1 Capa	Medio Plano Limitado por un Hiperplano			
2 Capas	Regiones Cerradas o Convexas			
3 Capas	Complejidad Arbitraria Limitada por el Número de Neuronas			

Figura 4.- Regiones de decisión

En el proceso de una red neuronal podemos distinguir dos fases o modos de operación: La fase de aprendizaje o entrenamiento, y la fase de ejecución o prueba.

Fase de entrenamiento

Una vez seleccionada el tipo de neurona artificial que se utilizará en una red neuronal y determinada su topología es necesario entrenarla para que la red pueda ser utilizada. Partiendo de un conjunto de pesos sinápticos aleatorio, el proceso de aprendizaje busca un conjunto de pesos que permitan a la red desarrollar correctamente una determinada tarea. Durante el proceso de aprendizaje se va refinando iterativamente la solución hasta alcanzar un nivel de operación suficientemente bueno.

El proceso de aprendizaje se puede dividir en tres grandes grupos de acuerdo a sus características (Isasi Viñuela y Galván León, 2004), (Yao, 1999):

• Aprendizaje supervisado. Se presenta a la red un conjunto de patrones de entrada junto con la salida esperada. Los pesos se van modificando de manera proporcional al error que se produce entre la salida real de la red y la salida esperada.

• Aprendizaje no supervisado. Se presenta a la red un conjunto de patrones de entrada. No hay información disponible sobre la salida esperada. El proceso de entrenamiento en este caso deberá ajustar sus pesos en base a la correlación existente entre los datos de entrada.

• Aprendizaje por refuerzo. Este tipo de aprendizaje se ubica entre los dos anteriores. Se le presenta a la red un conjunto de patrones de entrada y se le indica a la red si la salida obtenida es o no correcta. Sin embargo, no se le proporciona el valor de la salida esperada. Este tipo de aprendizaje es muy útil en aquellos casos en que se desconoce cuál es la salida exacta que debe proporcionar la red.

Fase de operación

Una vez finalizada la fase de aprendizaje, el modelo puede que se ajuste demasiado a las particularidades presentes en los patrones de entrenamiento, perdiendo su habilidad de generalizar su aprendizaje a casos nuevos.
Para evitar el problema del sobreajuste, es aconsejable utilizar un segundo grupo de datos diferentes a los de entrenamiento, que permita controlar el proceso de aprendizaje.
Una de las principales ventajas que posee este modelo, es que la red aprende la relación existente entre los datos, adquiriendo la capacidad de generalizar conceptos. De esta manera, una red neuronal puede tratar con información que no le fue presentada durante de la fase de entrenamiento.

4. Algoritmo de retropropagación

El algoritmo backpropagation es el método de entrenamiento más utilizado en redes con conexión hacia delante. Es un método de aprendizaje supervisado de gradiente descendente, en el que se distinguen claramente dos fases: primero se aplica un patrón de entrada, el cual se propaga por las distintas capas que componen la red hasta producir la salida de la misma. Esta salida se compara con la salida deseada y se calcula el error cometido por cada neurona de salida. Estos errores se transmiten hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de las capas intermedias (Fritsch, 1996). Cada neurona recibe un error que es proporcional a su contribución sobre el error total de la red. Basándose en el error recibido, se ajustan los errores de los pesos sinápticos de cada neurona.

La entrada total o neta que recibe una neurona oculta j, es:

$$net_j^p(t) = \sum_{j=1}^{N} w_{ji} * x_i^p(t) + \theta_j$$

donde θ_j es el umbral de la neurona que se considera como un peso asociado a una neurona ficticia con valor de salida igual a 1.
El valor de salida de la neurona oculta j, se obtiene aplicando la función de activación f sobre su entrada neta:

$$y_j^p(t) = f(\, net_j^p(t))$$

De igual forma, la entrada neta que recibe una neurona de salida k, es:

$$net_k^p(t) = \sum_{j=1}^{H} v_{kj} * y_j^p(t) + \theta_k$$

Por último, el valor de salida de la neurona de salida k, es:

$$y_k^p(t) = f(\, net_k^p(t))$$

La salida de la red de cada neurona y_k^p se compara con la salida deseada d_k para calcular el error en cada unidad de salida

$$\delta_k = (d_k - y_k^p)$$

El objetivo es hacer mínimo el error entre la salida obtenida por la red y la salida deseada por el usuario ante la presentación de un conjunto de patrones p denominado grupo de entrenamiento. Así el aprendizaje en las redes backpropagation es de tipo supervisado.

La función de error que se pretende minimizar para cada patrón p viene dada por:

$$E^p = \frac{1}{2}\sum_{k=1}^{M}(d_k - y_k^p)^2$$

Este proceso se repite para el número total de patrones de entrenamiento P, para un proceso de aprendizaje exitoso el algoritmo debe actualizar todos los pesos y ganancias de la red minimizando el error medio cuadrático total descrito en

$$E = \sum_{p=1}^{P} E^p$$

La base del algoritmo backpropagation para la modificación de los pesos es la técnica conocida como gradiente decreciente que es:

$$W(t+1) = W(t) + \Delta W(t)$$

El error que genera una red neuronal en función de sus pesos, genera un espacio de n dimensiones, donde n es el número de pesos de conexión de la red, al evaluar el gradiente del error en un punto de esta superficie se obtendrá la dirección en la cual la función del error tendrá un mayor crecimiento, como el objetivo del proceso de aprendizaje es minimizar el error debe tomarse la dirección negativa del gradiente para obtener el mayor

decremento del error y de esta forma su minimización, condición requerida para realizar la actualización de la matriz de pesos en el algoritmo Backpropagation:

$$W(t+1) = W(t) - \alpha \nabla E[W(t)]$$

siendo α el tamaño del paso o tasa de aprendizaje, que suele ser una constante de tamaño reducido $0 < \alpha < 1$.
La variación de los pesos será proporcional al gradiente de la función de error, así en una neurona de salida:

$$\begin{aligned}
\Delta v_{kj}(t+1) &= -\alpha \frac{\partial E^p}{\partial v_{kj}} = -\alpha \frac{\partial}{\partial v_{kj}} \left(\frac{1}{2} \sum_{k=1}^{M} \left(d_k - y_k^p \right)^2 \right) \\
&= -\alpha \frac{\partial}{\partial v_{kj}} \left(\frac{1}{2} \sum_{k=1}^{M} \left(d_k - f\left(net_k^p(t) \right) \right)^2 \right) \\
&= -\alpha \frac{\partial}{\partial v_{kj}} \left(\frac{1}{2} \sum_{k=1}^{M} \left(d_k - f\left(\sum_{j=1}^{H} v_{kj} * y_j^p(t) + \theta_k \right) \right)^2 \right) \\
&= \alpha \left(d_k - f\left(\sum_{j=1}^{H} v_{kj} * y_j^p(t) + \theta_k \right) \right) \frac{\partial y_k^p}{\partial v_{kj}} \; = \alpha \left(d_k - y_k^p \right) \frac{\partial y_k^p}{\partial v_{kj}} = \alpha \delta_k \frac{\partial y_k^p}{\partial v_{kj}}
\end{aligned}$$

Para calcular $\frac{\partial y_k^p}{\partial v_{kj}}$ se debe utilizar la regla de la cadena, pues el error no es una función explícita de los pesos de la red, así obtenemos que:

$$\frac{\partial y_k^p}{\partial v_{kj}} = \frac{\partial y_k^p}{\partial net_k^p} \frac{\partial net_k^p}{\partial v_{kj}}$$

donde

$$\frac{\partial y_k^p}{\partial net_k^p} = \frac{\partial}{\partial net_k^p} f\left(net_k^p(t) \right) = f'\left(net_k^p(t) \right)$$

y

$$\frac{\partial net_k^p}{\partial v_{kj}} = \frac{\partial}{\partial v_{kj}} \left(\sum_{j=1}^{H} v_{kj} * y_j^p(t) + \theta_k \right) = y_j^p(t)$$

Por lo tanto se tiene

$$\Delta v_{kj}(t+1) = -\alpha \frac{\partial E^p}{\partial v_{kj}} = \alpha * \delta_k^p * y_j^p(t)$$

donde

$$\delta_k^p = \delta_k * f'\left(net_k^p(t)\right)$$

Este algoritmo se denomina backpropagation o de propagación inversa debido a que el error se propaga de manera inversa al funcionamiento normal de la red, de esta forma, el algoritmo encuentra el error en el proceso de aprendizaje desde las capas más internas hasta llegar a la entrada; con base en el cálculo de este error se actualizan los pesos y ganancias de cada capa.
Después de conocer el error en la neurona de salida se procede a encontrar el error en la capa oculta el cual está dado por:

$$\begin{aligned}
\Delta w_{ji}(t+1) &= -\alpha \frac{\partial E^p}{\partial w_{ji}} = -\alpha \frac{\partial}{\partial w_{ji}}\left(\frac{1}{2}\sum_{k=1}^{M}\left(d_k - y_k^p\right)^2\right) \\
&= -\alpha \frac{\partial}{\partial w_{ji}}\left(\frac{1}{2}\sum_{k=1}^{M}\left(d_k - f\left(net_k^p(t)\right)\right)^2\right) \\
&= -\alpha \frac{\partial}{\partial w_{ji}}\left(\frac{1}{2}\sum_{k=1}^{M}\left(d_k - f\left(\sum_{j=1}^{H} v_{kj} * y_j^p(t) + \theta_k\right)\right)^2\right) \\
&= -\alpha \frac{\partial}{\partial w_{ji}}\left(\frac{1}{2}\sum_{k=1}^{M}\left(d_k - f\left(\sum_{j=1}^{H} v_{kj} * f(net_j^p(t)) + \theta_k\right)\right)^2\right) \\
&= -\alpha \frac{\partial}{\partial w_{ji}}\left(\frac{1}{2}\sum_{k=1}^{M}\left(d_k - f\left(\sum_{j=1}^{H} v_{kj} * f\left(\sum_{j=1}^{N} w_{ji} * x_i^p(t) + \theta_j\right) + \theta_k\right)\right)^2\right) \\
&= \alpha \sum_{k=1}^{M}\left(d_k - f\left(\sum_{j=1}^{H} v_{kj} * f\left(\sum_{j=1}^{N} w_{ji} * x_i^p(t) + \theta_j\right) + \theta_k\right)\right)\frac{\partial y_k^p}{\partial w_{ji}} \\
&= \alpha \sum_{k=1}^{M}\left(d_k - y_k^p\right)\frac{\partial y_k^p}{\partial w_{ji}} = \alpha \sum_{k=1}^{M} \delta_k^p \frac{\partial y_k^p}{\partial w_{ji}}
\end{aligned}$$

Para calcular $\frac{\partial y_k^p}{\partial w_{ji}}$ se debe aplicar la regla de la cadena en varias ocasiones puesto que la salida de la red no es una función explícita de los pesos de la conexión entre la capa de entrada y la capa oculta

$$\frac{\partial y_k^p}{\partial w_{ji}} = \frac{\partial y_k^p}{\partial net_k^p}\frac{\partial net_k^p}{\partial y_j^p}\frac{\partial y_j^p}{\partial net_j^p}\frac{\partial net_j^p}{\partial w_{kj}}$$

donde

$$\frac{\partial y_k^p}{\partial net_k^p} = \frac{\partial}{\partial net_k^p}\left(f\left(net_k^p(t)\right)\right) = f'\left(net_k^p(t)\right)$$

$$\frac{\partial net_k^p}{\partial y_j^p} = \frac{\partial}{\partial y_j^p}\left(\sum_{j=1}^{H} v_{kj} * y_j^p(t) + \theta_k\right) = v_{kj}$$

$$\frac{\partial y_j^p}{\partial net_j^p} = \frac{\partial}{\partial net_j^p}\left(f\left(\ net_j^p(t)\right)\right) = f'\left(\ net_j^p(t)\right)$$

$$\frac{\partial net_j^p}{\partial w_{kj}} = \frac{\partial}{\partial w_{kj}}\left(\sum_{j=1}^{N} w_{ji} * x_i^p(t) + \theta_j\right) = x_i^p(t)$$

Por lo tanto se tiene que

$$\Delta w_{ji}(t+1) = -\alpha \frac{\partial E^p}{\partial w_{ji}} = \alpha * \delta_j^p * x_i^p(t)$$

donde

$$\delta_j^p = f'\left(\ net_j^p(t)\right)\sum_{k=1}^{M} \delta_k^p\, v_{kj}$$

Para acelerar el proceso de convergencia de los pesos, Rumelhart et al. (1986) sugirieron añadir un factor momento β, que tiene en cuenta la dirección del incremento tomada en la iteración anterior esto es:

$$\Delta w_{ji}(t+1) = \alpha * \delta_j^p * x_i^p(t) + \ \beta \Delta w_{ji}(t)$$

Resumiendo, el algoritmo backpropagation queda expresado de la siguiente manera:

$$w_{ji}(t+1) = w_{ji}(t) + \left[\alpha * \delta_j^p * x_i^p(t) + \ \beta \Delta w_{ji}(t)\right]$$

Con

$$\delta_j^p = \begin{cases} \delta_j * \ f'\left(net_j^p(t)\right) & si\ j\ es\ una\ neurona\ de\ salida \\ f'\left(\ net_j^p(t)\right)\sum_{k=1}^{M} \delta_k^p\, v_{kj} & si\ j\ es\ una\ neurona\ oculta \end{cases}$$

Como se observa en la ecuación las funciones de activación utilizadas en este tipo de red deben ser continuas para que su derivada exista en todo el intervalo, ya que el término $f'\left(net_k^p(t)\right)$ es requerido para el cálculo del error.

Las funciones de activación más utilizadas en el algoritmo de retropagación y sus respectivas derivadas son las siguientes:

	Función de activación	Derivada
Identidad	$f(x) = x$	$f'(x) = 1$
Sigmoidal logarítmica	$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
Sigmoidal hiperbólica	$f(x) = \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$f'(x) = 1-(f(x))^{2}$

Ya que como se observa no se requiere derivar la función para obtener la derivada en algún punto, esto hace que su costo computacional sea menor y su eficiencia sea mejor.

Conclusiones

- Las redes neuronales pueden solucionar problemas no lineales y de alta complejidad.
- Son un instrumento muy flexible para la solución de problemas, ya que las neuronas pueden reconocer patrones que no han sido aprendidos.
- El algoritmo de retropropación disminuye el costo y tiempo en el proceso de aprendizaje de la red.
- Solo la experiencia puede proporcionar el tipo de topología que se utilizara en la red.

Bibliografía

Britos, M. I. P. (2005). ENTRENAMIENTO DE REDES NEURONALES BASADO EN ALGORITMOS EVOLUTIVOS.

Rivera, E. (2005). Introducción a las Redes Neuronales Artificiales.

Matich, D. J. (2001). Redes Neuronales: Conceptos básicos y aplicaciones.Cátedra de Informática Aplicada a la Ingeniería de Procesos–Orientación I.

Montaño Moreno, J. J. (2002). Redes neuronales artificiales aplicadas al análisis de datos (Doctoral dissertation, Tesis Doctoral, Universidad de Islas Balerares: Palma de Mallorca, España).

Moreno Rodríguez, A. (2009). Desarrollo de una interfaz gráfica de redes neuronales usando Matlab.

Tanco, F. (2003). Introducción a las redes neuronales artificiales. Grupo de Inteligencia Artificial.