**Mapa Conceptual Historia de la de Redes neuronales artificiales**

LINK [text2mindmap.com/xbgeTFd](http://text2mindmap.com/xbgeTFd)

**Redes neuronales artificiales RNA**

**VENTAJAS DE LAS RNA**

Según Eduardo Rivera en su artículo “Introducción a las redes neuronales artificiales” de 2007 [4] las principales ventajas de las redes neuronales artificiales son:

* Aprendizaje Adaptativo: Las RNA aprenden a realizar tareas a partir de un conjunto de datos dados, en el proceso de aprendizaje, estos datos son representados como las entradas y pesos.
* Auto-organización: Pueden crear su propia organización o representación de la información recibida.

Este auto organización provoca la generalización: facultad de las redes neuronales de responder apropiadamente cuando se les presentan datos o situaciones a las que no había sido expuesta anteriormente.

* Operación en tiempo real: Las operaciones realizadas pueden ser llevadas a cabo por computadores paralelos, o dispositivos de hardware especiales que aprovechan esta capacidad.
* Tolerancia a fallos parciales: La destrucción parcial de una red, daña el funcionamiento de la misma, pero no la destruye completamente. Esto es debido a la redundancia de la información contenida.

Aspectos distintos respecto a la tolerancia a fallos:

a) Las redes pueden aprender a reconocer patrones con ruido, distorsionados o incompletos. Esta es una tolerancia a fallos respecto a los datos.

b) Las redes pueden seguir realizando su función (con cierta degradación) aunque se destruya parte de la red.

* En el proceso de aprendizaje, los enlaces ponderados de las neuronas se ajustan de manera que se obtengan ciertos resultados específicos. Una red neuronal no necesita un algoritmo para resolver un problema, ya que ella puede generar su propia distribución de pesos en los enlaces mediante el aprendizaje. También existen redes que continúan aprendiendo a lo largo de su vida, después de completado su período de entrenamiento.
* Operación en tiempo real: los cómputos neuronales pueden ser llevados a cabo en paralelo; para esto se diseñan y fabrican máquinas con hardware especial para obtener esta capacidad.
* Inclusión flexible en la tecnología vigente: se pueden obtener chips especializados para redes neuronales que mejoran su capacidad en ciertas tareas. Ello permitirá la integración de módulos en los sistemas existentes.
* Las redes neuronales se autoajustan a los elementos procesales: Son dinámicas, pues son capaces de estar constantemente cambiando para adaptarse a las nuevas condiciones.
* La función del diseñador es únicamente la obtención de la arquitectura apropiada. No es problema del diseñador el cómo la red aprenderá a discriminar. Sin embargo, sí es necesario que desarrolle un buen algoritmo de aprendizaje que le proporcione a la red la capacidad de discriminar, mediante un entrenamiento con patrones.

**DESVENTAJAS DE LAS RNA**

* Modelar procesos paralelos en máquinas serie puede ser un proceso que consuma mucho tiempo. Como todo en este día y época, el tiempo es esencial, lo que a menudo deja las redes neuronales fuera de las soluciones viables a un problema.
* Falta de reglas definitorias que ayuden a construir una red
* Caja negra (no pueden explicar cómo resuelven un problema. La representación interna generada puede ser demasiado compleja para ser analizada, aún y en los casos más sencillos)
* Complejidad de aprendizaje para grandes tareas, cuanto más cosas se necesita que aprenda la red, más complicado será enseñarle
* Tiempo de aprendizaje elevado. Esto depende de dos factores: primero se incrementa la cantidad de patrones a identificar o clasificar y segundo se requiere mayor flexibilidad o capacidad de adaptación de la red neuronal para reconocer patrones que sean sumamente parecidos, se deberá invertir más tiempo en lograr que la red converja a valores de pesos que representan lo que se quiere enseñar.
* No permite interpretar lo que se ha aprendido, la red por si sola proporciona una salida, un número, que no puede ser interpretado por ella misma sino que se requiere de la intervención del programador y de la aplicación en si para encontrarle un significado la salida proporcionada.
* Elevada cantidad de datos para el entrenamiento, cuanto más flexible se requiera que sea la red neuronal, más información tendrá que enseñarle para que realice de forma adecuada la identificación

**TIPOS DE PROBLEMAS A RESOLVERSE CON RNA**

* Aproximación de funciones
* Clasificación y reconocimiento
* Agrupamiento de datos
* Predicción
* Selección de característica
* Optimización

**APLICACIONES**

∞ Reconocimiento de caracteres manuscritos, impresos, de font antiguo, etc. .

∞ Sistemas de memorias asociativas.

∞ Reconocimiento de voz ∞ Control de robots

∞ Toma de decisiones administrativas, financieras etc.

∞ Reconocimiento de enfermedades

∞ Reconocimiento de señales de radio

∞ Predicción de Señales y series de tiempo Caóticas

∞ Generación de reglas para sistemas expertos

∞ Aplicaciones en economía para predicción

∞ Aplicaciones en geología, meteorología, astronomía

[**FUNCIÓN DE ACTIVACIÓN**](http://rna.50webs.com/tutorial/RNA_intro.html#La función de activación) **APLICABLES A RNA**

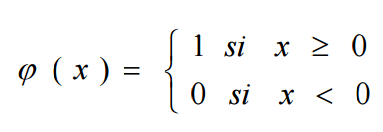
La función de activación se utiliza para limitar el rango de valores de la respuesta de la neurona. Generalmente los rangos de valores se limitan a [0,1] o [-1,1], sin embargo otros rangos son posibles de acuerdo a la aplicación o problema a resolver.

Existen diversas funciones de activación y la decisión entre una u otra dependerá nuevamente de la aplicación o problema a resolver.

Existen funciones de activación comúnmente utilizadas y con las cuales se han obtenido resultados satisfactorios en diversas aplicaciones.

* **La función Umbral o threshold function**

Esta función está definida como



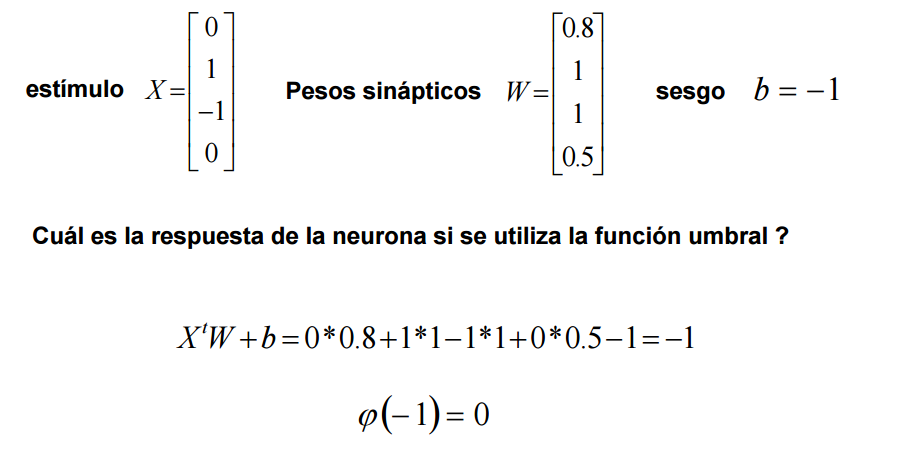
La respuesta de la neurona es entonces 1 o 0.



Generalmente se utiliza para establecer criterios de clasificación.

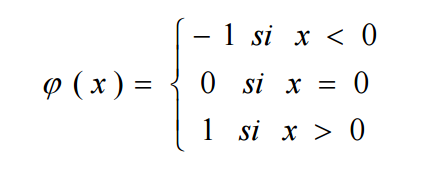
Por ejemplo, 1=Elemento tipo A, 0=Elemento tipo B

**Ejemplo:**



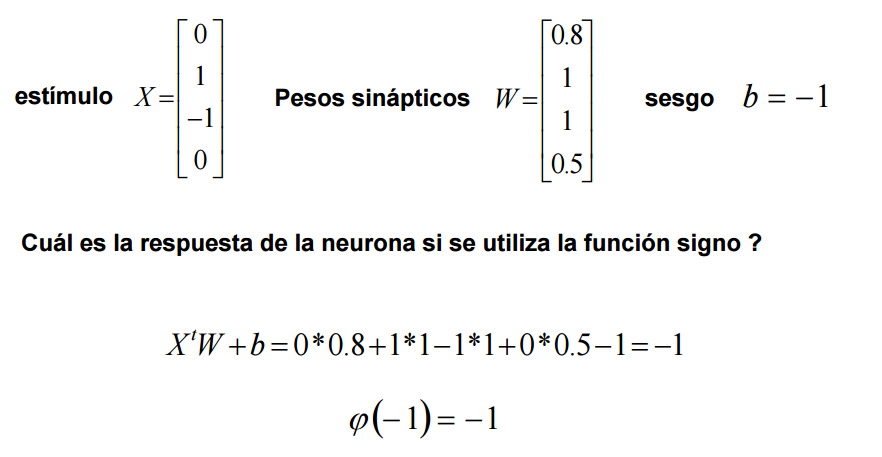
* **La función signo**

Esta función está definida como



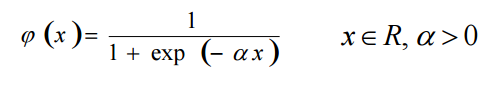
La respuesta de la neurona es entonces -1, 0 o 1.

**Ejemplo:**

****

* **La función logística**

Esta función está definida como

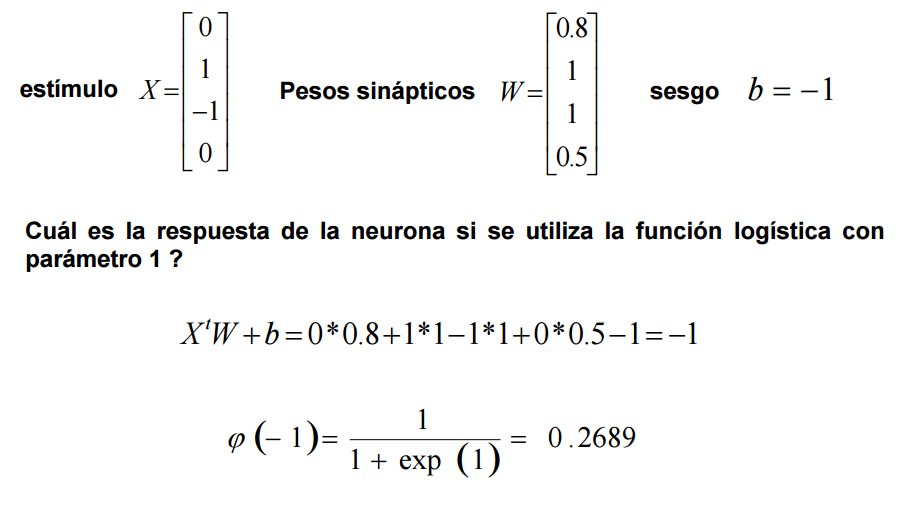


Es una de las funciones más utilizadas en la construcción de redes neuronales

Es continua a valores en [0, 1] e infinitamente diferenciable

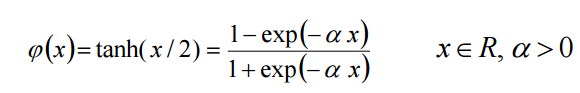
Es la versión continua de la función umbral y se utiliza en problemas de aproximación

**Ejemplo:**

****

* **La función tangente hiperbólica**

Esta función está definida como

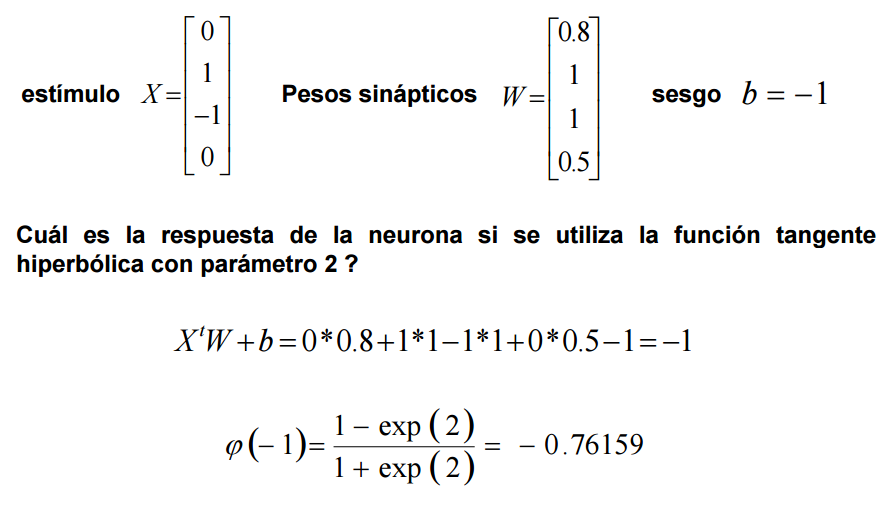
****

Es la versión continua de la función signo y se usa en problemas de aproximación

Es importante por sus propiedades analíticas

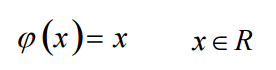
Es continua a valores en [-1, 1] e infinitamente diferenciable

**Ejemplo:**

****

* **La función Lineal**

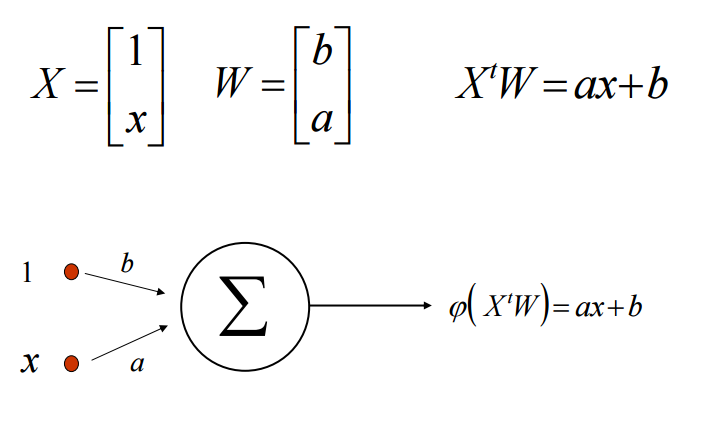
Esta función está definida como

****

No limita la respuesta de la neurona

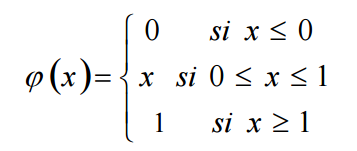
Se utiliza en problemas de aproximación o estimación lineal

Una regresión lineal simple puede ser vista como la respuesta de una neurona con función de activación lineal

****

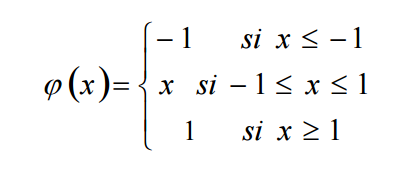
* **La función Lineal acotada**

Esta función está definida como

****

* **La función Lineal acotada simétrica**

Esta función está definida como

****

**Estimulo**

****

**HISTORIA PERCEPTRÓN**

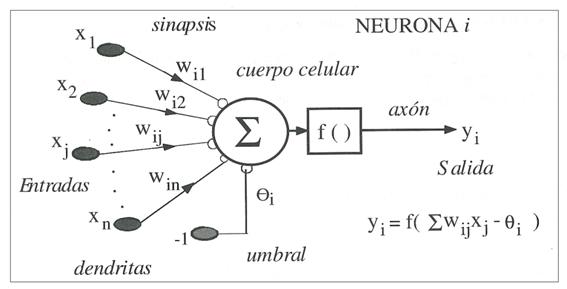
La primera red neuronal conocida, fue desarrollada en 1943 por Warren McCulloch y Walter Pitts; esta consistía en una suma de las señales de entrada, multiplicadas por unos valores de pesos escogidos aleatoriamente. La entrada es comparada con un patrón preestablecido para determinar la salida de la red. Si en la comparación, la suma de las entradas multiplicadas por los pesos es mayor o igual que el patrón preestablecido la salida de la red es uno (1), en caso contrario la salida es cero (0). Al inicio del desarrollo de los sistemas de inteligencia artificial, se encontró gran similitud entre su comportamiento y el de los sistemas biológicos y en principio se creyó que este modelo podía computar cualquier función aritmética o lógica.

La red tipo Perceptrón fue inventada por el psicólogo Frank Rosenblatt en el año 1957. Su intención era ilustrar algunas propiedades fundamentales de los sistemas inteligentes en general, sin entrar en mayores detalles con respecto a condiciones específicas y desconocidas para organismos biológicos concretos. Rosenblatt creía que la conectividad existente en las redes biológicas tiene un elevado porcentaje de aleatoriedad, por lo que se oponía al análisis de McCulloch Pitts en el cual se empleaba lógica simbólica para analizar estructuras bastante idealizadas. Rosenblatt opinaba que la herramienta de análisis más apropiada era la teoría de probabilidades, y esto lo llevó a una teoría de separabilidad estadística que utilizaba para caracterizar las propiedades más visibles de estas redes de interconexión ligeramente aleatorias.

En 1969 Marvin Minsky y Seymour Papert publicaron su libro: "Perceptrons: An introduction to Computational Geometry", el cual para muchos significó el final de las redes neuronales. En el se presentaba un análisis detallado del Perceptrón, en términos de sus capacidades y limitaciones, en especial en cuanto a las restricciones que existen para los problemas que una red tipo Perceptrón puede resolver; la mayor desventaja de este tipo de redes es su incapacidad para solucionar problemas que no sean linealmente separables.

A pesar de esta limitación, el Perceptrón es aún hoy una red de gran importancia, pues con base en su estructura se han desarrollado otros modelos de red neuronal como la red Adaline y las redes multicapa.

**FORMULA**

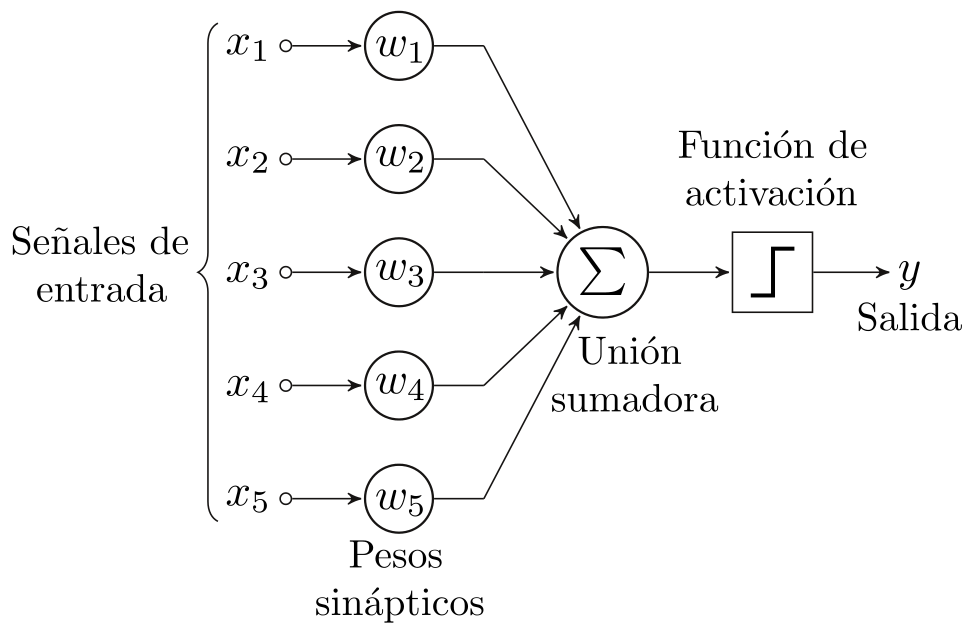


Pesos

Umbral

Entradas

**ESTRUCTURA**

****

**PARA QUE SE USA**

Se usa para demostrar el aprendizaje computacional

Soluciona problemas linealmente separables

**FUNCION DE ACTIVACION DE LA RED PERCEPTRÓN**

ES UNA FUNCION DE TRANSFERENCIA ESCALON

La red tipo Perceptrón emplea principalmente dos funciones de transferencia, escalón con salidas 1, 0 o escalón con salidas 1, -1; su uso depende del valor de salida que se espera para la red, es decir si la salida de la red es unipolar o bipolar; sin embargo la función escalón con salidas 1, -1 es preferida sobre la escalón con salidas 1, 0 ya que el tener un cero multiplicando algunas de los valores resultantes del producto de las entradas por el vector de pesos, ocasiona que estos no se actualicen y que el aprendizaje sea más lento.

No todas las funciones lógicas pueden computarse con un perceptrón simple. Este hecho tiene que ver con la geometría del cubo n-dimensional cuyos vértices representa la combinación de los valores lógicos de los argumentos. Cada función lógica separa los vértices en dos clases: aquéllos para los que la función es l, y aquéllos para los que la función es 0. Si los puntos en los que la función vale 1 pueden separarse mediante un hiperplano de los puntos en los que la función es 0, entonces la función es computable con un perceptrón. Si esto no es así, la función no es computable. Las funciones computables con un perceptrón son las funciones linealmente separables. Un ejemplo de función lógica no linealmente separable es el XOR.

**La función de activación de las neuronas de un perceptrón es del tipo escalón, dando de esta manera sólo salidas binarias. Cada neurona de salida del perceptrón representa a una clase. Una neurona de salida responde con 1 si el vector de entrada pertenece a la clase a la que representa y responde con 0 en caso contrario.**

**REGLA DE APRENDIZAJE DEL PERCEPTRÓN.**

* El algoritmo de aprendizaje del Perceptrón es de tipo supervisado, lo que requiere que sus resultados sean evaluados y se realicen las oportunas modificaciones de los pesos si fuera necesario.
* El algoritmo de entrenamiento del perceptrón se encuentra dentro de los denominados algoritmos por corrección de errores. Este tipo de algoritmos ajustan los pesos de manera proporcional a la diferencia entre la salida actual proporcionada por la red y la salida objetivo, con el fin de minimizar el error producido por la red. Se puede demostrar que este método de entrenamiento converge siempre en un tiempo finito y con independencia de los pesos de partida, siempre que la función a representar sea linealmente separable. El principal problema de este método de entrenamiento es que cuando la función a representar no es linealmente separable el proceso de entrenamiento oscilará y nunca alcanzará la solución. Las funciones no separables linealmente no pueden ser representadas por un perceptrón.

### **Aprendizaje Supervisado**

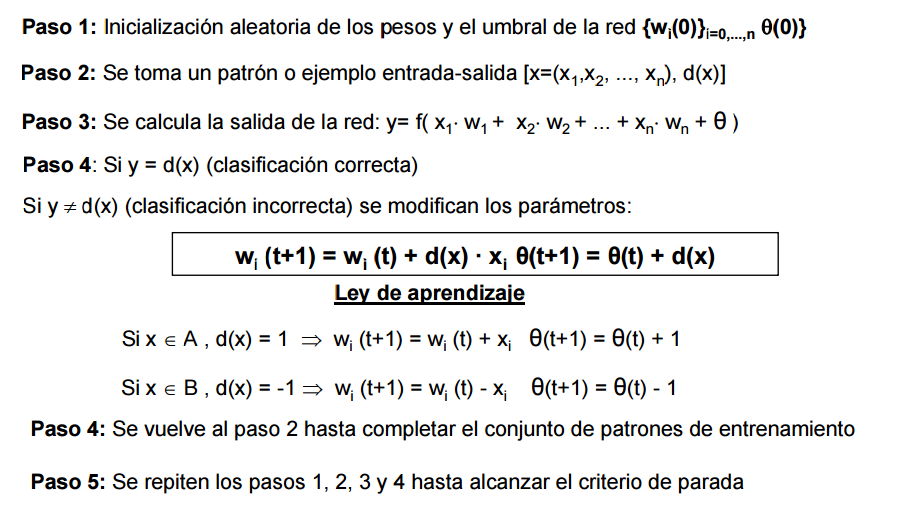
Para el proceso de aprendizaje del Perceptron Simple es necesario saber:

* Los datos son puntos en un espacio multidimensional.
* Se dispone de un conjunto de observaciones, puntos representados en el hiperplano, de los que se sabe su categoría o clase.
* Se tiene que determinar la ecuación del hiperplano que separa los ejemplos de un tipo de los de otro.
* La ecuación del hiperplano se deduce a partir de los ejemplos.
* El aprendizaje es proceso iterativo supervisado.
* Hasta encontrar el hiperplano discriminante se debe ir modificando los parametros de la red (pesos y umbral).
* Este proceso de aprendizaje supervisado tiene un número finito de iteracciones.

**ALGORITMO** **PERCEPTRON**

Los pasos a seguir para el aprendizaje son:

1. Inicialización aleatoria de los pesos y umbral.
2. Se toma un patrón de entrada-salida.
3. Se calcula la salida de la red.
4. Si y = d(x) (clasificación correcta).   
   Si y ≠ d(x) (clasificación incorrecta) se modifican los parámetros.
5. Se vuelve al paso 2 hasta completar el conjunto de patrones de entrenamiento.
6. Se repiten los pasos anteriores hasta alcanzar el criterio de parada.



**Ejemplos del uso del perceptron**

* Pilotos automáticos
* Simulaciones y predicciones de vuelo
* Detención de fallas de componentes de un avión
* Comando por voz
* Identificador de firmas
* Reconocimiento facial

**EL ADALINE**

Desde hace miles de años se ha estudiado el cerebro humano con el fin de modelar su funcionamiento, pero fue con el avance de la electrónica que esta idea tomó fuerza.

El primer paso se dio en 1943 con el neurofisiólogo Warren McCulloch y el matemático Walter Pitts quienes escribieron un documento en el cual explicaban el posible funcionamiento de las neuronas e hicieron un modelo simple de una red neuronal con circuitos eléctricos.

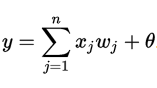
En 1949, Donald Hebb apoya el concepto de neurona y su funcionamiento escribiendo un libro titulado “The Organization of Behavior” en el cual comenta la actividad existente en las neuronas cada vez que son usadas.

En la década de 1950, con el avance de las computadoras, se hizo posible simular una red neuronal. Nathanial Rochester, de los laboratorios de investigación de la IBM, dio el primer paso en la simulación de la red neuronal, pero desafortunadamente fracaso. Posteriormente se hicieron otras simulaciones ya con éxito.

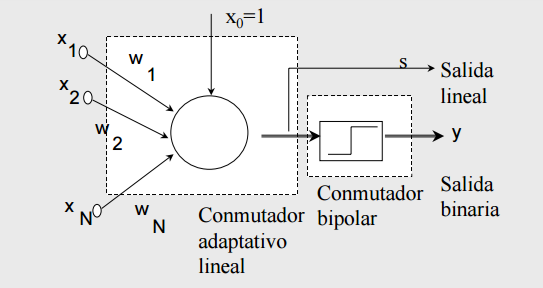
En 1959, Bernard Widrow y Marcian Hoff de la universidad de Stanford desarrollaron un modelo llamado “ADALINE” y “MADALINE”. ADALINE proviene de ADAptive LINear Element y MADALINE de Multiple ADAptive LINear Element (Many Adalines).

**FORMULA**

* x es el vector de entrada
* w es el vector de pesos
* n es el número de entradas
* {\ Displaystyle \ theta}  alguna constante
* y es la salida del modelo



**ESTRUCTURA**



**PARA QUE SE USA**

* Procesamiento de señales (Diseño de filtros capaces de eliminar ruido en señales portadoras de información )
* Filtros adaptivos (Predecir el valor futuro de una señal a partir de su valor actual)
* Asociación de patrones
* Reconocimiento de voz y caracteres

**FUNCION DE ACTIVACION DE LA RED ADALINE**

ES UNA FUNCION DE TRANSFERENCIA LINEAL

**ADALINE APRENDIZAJE**

El entrenamiento se realiza presentando repetidamente una serie de parejas de entradas y salidas. El objetivo de la Adaline durante el proceso de la adaptación es producir la salida deseada como propia suya.

**ALGORITMO LMS**

Este algoritmo (Least Mean Square), similar al perceptrón, se caracteriza por utilizar la regla delta. Para poder efectuar con éxito la derivación que la aplicación de esta regla exige, es necesario que la función de activación sea lineal.

**Pasos**

1.- Inicializar los pesos en forma aleatoria

2.- Introducir un patrón de entrada

3.- Calcular la  salida (y), compararla con la deseada (d) y obtener la diferencia (dp - yp)

4.- Multiplicar el resultado del paso anterior por la entrada correspondiente a cada uno de los pesos  y ponderarla por la tasa de aprendizaje.

5.- Actualizar los pesos, sumando al valor antiguo la cantidad obtenida en el paso anterior

6.- Si no se ha cumplido el criterio de parada, regresar al paso 2, si se ha acabado todos los  patrones repetir el algoritmo.

**Criterios de parada:**

Criterio 1: Fijar un número de ciclos máximo. Dicho número debe garantizar que el error cuadrático para los patrones de entrenamiento se haya estabilizado.

Criterio 2: Cuando el error cuadrático sobre los patrones de entrenamiento no cambia durante x ciclos.

Criterio 3: Cuando el error cuadrático sobre los patrones de validación no aumenta o se mantiene estable a lo largo de x ciclos.

Y se repite el proceso hasta que el error sea menor a un límite previamente establecido.

Para el buen desarrollo de este algoritmo, α debe tomar valores que, teóricamente, deberían ser infinitamente pequeños. Generalmente se obtienen buenos resultados con valores cercanos a 0,1.

**Ejemplos del uso ADALINE**

* Eliminación de ecos en circuitos telefónicos
* Cancelación del ruido materno de grabaciones ECG del latido del feto
* Filtros de ecualización adaptativos en módems de alta velocidad
* Eliminación de ecos en llamadas de larga distancia y comunicaciones vía satélite
* Filtros que eliminan el ruido de señales portadoras de información

**BACKPROPAGATION**

Qué es Backpropagation o Retropropagación?

1. También conocido como retropropagación del error.

2. Método para calcular el gradiente del error

**Backpropagation (BPN)**

La Backpropagation es una red formada por capas, con propagación hacia delante, que está completamente interconectada entre capas. Por lo tanto no existen conexiones de retroalimentación, ni conexiones que salten una capa para ir directamente a otra superior.

La BPN es una red de aprendizaje supervisado, que emplea un ciclo propagación–adaptación de dos fases. Durante el entrenamiento, una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa hacia delante a través de las capas superiores hasta generar una salida. Ésta se compara con la salida deseada y se calcula una señal de error que se propaga hacia atrás, hacia todas las neuronas de las capas ocultas. Sin embargo las neuronas ocultas sólo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida de la red. En función de esta señal de error percibida, se actualizan los pesos de conexión de cada neurona, para hacer que la red converja hacia un estado que permita clasificar correctamente todos los patrones de entrenamiento.

La importancia de este proceso de aprendizaje consiste en que, a medida que se entrena la red, las neuronas de las capas intermedias se organizan a sí mismas de tal modo que las distintas neuronas aprenden a reconocer distintas características del espacio total de entrada. Varias investigaciones han demostrado que la BPN tiende a desarrollar relaciones internas entre neuronas con el fin de organizar los datos de entrenamiento en clases. Esta tendencia se puede extrapolar, para llegar a la hipótesis consistente en que todas las unidades de la capa oculta de una Backpropagation son asociadas de alguna manera a características específicas del patrón de entrada como consecuencia del entrenamiento. Lo que sea o no exactamente la asociación puede no resultar evidente para el observador humano, lo importante es que la red encuentra una representación interna que le permite generar las salidas deseadas cuando se le dan las entradas, en el proceso de entrenamiento.

Esta misma representación interna se puede aplicar a entradas que la red no haya visto antes, y la red clasificará estas entradas según las características que compartan con los ejemplos de entrenamiento.

**PARA QUE SE USA**

* **Mapeo de patrones**
* **Reconocimiento de caracteres**
* **Comprensión de imágenes**
* **Filtrado de ruido**
* **Problema XOR**

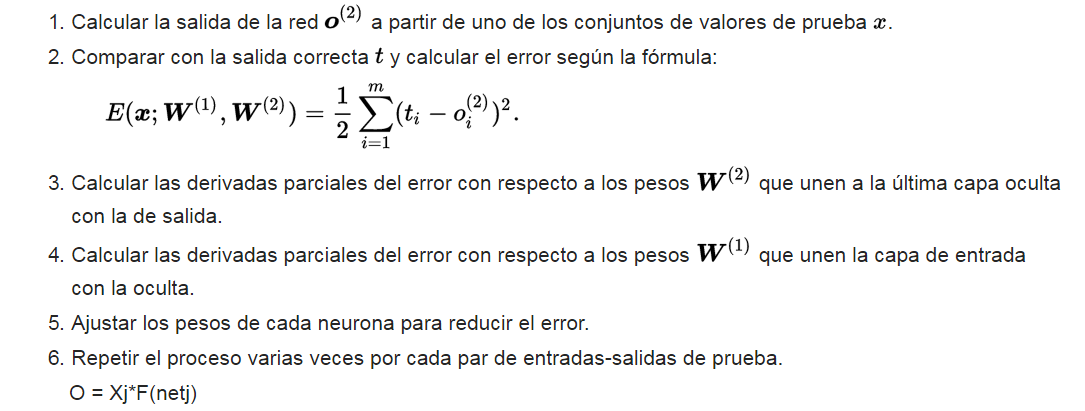
**Algoritmo BackPropagation**

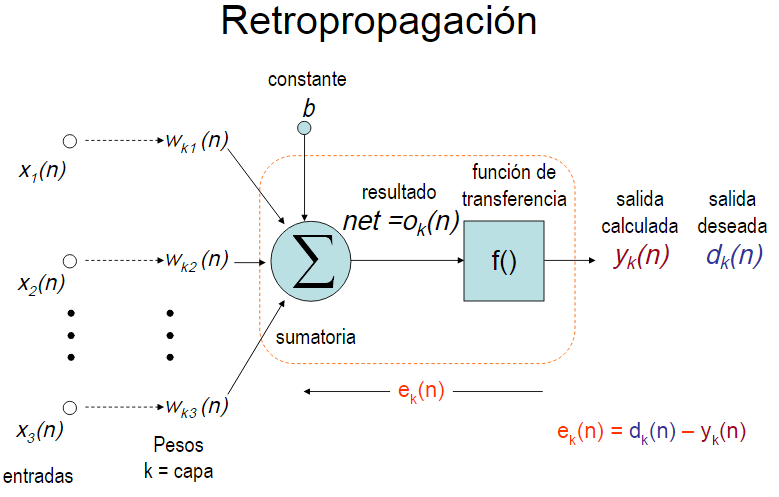
El algoritmo Backpropagation emplea un ciclo propagación – adaptación de dos fases. Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas superiores de la red, hasta generar una salida. La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas.

Las salidas de error se propagan hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa oculta que contribuyen directamente a la salida. Sin embargo las neuronas de la capa oculta solo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido una señal de error que describa su contribución relativa al error total. Basándose en la señal de error percibida, se actualizan los pesos de conexión de cada neurona, para hacer que la red converja hacia un estado que permita clasificar correctamente todos los patrones de entrenamiento.

La importancia de este proceso consiste en que, a medida que se entrena la red, las neuronas de las capas intermedias se organizan a sí mismas de tal modo que las distintas neuronas aprenden a reconocer distintas características del espacio total de entrada. Después del entrenamiento, cuando se les presente un patrón arbitrario de entrada que contenga ruido o que esté incompleto, las neuronas de la capa oculta de la red responderán con una salida activa si la nueva entrada contiene un patrón que se asemeje a aquella característica que las neuronas individuales hayan aprendido a reconocer durante su entrenamiento. Y a la inversa, las unidades de las capas ocultas tienen una tendencia a inhibir su salida si el patrón de entrada no contiene la característica para reconocer, para la cual han sido entrenadas.

**ALGORITMO BACKPROPAGATION - PROPAGACION HACIA ATRÁS**





**Teorema de Komogorov**

Teorema que resuelve parcialmente el problema de los divisores pequeños, que originan problemas de convergencia en sistemas con múltiples frecuencias. Está asociado a sistemas dinámicos con persistencia de movimientos cuasi periódicos.

**TEOREMA 1.**

Si f es un evento nulo o vacío, entonces la probabilidad de que ocurra f debe ser cero.

p(f)=0

**Ejemplo:** La probabilidad de que un estudiante sea mujer es "1 menos la probabilidad de que no sea varón".

**DEMOSTRACIÓN:**

Si sumamos a fun evento A cualquiera, como f y A son dos eventos mutuamente excluyentes, entonces p(AfÈ)=p(A) +p(f)=p(A). LQQD

**TEOREMA 2.**

La probabilidad del complemento de A, Ac debe ser,

p(Ac)= 1 – p(A).

**DEMOSTRACIÓN:**

Si el espacio muestral d, se divide en dos eventos mutuamente exclusivos, A y Ac luego d=AÈAc, por tanto p(d)=p(A) + p(Ac) y como en el axioma dos se afirma que p(d)=1, por tanto, p(Ac)= 1 - p(A) .LQQD

**TEOREMA 3.**

Si un evento A Ì B, entonces la p(A) £ p(B).

**DEMOSTRACIÓN:**

Si separamos el evento B en dos eventos mutuamente excluyentes, A y B \ A (B menos A), por tanto, B=AÈ(B \ A) y p(B)=p(A) +p(B \ A), luego entonces si p(B \ A)³0 entonces se cumple que p(A)£p(B). LQQD

**TEOREMA 4.**

La p( A \ B )= p(A) – p(AÇB)

**DEMOSTRACIÓN:** Si A y B son dos eventos cualquiera, entonces el evento A se puede separar en dos eventos mutuamente excluyentes, (A \ B) y AÇB, por tanto, A=(A \ B)È(AÇB), luego p(A)=p(A \ B) + p(AÇB), entonces, p(A \ B) = p(A) – p(AÇB). LQQD

**TEOREMA 5.**

Para dos eventos A y B, p(AÈB)=p(A) + p(B) – p(AÇB).

**DEMOSTRACIÓN:**

Si AÈB = (A \ B) È B, donde (A \ B) y B son eventos mutuamente excluyentes, por lo que p(A È B) = p(A \ B) + p(B) y del teorema anterior tomamos que p(A \ B) = p(A) – p(AÇB), por tanto, p(AÈB) = p(A) + p(B) – p(AÇB). LQQD

# Bibliografía

*DECSI (Universidad de Granada)* . (s.f.). Obtenido de http://decsai.ugr.es/

*Universidad carlos III de madrid*. (s.f.). Obtenido de http://ocw.uc3m.es/ingenieria-informatica/redes-de-neuronas-artificiales

*wikipedia*. (s.f.). Obtenido de es.wikipedia.org