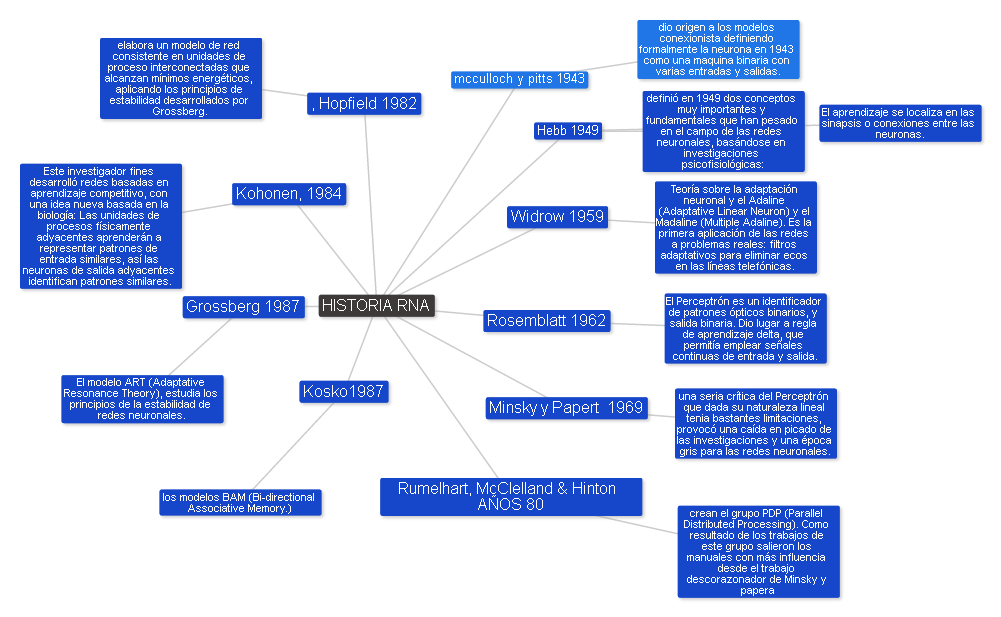
|  |
| --- |
| corporacion de estudios tecnologicos del norte del valle |
| Ingenieria Artificial |
| Redes Neuronales Artificiales |
|  |
| **Juan Carlos Serna Gomez** |
| **02/09/2016** |

1. **HISTORIA DE LAS REDES NEURONALES ARTIFICIALES**



1. **VENTAJAS DE LAS REDES NEURONALES ARTIFICIALES**

Las redes neuronales artificiales (RNA) tienen muchas ventajas debido a que están basadas en la estructura del sistema nervioso, principalmente el cerebro.

**Aprendizaje:** Las RNA tienen la habilidad de aprender mediante una etapa que se llama etapa de aprendizaje. Esta consiste en proporcionar a la RNA datos como entrada a su vez que se le indica cuál es la salida (respuesta) esperada.

**Auto organización:** Una RNA crea su propia representación de la información en su interior, descargando al usuario de esto.

**Tolerancia a fallos:** Debido a que una RNA almacena la información de forma redundante, ésta puede seguir respondiendo de manera aceptable aun si se daña parcialmente.

**Flexibilidad:** Una RNA puede manejar cambios no importantes en la información de entrada, como señales con ruido u otros cambios en la entrada (ej. si la información de entrada es la imagen de un objeto, la respuesta correspondiente no sufre cambios si la imagen cambia un poco su brillo o el objeto cambia ligeramente)

**Tiempo real:** La estructura de una RNA es paralela, por lo cuál si esto es implementado con computadoras o en dispositivos electrónicos especiales, se pueden obtener respuestas en tiempo real.

Hay muchas buenas razones para el uso de redes neuronales y los avances en este campo incrementarán su popularidad. Son excelentes como clasificadores/reconocedores de patrones y pueden ser usadas donde las técnicas tradicionales no funcionan. Las redes neuronales pueden manejar excepciones y entradas de datos anormales, muy importante para sistemas que manejan un amplio rango de datos (sistemas de radar y sonar, por ejemplo). Muchas redes neuronales son biológicamente plausibles, lo que significa que pueden proveer pistas de como trabaja el cerebro según progresen. Avances en la neurociencia también ayudarán al avance en las redes neuronales y hasta el punto en que sean capaces de clasificar objetos con la precisión de un humano y la velocidad de una computadora! El futuro es brillante.

• Complejidad de aprendizaje para grandes tareas, cuanto más cosas se necesiten que aprenda una red, mas complicado será enseñarle.

• Tiempo de aprendizaje elevado. Esto depende de dos factores: primero si se incrementa la cantidad de patrones a identificar o clasificar y segundo si se requiere mayor flexibilidad o capacidad de adaptación de la red neuronal para reconocer patrones que sean sumamente parecidos, se deberá invertir mas tiempo en lograr que la red converja a valores de pesos que representen lo que se quiera 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 a la salida proporcionada.

• Elevada cantidad de datos para el entrenamiento, cuanto mas flexible se requiera que sea la red neuronal, mas información tendrá que enseñarle para que realice de forma adecuada la identificación.

Otros problemas con las redes neuronales son la falta de reglas definitorias que ayuden a realizar una red para un problema dado.

### [Definicion, Ventajas y Desventajas - REDES NEURONALES](http://redes-neuronales.wikidot.com/definicion-ventajas-desventajas)

**redes**-**neuronales**.wikidot.com/definicion-**ventajas**-**desventajas**

1. **APLICACIONES GENERALES**

Las RNA son una tecnología computacional emergente que puede utilizarse en un gran número y variedad de aplicaciones. A continuación, proporcionamos

un listado de aplicaciones de RNA en diferentes campos (McCord Nelson y Illingworth, 1991; Hilera y Martínez, 1995; Buscema, 1997):

*–Biología*

Estudio del cerebro

Obtención de modelos de retina

*–Empresa*

Identificación de candidatos para posiciones específicas

Reconocimiento de caracteres escritos

Predicción del rendimiento económico de las Empresas

*-Medio ambiente*

Previsión del tiempo

*–Finanzas*

Previsión de la evolución de los precios

Valoración del riesgo de los créditos

Identificación de firmas

*–Manufacturación*

Robots automatizados y sistemas de control

(visión artificial y sensores de presión, temperatura,gas, etc.)

Control de producción en líneas de proceso

*–Medicina*

Diagnóstico y tratamiento a partir de síntomas y/o de datos analíticos (electrocardiograma, encefalograma, análisis sanguíneo, cuestionarios, etc.)

Monitorización en cirugía Predicción de reacciones adversas a los medicamentos

Lectores de rayos X

*–Militares*

Clasificación de las señales de radar

Creación de armas inteligentes

Reconocimiento y seguimiento de tiro al blanco , Detección de bombas

*–Psicología y Psiquiatría*

Modelización de procesos psicológicos básicos

Reconocimiento del habla (análisis e interpretación de frases habladas)

Diagnóstico de diversos trastornos (demencia, epilepsia, alcoholismo, etc.) en función de señales

EEG

Clasificación de las fases del sueño

Diagnóstico psicológico

Predicción de rendimiento académico

### [¿Qué son las redes neuronales artificiales? Aplicaciones realizadas ...](https://www.google.com.co/url?sa=t&rct=j&q=&esrc=s&source=web&cd=9&ved=0ahUKEwiZ6aewqefOAhVEox4KHRZ8CCkQFghWMAg&url=http%3A%2F%2Fdisi.unal.edu.co%2F%7Elctorress%2FRedNeu%2FLiRna001.pdf&usg=AFQjCNHNT4e01RQqAbd8mltQHnl8NTXiyA&sig2=0l1OcHUP5CkwWZXBCzbFcQ)

disi.unal.edu.co/~lctorress/**Red**Neu/LiRna001.pdf

1. [**FUNCIÓN DE ACTIVACIÓN**](http://rna.50webs.com/tutorial/RNA_intro.html#La%20funci%C3%B3n%20de%20activaci%C3%B3n)**:**

Es quizás la característica principal o definitoria de las neuronas, la que mejor define el comportamiento de la misma. Se usan diferentes tipos de funciones, desde simples funciones simples de umbral a funciones no lineales. Se encarga de calcular el nivel o estado de activación de la neurona en función de la entrada total.

**TIPOS DE FUNCIONES DE ACTIVACIÓN**

* ***Función identidad***

          Se trata de una función muy simple que no tiene en cuenta el anterior estado de activación de la unidad. Se puede expresar con la siguiente fórmula:

**a**i(t+1) = **∑**j **w**ij **o**j(t)

          Según esta función de activación, el estado de activación que le corresponde a la unidad *i* (**a**i(t+1)) tras recibir las señales desde las unidades a las que está conectada es el resultado de sumar todas las señales recibidas (**o**j), ponderadas cada una de ellas por el peso sináptico (**w**ij) asociado a la sinapsis por la que se transmite la señal.

          Podemos representar el conjunto de estímulos de entrada que le llegan a una unidad desde la capa anterior como un vector, y los pesos sinápticos correspondientes mediante una matriz; en esta circunstancia, el cálculo del nuevo estado de activación de todas las neuronas se obtiene simplemente mediante el producto de matrices.

          En algunas redes se incluye en la función de activación un factor que representa una fracción del valor de activación anterior, por ejemplo:

**a**i(t+1) =  ½ **a**i(t) + **∑**j **w**ij **o**j(t)

          Con esta variante conseguimos que la unidad o neurona mantenga cierto estado de activación durante varios ciclos, incluso en el caso de que las entradas sean cero, en esta última circunstancia en cada período de tiempo su estado de activación corresponderá a la mitad del valor de activación anterior. Con este tipo de funciones de activación se garantiza que cada neurona no varíe en su estado y en su respuesta demasiado bruscamente (recordamos que la respuesta de la red depende en gran medida de los estados de activación de las unidades que la componen).

* ***Función de activación tipo umbral***

          Esta función es algo más complicada que la anterior y se utiliza con frecuencia en las redes conexionistas, en particular cuando cada unidad sólo puede tener dos estados: activada o desactivada (1 ó 0, respectivamente). Se suele representar la función tipo umbral del siguiente modo:

|  |  |  |
| --- | --- | --- |
| **a**i(t+1) = | 1 si **∑**j **w**ij **o**j(t) > umbral |  |
| 0 si **∑**j **w**ij **o**j(t) ≤ umbral |  |

            Estas fórmulas quieren indicar, simplemente, que la neurona *i* se activará (tomará el valor 1) si y sólo si la suma ponderada de sus entradas supera cierto umbral, umbral que ha de ser decidido por el diseñador de la red.

**Lineal**:  
Algunas redes neuronales usan esta función de activación como el Adeline por su eficiencia y facilidad.

**Escalón:**   
Esta función es la más usada para redes neuronales binarias ya que no es lineal y es muy simple. Algunas redes que usan esta función son el Perceptrón y Hopfield. Para redes que trabajan en el rango [-1,1] se usa la funcion signo

**Hiperbólicas o tangenciales:**Las redes con salidas continuas, como el Perceptron multicapa con retropropagación, usan esta función ya que su algoritmo de aprendizaje necesita una función derivable

### [Función sigmoidal - RNA](http://rna.50webs.com/tutorial/RNA_intro.html)

rna.50webs.com/tutorial/RNA\_intro.html

1. **PERCEPTRON**

## HISTORIA

McCullogh y Pitts realizaron en 1943 un estudio biológico del cerebro obteniendo un modelo formal de neurona, con lo que introdujeron así el concepto de umbral: una neurona responde a un cierto estímulo siempre que éste sobrepase un cierto umbral de activación.

Posteriormente, en 1949, Hebb desarrolló el Hebbian Learning: aprendizaje mediante adaptación de sinapsis o reforzamiento de las conexiones.

En 1959, Rosenblat definió el perceptrón, uno de los conceptos más importantes dentro del desarrollo de las Redes Neuronales: el perceptrón consiste en una estructura más una regla de aprendizaje o regla del perceptrón. Esa estructura es la combinación de una neurona y una función de salida que es la que define el umbral de activación. La misión de la neurona es implementar una combinación lineal de las entradas. Cada entrada posee un peso, que se adapta temporalmente. Es esto lo que se conoce como aprendizaje.

Misky y Papert desarrollaron en 1969 un perceptrón unicapa que conseguía una clasificación de primer orden (XOR). Se planteó entonces el problema del entrenamiento de varias capas. Así, en 1974 Werbos definió el algoritmo de retropropagación y el uso de la función sigmoidal como función de salida de un perceptrón.

1. **FORMULA MATEMATICA**

Las entradas *Xi* representan las señales que provienen de otras neuronas y que son capturadas por las dendritas.

Los pesos *Wi* son la intensidad de la sinápsis que conecta dos neuronas; tanto *Xi* como *Wi* son valores reales.

es la función umbral que la neurona debe sobrepasar para activarse este proceso ocurre biológicamente en el cuerpo de la célula

Las señales de entrada a una neurona artificial *X1, X2,.., Xn* son variables continuas en lugar de pulsos discretos, como se presentan en una neurona biológica. Cada señal de entrada pasa a través de una ganancia o peso, llamado peso sináptico o fortaleza de la conexión cuya función es análoga a

la de la función sináptica de la neurona biológica. Los pesos pueden ser positivos (excitatorios), o negativos (inhibitorios), el nodo sumatorio acumula todas las señales de entradas

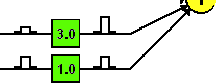
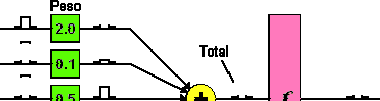


multiplicadas por los pesos o ponderadas y las pasa a la salida a través de una función umbral o función de transferencia. La entrada neta a cada unidad puede escribirse de la siguiente manera.



Una idea clara de este proceso se muestra en la figura 1.3.3, en donde puede observarse el recorrido de un conjunto de señales que entran a la red.







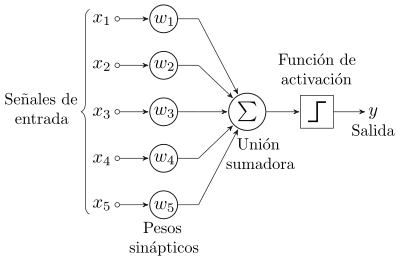
Una vez que se ha calculado la activación del nodo, el valor de salida equivale a

## 

Donde representa la función de activación para esa unidad, que corresponde a la función

escogida para transformar la entrada netai en el valor de salida y que depende de las características específicas de cada red.

1. **DIBUJO DE UN PERCEPTRON**



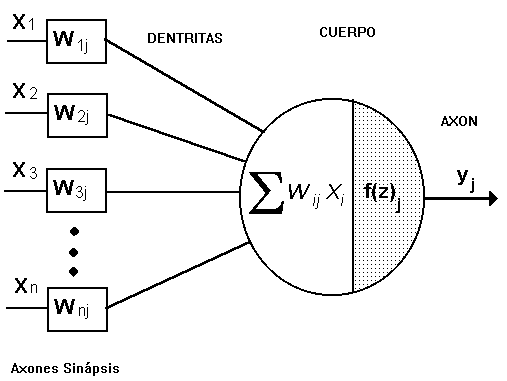
1. **PARA QUE SE USA**

su éxito incial se debió a que era capaz de aprender y reconocer patrones sencillos. Con el desarrollo del perceptrón, surge el área de las Redes Neuronales Artificiales dentro de la Inteligencia Artificial. Sin embargo, **Marvin Minsky**y **Seymur Papert** escriben el libro **"Perceptrons"**, en el que se hace un análisis del Perceptrón mostrando sus flaquezas y decae el apoyo dado a la investigación de las Redes Neuorales Artificiales durante algunas décadas.

Las principales limitaciones del perceptrón son que sirve únicamente para problemas linealmente separables y que sean de dos clases. Hablando vulgarmente, esto quiere decir que el perceptrón sólo lo podemos usar cuando el problema sea distinguir entre una de dos posibles clases y, que trazando una línea, plano o hiperplano en un plano o hiperplano, se puedan separar perfectamente estas dos clases.

1. **CUAL ES SU FUNCION DE ACTIVACION**

En la siguiente figura se representa una neurona "artificial", que intenta modelar el comportamiento de la neurona biológica. Aquí el cuerpo de la neurona se representa como un sumador lineal de los estímulos externos zj, seguida de una función no lineal yj = f(zj). La función f(zj) es llamada la función de activación, y es la función que utiliza la suma de estímulos para determinar la actividad de salida de la neurona.



Este modelo se conoce como perceptrón de McCulloch-Pitts, y es la base de la mayor parte de la arquitectura de las RNA que se interconectan entre sí. Las neuronas emplean funciones de activación diferentes según la aplicación, algunas veces son funciones lineales, otras funciones sigmoidales (p.ej. la tanh), y otras funciones de umbral de disparo. La eficiencia sináptica se representa por factores de peso de interconección wij, desde la neurona i, hasta la neurona j.

Los pesos pueden ser positivos (excitación) o negativos (inhibición). Los pesos junto con las funciones f(z) dictan la operación de la red neuronal. Normalmente las funciones no se modifican de tal forma que el estado de la red neuronal depende del valor de los factores de peso (sinápsis) que se aplica a los estímulos de la neurona.

En un perceptrón, cada entrada es multiplicada por el peso W correspondiente, y los resultados son sumados, siendo evaluados contra el valor de umbral, si el resultado es mayor al mismo, el perceptrón se activa.

### [Perceptron](http://electronica.com.mx/neural/informacion/perceptron.html)

electronica.com.mx/neural/informacion/**perceptron**.html

1. **¿COMO SE ENTRENA UN PERCEPTRON INDIQUE EL NOMBRE DEL ALGORITMOS Y LOS PASOS?**

**Algoritmo del perceptron DISCRETO**

**VARIABLES**

I (nº de iteraciones)

K (contador de ejemplos)

W[i:1..n+1] (pesos del perceptron)

X[i:1..n] (valores de la entrada en el ejemplo actual)

Y (valor de la salida en el ejemplo actual)

SP (salida del perceptrón con la entrada del ejemplo actual)

Error (error en el ejemplo actual)

T (Variable para indicar si el vector es correcto)

PASO 1: Inicialización de variables

PASO 2: Bucle de iteraciones (hasta condición de parada)

Paso 2.1: Bucle de paso por todos los ejemplos

2.1.0 Leer valores del ejemplo

2.1.1 Calcular error en ese ejemplo

2.1.2 Actualizar los pesos según el error de ese ejemplo

2.1.2.1 Actualizar los pesos de las entradas

2.1.2.2 Actualizar el bias (= -umbral)

2.1.3 Incrementar contador de ejemplos

Paso 2.2.: Ver si el vector de pesos es correcto

Paso 2.3.: Incrementar el contador de iteraciones

PASO 3: Salida

**PASO 1: Inicialización de variables**

*I=0;*

*Para i=1..n+1*

*W[i]=<nº aleatorio* (normalmente entre –1 y 1*)>* (muy importante que el nº sea aleatorio,

distinto en cada ejecución)

*T=false*

**PASO 2: Bucle de iteraciones**

*Mientras (I < NºMaxIteraciones) y (no T)*

**Paso 2.1: Bucle de paso por todos los ejemplos**

*Abrir(<fichero de ejemplos>)*

*K=0*

*Mientras no este en el final del fichero*

**2.1.0 Leer valores del ejemplo**

*Leer(x[i] (i=1..n), y)*

**2.1.1 Calcular error en ese ejemplo**

*SP = W[n+1] +* Σ *x[i] \* W[i]* (salida del perceptrón con la entrada del ejemplo h)

*Si SP < 0, entonces SP= 0, en otro caso SP=1*

*Error = y - SP* (diferencia entre el valor de y en el ejemplo h y SP)

**2.1.2 Actualizar los pesos según el error de ese ejemplo**

**2.1.2.1 Actualizar los pesos de las entradas**

*Para cada i=1..n*

*W[i] = W[i] + x[i]\*Error*

**2.1.2.2 Actualizar el bias (= -umbral)**

*W[n+1]=W[n+1] + Error*

**2.1.3 Incrementar contador de ejemplos**

*k=k+1*

*Cerrar(<fichero de ejemplos>)*

**Paso 2.2.: Ver si el vector de pesos es correcto**

*T= True*

*Abrir(<fichero de ejemplos>)*

*Mientras ( no este en el final del fichero y T)*

*Leer(x[i] (i=1..n), y)*

*SP = W[n+1] +* Σ *x[i] \* W[i]*

*Si SP < 0, entonces SP= 0, en otro caso SP=1*

*Error = y – SP*

*Si Error ≠ 0, T = False*

*Cerrar(<fichero de ejemplos>)*

**Paso 2.3.: Incrementar el contador de iteraciones**

*I=I+1*

**PASO 3: Salida**

*Escribe( “El perceptrón aprendido es el de pesos:” )*

*Para i=1..n*

*Escribe(“ W”,i,”=”,W[i])*

*Escribe(“Con bias =”, W[n+1])*

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

### APLICACIONES DEL PERCEPTRÓN

El rango de tareas que el Perceptrón puede manejar es mucho mayor que simples decisiones y reconocimiento de patrones. Por ejemplo, se puede entrenar una red para formar el tiempo pasado de los verbos en ingles, leer texto en ingles y manuscrito.

El Perceptrón multicapa  (MLP)  puede ser usado para la predicción de una serie de datos en el tiempo; tal a sido su  éxito en la medición de  la demanda de gas y electricidad, además de la predicción de cambios en el valor de los instrumentos financieros.

Predicción de mercados financieros, diagnósticos médicos, el Perceptrón como una red codificadora, el Perceptrón aprende a sumar enteros.

NETtalk es un Perceptrón que es capaz de transformar texto en ingles en sonido individual (representaciones fonéticas) y la pronunciación con la utilización de un sintetizador de voz; cuenta con aproximadamente 300 nodos de neuronas (siendo 80 en la capa escondida)  y 20,000 conexiones individuales.

1. **ADALINE**
2. **HISTORIA**

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).

ADALINE fue desarrollado para el reconocimiento de patrones binarios, por ejemplo predecir el siguiente bit en una línea telefónica.

ADALINE funciona tomando la suma de los pesos de las entradas y produce una salida con 0 o 1 dependiendo si pasa o no un umbral, esto haciendo analogía al funcionamiento de una neurona que se dispara si la actividad total procedente de las conexiones con las otras neuronas sobrepasa un nivel.

Varias ADALINE pueden ser organizadas en capas de tal manera que se obtengan grandes arquitecturas formando una red MADALINE la cual produce funciones más complicadas.

MADALINE fue la primera red neuronal aplicada a un problema real, se uso como un filtro para eliminar el eco en las líneas telefónicas.

En los años siguientes el neuro-biólogo Frank Rosenblatt comenzó a trabajar con el perceptrón. Se encontró que una simple capa de perceptrones era suficiente para clasificar un conjunto de valores continuos en una de dos clases.

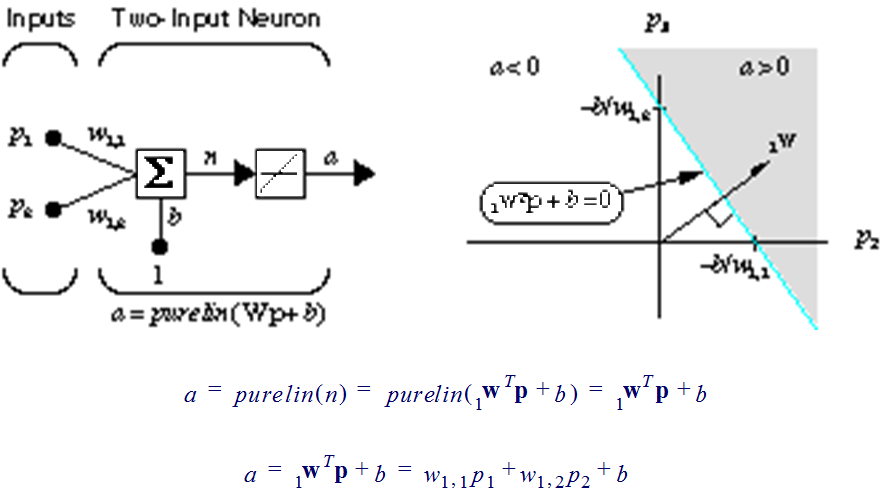
Desafortunadamente el perceptrón estaba limitado y fue probado por Marvin Minsky y Seymour Papert durante los años de desilusión en el libro “Perceptrons”.

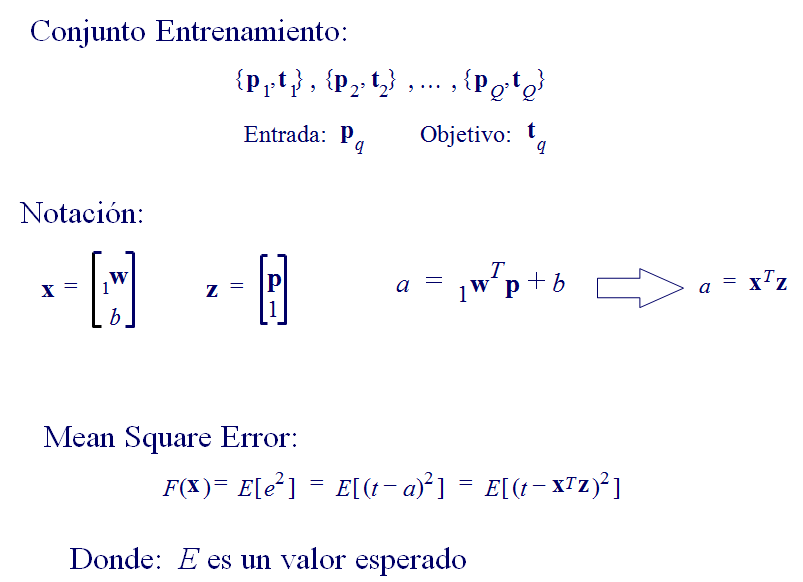
Durante los próximos 20 años se dejo atrás el estudio de las redes neuronales y fue hasta 1982 con John Hopfield que se retomó el interés en la materia. Hopfield presentó un documento a la Academia Nacional de Ciencias en el cual a través de análisis matemático mostraba como las redes neuronales funcionan y para que podrían servir.

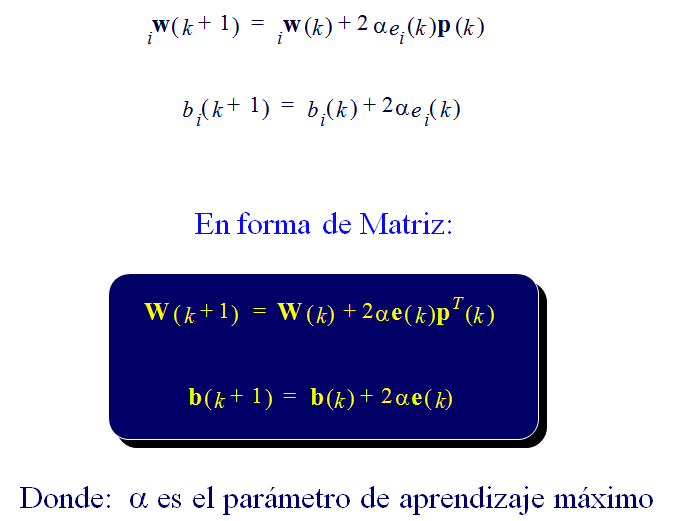
### DOC][ADALINE Y MADALINE](http://cursos.itam.mx/akuri/2002/S22002/RNS/Presentaciones/Adaline/ADALINEYMADALINE2.doc)

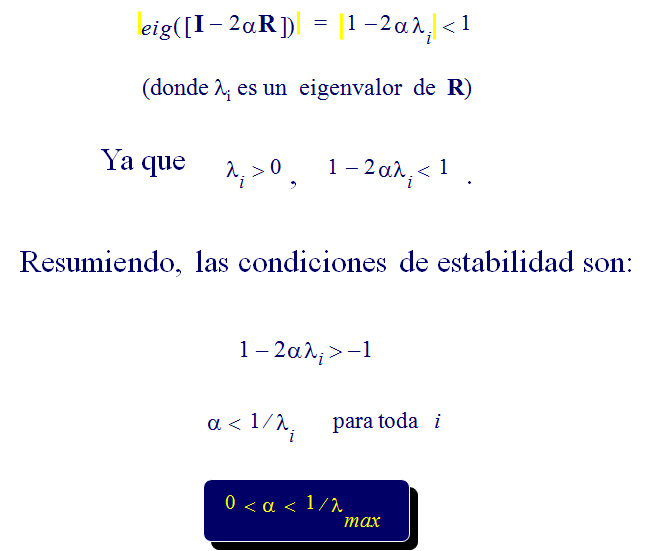
cursos.itam.mx/akuri/2002/S22002/RNS/.../**Adaline**/**ADALINE**Y**MADALINE**2.doc

**FORMULA MATEMATICA**

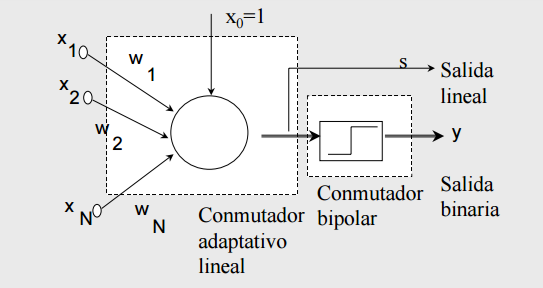








**ESTRUTURA DE LA RED ADALINE**



**FUNCION DE ENTRENAMIENTO**

1. **USO DE ADALINE**

## ADALINE que por sus siglas en inglés significa ADAptive LINear Element es un dispositivo que consta de un solo elemento de procesamiento, por lo que técnicamente no es una red. Sin embargo es un elemento muy importante, ya que de él se derivan redes más complejas.

Una ADALINE consiste de un ALC (Adaptive Linear Combiner) y un cuantizador (función bipolar de salida). Se alimenta con un vector de entrada (valores observados) y con una entrada constante igual a 1 denominada sesgo (bias). Posteriormente se efectúa una suma ponderada de los valores de entrada con sus pesos asociados; si el resultado es positivo, la salida del ADALINE es 1, en caso contrario es 0 (o -1). En consecuencia ADALINE sólo resuelve adecuadamente problemas binarios linealmente separables.

La Red Adaline es un mecanismo físico capaz de realizar aprendizaje es un elemento Combinador Adaptativo, recibe un conjunto de entradas y las combina para producir una salida entonces ADALINE es un modelo de aprendizaje.

Las redes Adaline se basan en el objetivo de evaluar  de la forma más  correcta la salida, para de esta manera poder minimizar el error medio cuadrático, por tanto son muy similares al Perceptrón, lo único en lo que cambian es la transferencia en la salida.

Adaline tiene una considerable diferencia con respecto a Perceptron, puesto que en la modificación de los pesos que se dan en la preparación, la Red Adaline tiene muy presente el valor de corrección de la salida estimada con relación a la esperada.

### ALGORITMO DE ENTRENAMIENTO

El entrenamiento del ADALINE está basado en la regla LMS (Least Mean Square) que busca minimizar el error cuadrático medio por medio de la regla delta.

Dado un vector de entrada (x0, ..., xn) donde x0 =1, el correspondiente valor de pesos (w0, ..., wn) y el valor deseado de salida d, el error cuadrático es el siguiente:



El objetivo del método es minimizar dicho error mediante la modificación del vector de pesos (w0, ..., wn) sumándole un **w** de tal forma que nos acerquemos al error mínimo en la dirección del gradiente negativo, es decir, lo más rápidamente posible.

El procedimiento de derivación se presenta a continuación:



Por lo que el error local será reducido más rápidamente si se ajustan los pesos de acuerdo a la regla delta:



1. **APLICACIONES DE ADALINE**

Procesamiento de Señales.

Filtros que eliminen el ruido en señales portadoras de información.

Filtros de ecualización adaptativos en Módems de alta velocidad.

Cancelación del ruido materno de grabaciones ECG del latido del feto humano.

Aplicaciones Eliminación de ecos en circuitos telefónicos.

Canceladores adaptativos del eco para el filtrado de señales en comunicaciones telefónicas de larga distancia y comunicaciones vía satélite.

1. **¿QUE ES Y PARA QUE SE USA LAS REDES RETROPROGRAMACION?**

Al hablar de redes de retropropagación o redes de propagación hacia atrás hacemos referencia a un algoritmo de aprendizaje más que a una arquitectura determinada. La retropropagación consiste en propagar el error hacia atrás, es decir, de la capa de salida hacia la capa de entrada, pasando por las capas ocultas intermedias y ajustando los pesos de las conexiones con el fin de reducir dicho error. Hay distintas versiones o reglas del algoritmo de retropropagación y distintos arquitecturas conexionistas a las que pueden ser aplicados.

Durante mucho tiempo no se dispuso de algoritmos para entrenar redes multicapa, y como las redes de una capa estaban muy limitadas en cuanto a lo que eran capaces de representar, el campo de las redes neuronales artificiales estaba estancado. La invención y perfeccionamiento del algoritmo de retropropagación dio un gran impulso al desarrollo de este campo. Tiene un buen fundamento matemático y a pesar de sus limitaciones ha expandido enormemente el rango de problemas donde se aplican las redes neuronales artificiales.

La red Back-Propagation está basada en la generalización de la regla delta

Al igual que el Perceptron, ADALINE y MADALINE, la red Back-Propagation se caracteriza por tener una arquitectura en niveles y conexiones estrictamente hacia adelante entre las neuronas.

Primero fase se aplica un patrón de entrada como estímulo para la primera capa de las neuronas de la red, se va propagando a través de todas las capas superiores hasta generar una salida.

Después se compara el resultado obtenido en las neuronas de salida con la salida que se desea obtener y se calcula un valor del error para cada neurona de salida.

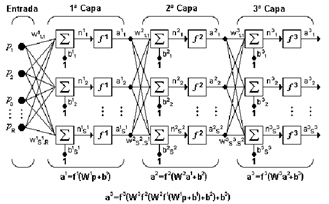
Segundo fase estos errores se transmiten hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida, recibiendo el porcentaje de error aproximado a la participación de la neurona intermedia en la salida original.

La importancia de este algoritmo consiste en su capacidad de auto adaptar los pesos de las neuronas de las capas intermedias para aprender la relación que existe entre un conjunto de patrones dados como ejemplo y sus salidas correspondientes.

### [RED De Retro-propagación Neuronal - SlideShare](http://es.slideshare.net/mentelibre/red-de-retropropagacin-neuronal)

es.slideshare.net/mentelibre/**red**-de-**retro**propagacin-neuronal

**8.EXPLIQUE EL ALGORITMO BLACKPROGRAMATION USANDO GRAFICOS**



Puede notarse que esta red de tres capas equivale a tener tres redes tipo Perceptrón en cascada; la salida de la primera red, es la entrada a la segunda y la salida de la segunda red es la entrada a la tercera. Cada capa puede tener diferente número de neuronas, e incluso distinta función de transferencia.

**10. ¿QUE ES EL TEOREMA DE KOLMOGOROV, EXPLIQUELO?**

El teorema KAM establece que, si un sistema está sometido a una pequeña perturbación no lineal, algunos toros serán deformados y otros destruidos. Los que sobreviven son aquellos que tienen un cociente de frecuencias suficientemente irracional. Es decir, se destruyen aquellos cuyo cociente de frecuencias se acerca más a un número racional, dados por la relación

| ω 2 ω 1 − m s | > k ( ϵ ) s {\displaystyle \left\vert {\frac {\omega \_{2}}{\omega \_{1}}}-{\frac {m}{s}}\right\vert >{\frac {k(\epsilon )}{\sqrt {s}}}} https://encrypted-tbn1.gstatic.com/images?q=tbn:ANd9GcSmIvMtKfhpl5xZNElBkXTMACMt8ev_CWKhzKs-LEaPUPTpW9Px9RKUxae2

Con k ( ϵ → 0 ) → 0 {\displaystyle k(\epsilon \rightarrow 0)\rightarrow 0} . El último toro en destruirse es el más irracional de todos (el que guarda mayor semejanza con el [número áureo](https://es.wikipedia.org/wiki/N%C3%BAmero_%C3%A1ureo)). Informalmente el teorema establece que:

"Para perturbaciones suficientemente pequeñas, casi todos los toros invariantes se preservan [en el sistema perturbado] (excluyendo aquellos con vectores de frecuencia racionales)"[1](https://es.wikipedia.org/wiki/Teorema_de_Kolmog%C3%B3rov-Arnold-Moser#cite_note-1)