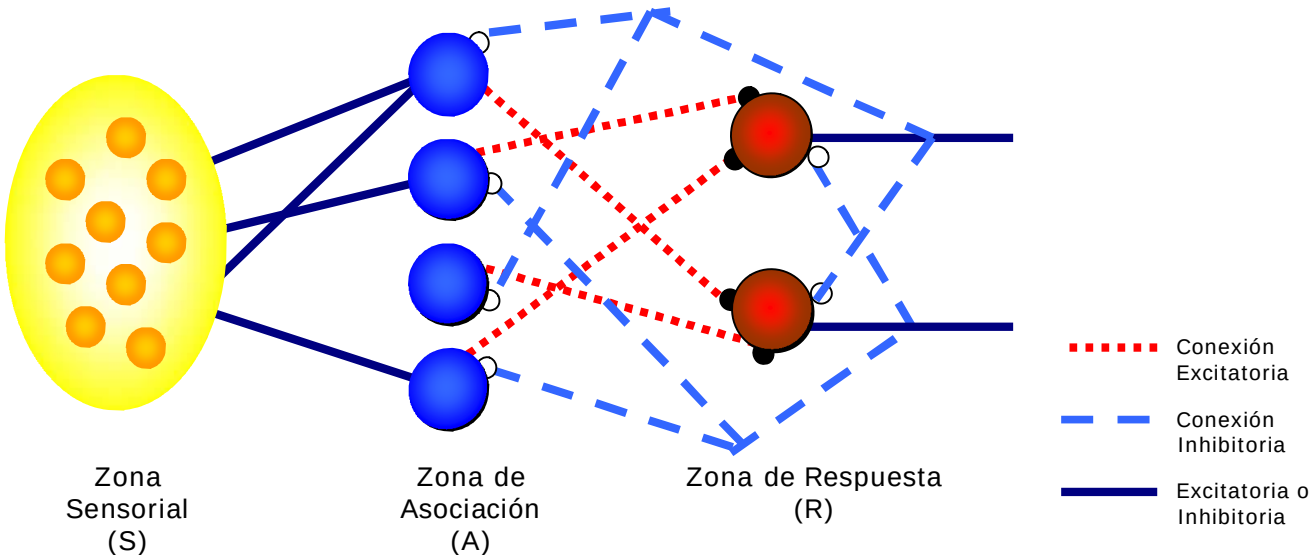
El Perceptrón

INTRODUCCIÓN

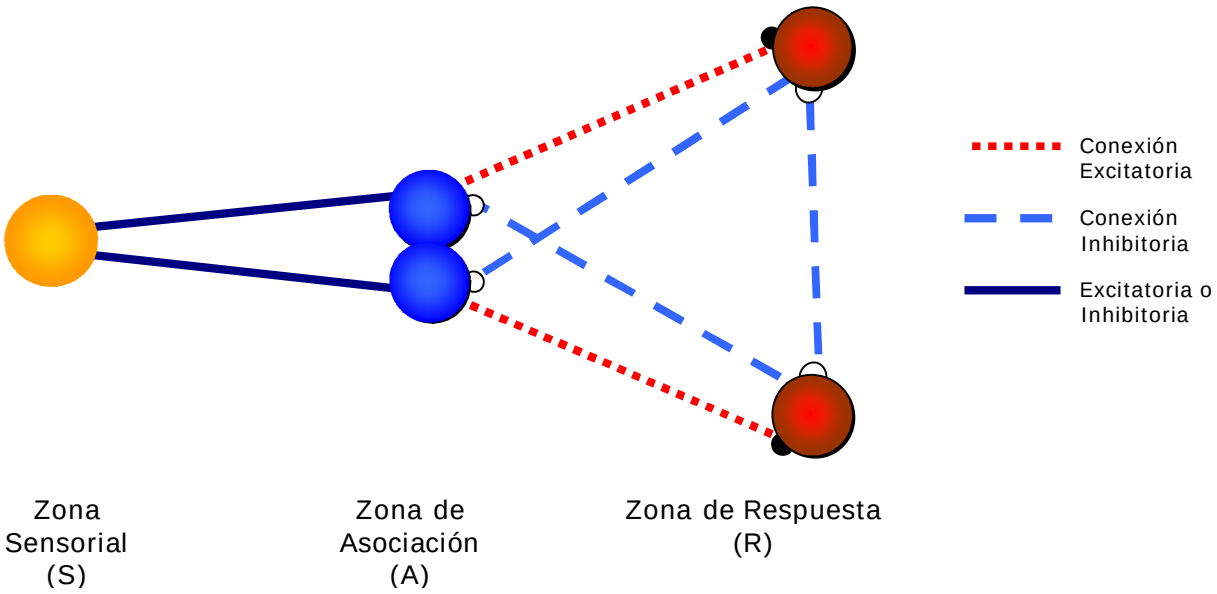
La primera red neuronal conocida, fue desarrollada en 1943 por Warren McCulloch y Walter Pitts; esta consistía en una suma de las señales de entrada, multiplicadas por unos valores de pesos escogidos aleatoriamente. La entrada es comparada con un patrón preestablecido para determinar la salida de la 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.

El primer modelo de Perceptrón fue desarrollado en un ambiente biológico imitando el funcionamiento del ojo humano, el fotoperceptrón como se le llamo era un dispositivo que respondía a señales ópticas; la luz incide en los puntos sensibles (S) de la estructura de la retina, cada punto S responde en forma todo-nada a la luz entrante, los impulsos generados por los puntos S se transmiten a las unidades de asociación (A) de la capa de asociación; cada unidad A está conectada a un conjunto aleatorio de puntos S, denominados conjunto fuente de la unidad A, y las conexiones pueden ser tanto excitatorias como inhibitorias. Las conexiones tienen los valores posibles +1, -1 y 0, cuando aparece un conjunto de estímulos en la retina, una unidad A se activa si la suma de sus entradas sobrepasa algún valor umbral; si la unidad esta activada, A produce una salida que se envía a la siguiente capa de unidades.



De forma similar, las unidades A están conectadas a unidades de respuesta (R) dentro de la capa de respuesta y la conectividad vuelve a ser aleatorio entre capas, pero se añaden conexiones inhibitorias de realimentación procedentes de la capa de respuesta y que llegan a la capa de asociación, también hay conexiones inhibitorias entre las unidades R. Todo el esquema de conexiones se describe en forma general en un diagrama de Venn, para un Perceptrón sencillo con dos unidades de respuesta.

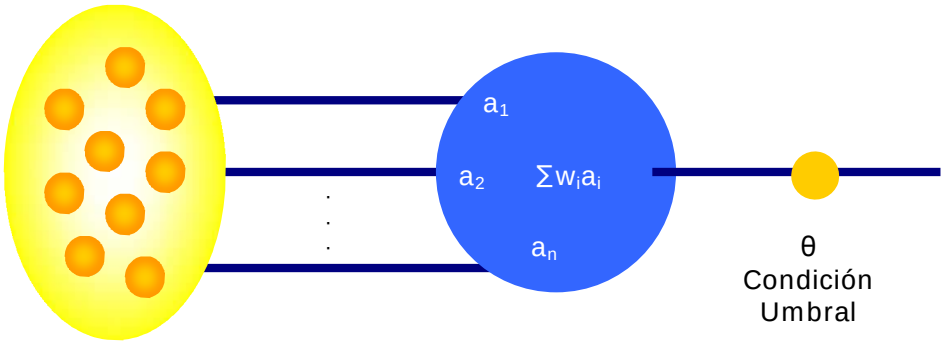


El Perceptrón era inicialmente un dispositivo de aprendizaje, en su configuración inicial no estaba en capacidad de distinguir patrones de entrada muy complejos, sin embargo mediante un proceso de aprendizaje era capaz de adquirir esta capacidad. En esencia, el entrenamiento implicaba un proceso de refuerzo mediante el cual la salida de las unidades A se incrementaba o se decrementaba dependiendo de si las unidades A contribuían o no a las respuestas correctas del Perceptrón para una entrada dada. Se aplicaba una entrada a la retina, y el estímulo se propagaba a través de las capas hasta que se activase una unidad de respuesta. Si se había activado la unidad de respuesta correcta, se incrementaba la salida de las unidades A que hubieran contribuido. Si se activaba una unidad R incorrecta, se hacía disminuir la salida de las unidades A que hubiesen contribuido.

Mediante estas investigaciones se pudo demostrar que el Perceptrón era capaz de clasificar patrones correctamente, en lo que Rosenblatt denominaba un entorno diferenciado, en el cual cada clase estaba formada por patrones similares. El Perceptrón también era capaz de responder de manera congruente frente a patrones aleatorios, pero su precisión iba disminuyendo a medida que aumentaba el número de patrones que intentaba aprender.

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.

Minsky y Papert se apartaban de la aproximación probabilística de Rosenblatt y volvían a las ideas de cálculo de predicados en el análisis del Perceptrón.



La estructura de un Perceptrón sencillo es similar a la del elemento general de procesamiento que se muestra en la figura 2.1.3; en la que se observa la adición de una condición umbral en la salida. Si la entrada neta, a esta condición es mayor que el valor umbral, la salida de la red es 1, en caso contrario es 0.

La función de salida de la red es llamada función umbral o función de transferencia

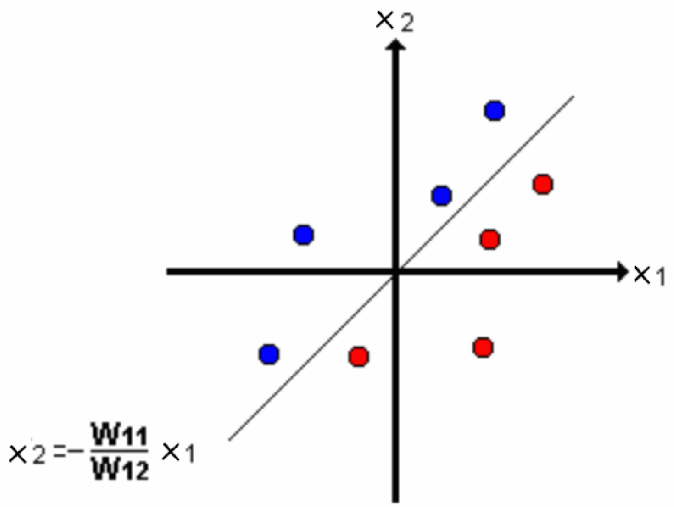
$$f(salida) = \begin{cases} 1 \; si \;\; salida \geq \theta \\ 0 \; si \;\; salida < \theta \end{cases}$$

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.

ESTRUCTURA DE LA RED

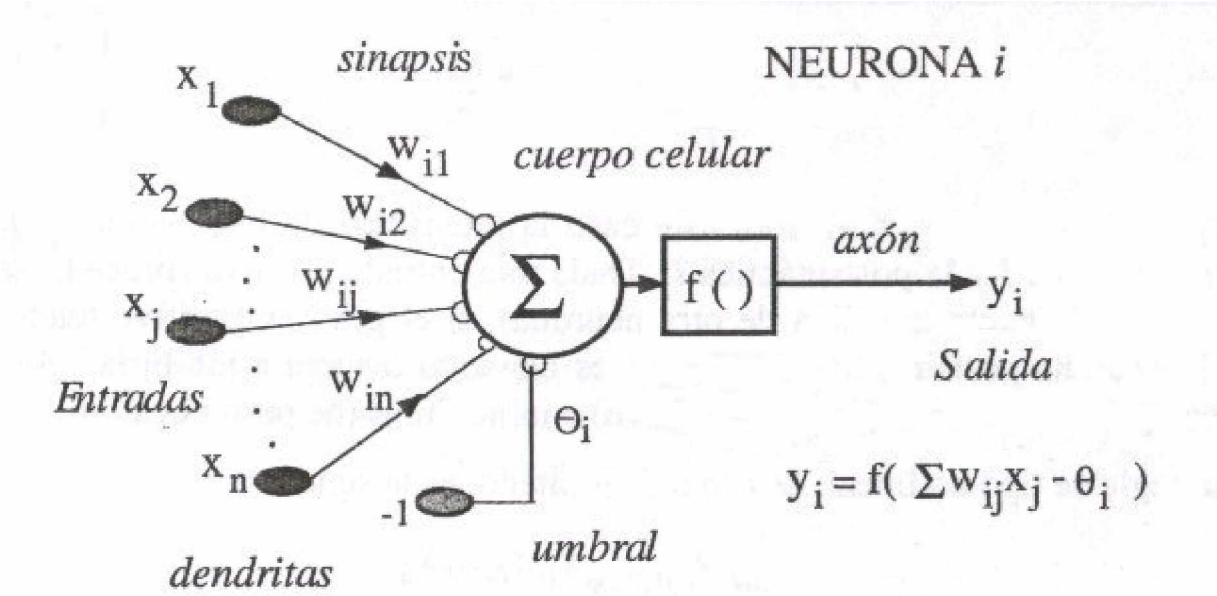
Definición

La única neurona de salida del Perceptrón realiza la suma ponderada de las entradas, resta el umbral y pasa el resultado a una función de transferencia de tipo escalón. La regla e decisión es responder +1 si el patrón presentado pertenece a la clase A, o –1 si el patrón pertenece a la clase B la salida depende de la entrada neta (*n* = suma de las entradas x_i ponderadas).



Definición Formal

Definimos " perceptrón simple" como un dispositivo de computación con umbral *U* y *n* entradas reales $x_1,\ldots, x_n$
a través de arcos con pesos $w_1,\ldots, w_n$ y que tiene salida 1 cuando $\Sigma_i w_i x_i \geq U$ y 0 en caso contrario.



Se puede simplificar la función de activación incorporando el valor umbral en el sumatorio. Basta añar a la red una unidad extra tal que:

- Siempre tiene la salida al valor 1
- Se conecta a todas las unidades de la red
- El peso de la conexión es el umbral de la neurona a la que se conecta

$$y = g\left(\sum_{i=1}^{n} w_i x_i - \theta\right) = g\left(\sum_{i=0}^{n} w_i x_i\right)$$

El origen de las entradas no es importante. Pueden proceder de otros perceptrones o de otra clase de unidades de ccmputación. La interpretación geométrica es sencilla: la superficie de separación entre las entradas que originan una respuesta 1 y las de respuesta 0, es un hiperplano que divide al espacio de entradas en dos semiespacios.

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 representan 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.

Conjuntos linealmente separables

Dos conjuntos de puntos A y B en un espacio n-dimensional son linealmente separables si existen n + 1 números reales $w_1,\ldots, w_{n+1}$ tales que cada punto $(x_1,\ldots, x_n)$ A satisface $\Sigma_i w_i x_i \geq w_{n+1}$ y que cada punto $(x_1,\ldots, x_n)$ B satisface que $\Sigma_i w_i x_i < w_{n+1}$ (el signo = puede ser en el conjunto B o en el A).
Si la desigualdad es estricta en ambos casos, se habla de Separabilidad lineal absoluta. Si los dos conjuntos A y B son finitos, ambas definiciones son equivalentes.

Para hacernos una idea de la potencia de un perceptrón como aproximador de funciones lógicas, es interesante ver cuántas son linealmente separables. Para n = 2, 14 de las 16 funciones lógicas posibles son linealmente separables (no lo son XOR y NO XOR). Cuando n = 3, 104 de las 256 funciones posibles y cuando n = 4, 1882 de las 65536. Hasta ahora no se conoce la fórmula que indica el número de funciones linealmente separables en función del número de entradas, aunque se conocen cotas.

A diferencia de las redes de McCulloch-Pitts, los perceptrones tienen cierta importancia práctica. La limitación de poder aplicarse solo a problemas separables linealmente es importante para construir con perceptrones un modelo general de computación, pero no es un inconveniente grave en aplicaciones de clasificación (los clasificadores lineales son ampliamente empleados).

Teorema de convergencia

Si las clases son linealmente separables, el algoritmo del perceptrón converge a una solución correcta en un número finito de pasos para cualquier elección inicial de pesos

APRENDIZAJE DEL PERCEPTRÓN

Como sabemos, un algoritmo de aprendizaje es un método adaptativo por el que una red de PE se automodifica para implementar el comportamiento deseado. Esto se hace presentando algunos ejemplos de la función entrada-salida a la red. Se presenta un ejemplo y se ejecuta una acción correctiva iterativamente hasta que la red aprende a producir la respuesta deseada. El conjunto de entrenamiento es el conjunto de los ejemplos de los que la red va a aprender.

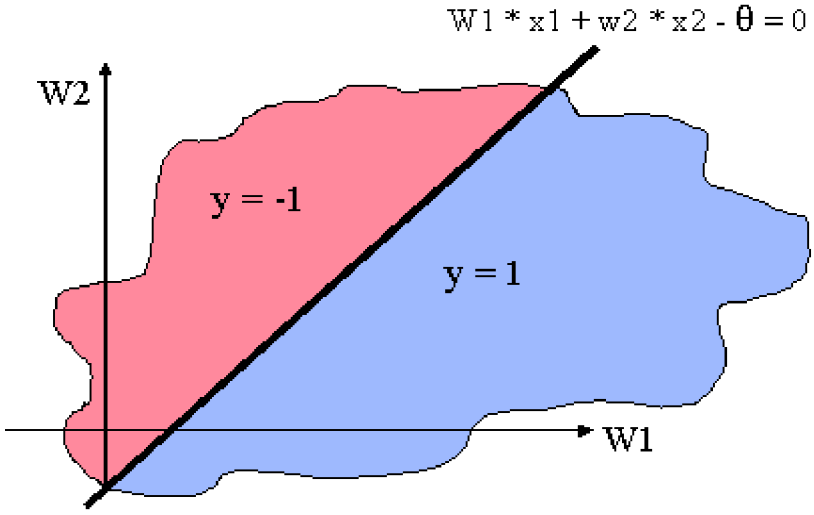
Usamos la siguiente notación: El vector de entrada al perceptron es $x = (x_1,\ldots, x_n)$. Si los pesos son los valores reales $w_1,\ldots, w_n$ y el umbral es U, diremos que $w = (w_1, \ldots, w_n, w_{n+1})$ con $w_{n+1} = – U$ es el vector extendido de

pesos del perceptrón y que (x_1,..., x_n, 1) es el vector extendido de entradas (se añade la entrada de tendencia o bias con valor 1 fijo). La computación de un perceptrón puede expresarse mediante un producto escalar, ya que $\Sigma_i w_i x_i \geq U$ es equivalente a $w \bullet x \geq 0$ donde w y x son los vectores extendidos de pesos y entradas, respectivamente.

La convergencia del algoritmo de aprendizaje del perceptrón se basa en que cada perceptrón realiza la comprobación $w \bullet x > 0$, ó $w \bullet x \geq 0$, pero son equivalentes cuando el conjunto de entrenamiento es finito, lo cual siempre es cierto en problemas prácticos. Una forma habitual de comenzar el algoritmo de entrenamiento es inicializando aleatoriamente los pesos de la red y mejorar los parámetros iniciales, comprobando a cada paso, si puede lograrse una separación mejor del conjunto de entrenamiento. Cada vector de pesos w define un hiperplano que separa los puntos con salida 1 ($w \bullet x \geq 0$) de los puntos con salida 0 ($w \bullet x \leq 0$).

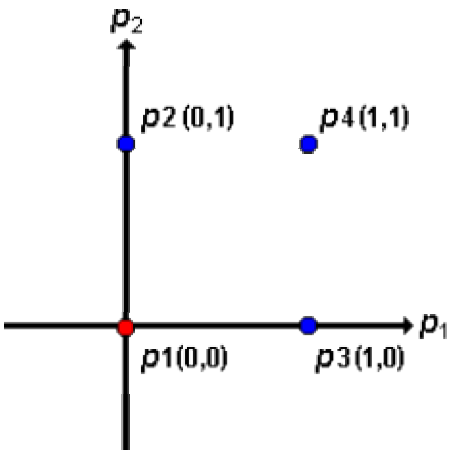
Interpretación gráfica o geométrica del perceptrón

La ecuación de evaluación de las salidas representa un plano (un hiperplano en dimensión n) que divide el espacio en dos partes. El entrenamiento consiste en buscar el plano que hace la división adecuada, para que los puntos cuya salida es 1 queden todos en la misma región, y los que tienen salida esperada 0, queden todos en la otra región. Sean P y N dos conjuntos finitos de puntos en R_n que queremos separar linealmente. Se desea encontrar un vector w tal que su hiperplano asociado separe los dos conjuntos (salida 1 para P y 0 para N). El error de un perceptron con un vector de pesos w es el número de puntos incorrectamente clasificados. El algoritmo de entrenamiento debe minimizar este error E(w). Una de las posibles estrategias es la de usar un algoritmo voraz local que calcule el error del perceptrón para un vector de pesos dado, buscando una dirección en el espacio de pesos en la que moverse y actualizandolos, seleccionando para ello nuevos valores de acuerdo con la dirección de búsqueda.



EJEMPLO

Como ejemplo de funcionamiento de una red neuronal tipo Perceptrón, se solucionará el problema de la función OR, para esta función la red debe ser capaz de devolver a partir de los cuatro patrones de entrada, a qué clase pertenece cada uno; es decir para el patrón 00 debe devolver la clase cero y para los restantes la clase 1.



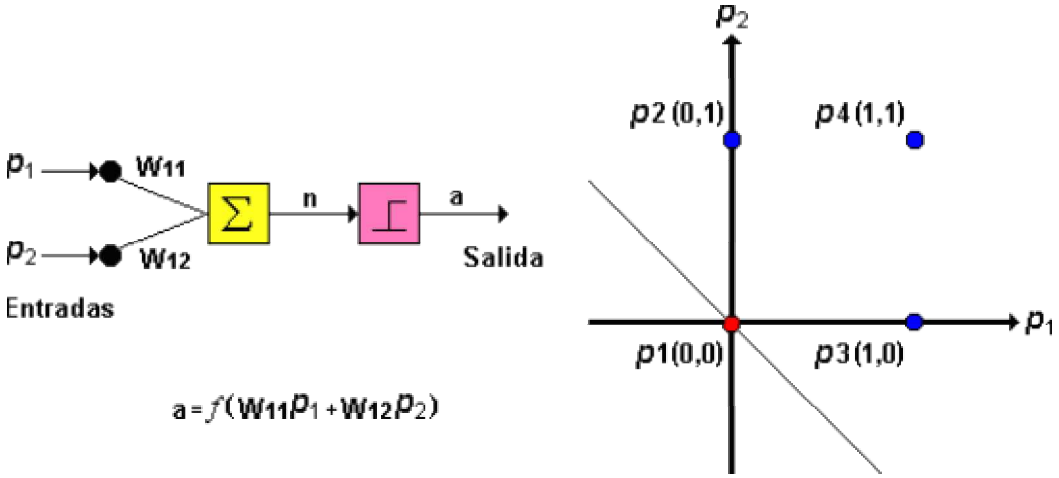
p_1	p_2	a
0	0	0
0	1	1
1	0	1
1	1	1

Función OR

Para este caso las entradas a la red serán valores binarios, la salida de la red esta determinada por

$$y = f(w1x1+w2x2)$$

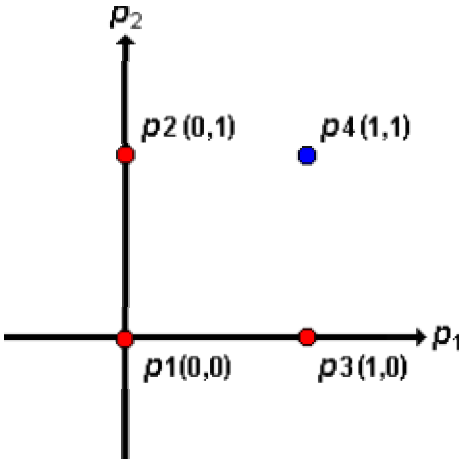
Si $w_1x_1+w_2x_2$ es mayor que 0 la salida será 1, en caso contrario la salida será –1 (función escalón unitario). Como puede verse la sumatoria que se le pasa a cada parámetro (entrada total) a la función *escalon 0 - 1*(función de salida o de transferencia) es la expresión matemática de una recta, donde w_1 y w_2 son variables y x_1 y x_2 son constantes. En la etapa de aprendizaje se irán variando los valores de los pesos obteniendo distintas rectas, lo que se pretende al modificar los pesos de las conexiones es encontrar una recta que divida el plano en dos espacios de las dos clases de valores de entrada, concretamente para la función OR se deben separar los valores 01, 10, y 11 del valor 00.



Perceptrón aplicado a la función OR

Puede verse como las posibles rectas pasarán por el origen de coordenadas, por lo que la entrada 00 quedará sobre la propia recta.

Se aplicará este método para resolver también el problema de la función AND, el cual se describe en la siguiente figura

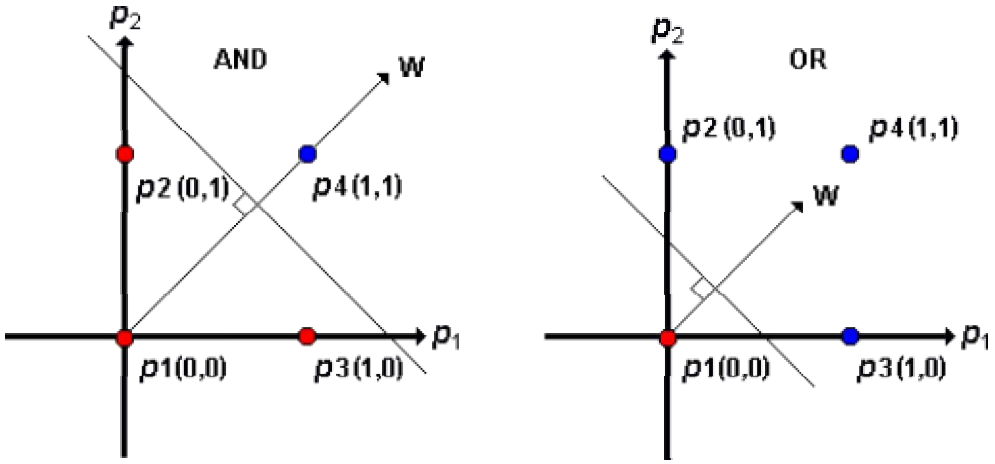


Espacio de salida de una compuerta AND

Analizando el comportamiento de la AND se llega a la conclusión de que es imposible que una recta que pase por el origen, separe los valores 00,01 y 10 del valor 11, por lo que se hace necesario introducir un término independiente para realizar esta tarea, a este término se le da el nombre de ganancia y se representa por la letra b, al cual por lo general se le asigna un valor inicial de 1 y se ajusta durante la etapa de aprendizaje de la red; este nuevo término permite desplazar la recta del origen de coordenadas dando una solución para el caso de la función AND y ampliando el número de soluciones de la función OR

Ahora la salida de la neurona esta dada por

$$y = f(w1x1+w2x2 + b)$$



Solución para una función AND y una OR

Al goritmo de Aprendizaje.

Realmente, se ha llamado Perceptron a una RNA donde todas las entradas están conectadas a varias salidas, que se calculan como se ha indicado antes. Entonces la separabilidad lineal hay que mirarla en cada salida. El algoritmo de entrenamiento del perceptron es supervisado y usa n vectores que sepueden

clasificar en dos conjuntos P y N en el espacio extendido n+ 1-dimensional. Se considera el umbral como una entrada de tendencia con valor fijo 1, ya que Σiwi xi < wn+1 equivale a
Σiwi xi - 1 wn+1 < 0

Buscamos un vector w capaz de separar absolutamente ambos conjuntos. El algoritmo es (para cada salida):

ALGORITMO DE ENTRENAMIENTO DEL PERCEPTRON

Iniciar: w0 aleatoriamente ; t ← 0
Repetir Mientras queden vectores mal clasificados
seleccionar aleatoriamente un vector x
si x ϵ P (salida esperada 1) y wt • x < 0 (salida calculada 0) entonces
[Hay un error]
wt+1 ← wt + x ; t ← t+1
fin si
si x ϵ N (salida esperada 0) y wt • x ≥ 0 (salida calculada 1) entonces
[Hay un error]
wt+1 ← wt – x ; t ← t+1
fin si
fin repite

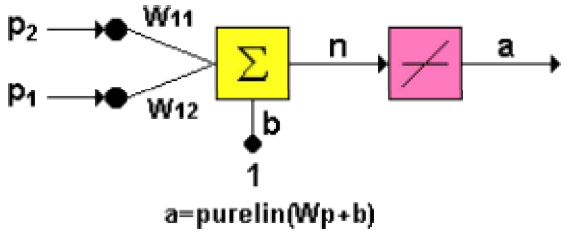
Otra forma de escribir el algoritmo, en forma esquemática, para varias salidas.
Notamos los ejemplos de entrenamiento (X, d), con X la entrada y d la salida esperada.
0.- Fijar los pesos para las entradas Wij i = 1..p, j = 1..q
1.- Repetir pasos 2-4 Mientras existan ejemplos mal clasificados
2.- Seleccionar otro elemento de entrenamiento (X, d) mientras haya.
3.- Introducir X en red, calcular la salida Y : Yj = 1 si Σi
Xi Wij > U y 0 en otro caso
4.- [Comparar Y con d (salida esperada)]
Para j = 1..q
Si salida j es incorrecta , Entonces ; W'ij = Wij - (dj –Yj) Xi ; FinSi
FinPara [para cada salida (j) no correcta, Si Yj = 0 [x ϵ P ; ΣiXi Wij < U] , W'ij = Wij + Xi]
[Si Yj =1 [x ϵ N ; ΣiXi Wij > U], W'ij = Wij - Xi]
5.- FIN

El Adaline

Introducción

Al mismo tiempo que Frank Rosenblatt trabajaba en el modelo del Perceptrón Bernard Widrow y su estudiante Marcian Hoff introdujeron el modelo de la red Adaline y su regla de aprendizaje llamada algoritmo LMS (Least Mean Square).

La red Adaline es similar al Perceptrón, excepto en su función de transferencia, la cual es una función de tipo lineal en lugar de un limitador fuerte como en el caso del Perceptrón. La red Adaline presenta la misma limitación del Perceptrón en cuanto al tipo de problemas que pueden resolver, ambas redes pueden solo resolver problemas linealmente separables, sin embargo el algoritmo LMS es más potente que la regla de aprendizaje del Perceptrón ya que minimiza el error medio cuadrático, la regla sirvió de inspiración para el desarrollo de otros algoritmos, este es el gran aporte de esta red.



Adaline de una neurona y dos entradas

El término Adaline es una sigla, sin embargo su significado cambió ligeramente a finales de los años sesenta cuando decayó el estudio de las redes neuronales, inicialmente se llamaba ADAptive LInear NEuron (Neurona Lineal Adaptiva), para pasar después a ser Adaptive LInear Element (Elemento Lineal Adaptivo), este cambio se debió a que la Adaline es un dispositivo que consta de un único elemento de procesamiento, como tal no es técnicamente una red neuronal.

El elemento de procesamiento realiza la suma de los productos de los vectores de entrada y de pesos, y aplica una función de salida para obtener un único valor de salida, el cual debido a su función de transferencia lineal será +1 si la sumatoria es positiva o –1 si la salida de la sumatoria es negativa. En términos generales la salida de la red está dada por

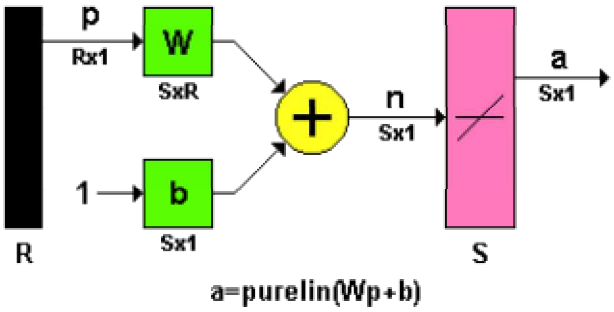
$$Y = W^t x$$

En este caso, la salida es la función unidad al igual que la función de activación; el uso de la función identidad como función de salida y como función de activación significa que la salida es igual a la activación, que es la misma entrada neta al elemento.

El Adaline es ADaptivo en el sentido de que existe un procedimiento bien definido para modificar los pesos con objeto de hacer posible que el dispositivo proporcione el valor de salida correcto para la entrada dada; el significado de correcto para efectos del valor de salida depende de la función de tratamiento de

señales que esté siendo llevada a cabo por el dispositivo. El Adaline es LIneal porque la salida es una función lineal sencilla de los valores de la entrada. Es una NEurona tan solo en el sentido (muy limitado) del elemento de procesamiento. También se podría decir que el Adaline es un Elemento Lineal, evitando por completo la definición como NEurona.

ESTRUCTURA DE LA RED



Estructura de una red Adaline

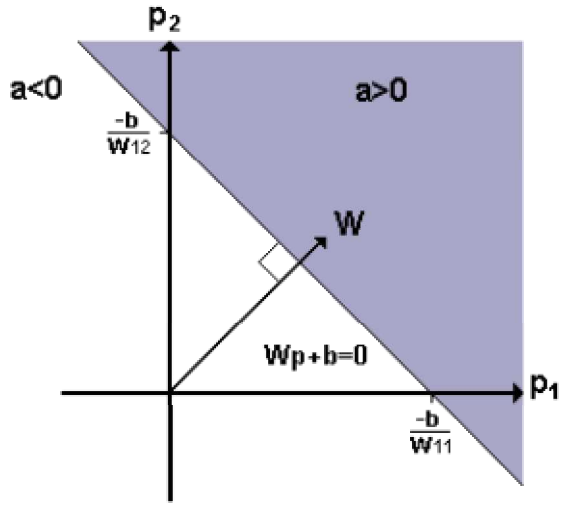
La salida de la red está dada por:

$$a = purelin(Wp + b) = Wp + b$$

En similitud con el Perceptrón, el límite de la característica de decisión para la red Adaline se presenta cuando n = 0, por lo tanto:

$$\boldsymbol{w}^T \boldsymbol{p} + b = 0$$

específica la línea que separa en dos regiones el espacio de entrada



Característica de decisión de una red tipo Adaline

La salida de la neurona es mayor que cero en el área gris, en el área blanca la salida es menor que cero. Como se mencionó anteriormente, la red Adaline puede clasificar correctamente patrones linealmente separables en dos categorías.

REGLA DE ENTRENAMIENTO.

Sea un conjunto de entrenamiento $\{x_1, ..., x_n\}$ cada uno con su salida (esperada) di . El entrenamiento consiste en encontrar un único conjunto de pesos w que permita que la Red calcule el valor esperado para todos los vectores de entrada, o al menos el que mejor consigue este objetivo. Esto es muy fácil de realizar para un solo vector de entrada.

En primer lugar, entenderemos que un conjunto de pesos w^* es el mejor , si minimiza las diferencias entre las salidas calculadas y las esperadas, para todos los ejemplos de entrenamiento. La medida de estas diferencias se hace con la media (esperada) de los cuadrados de los errores para cada ejemplo de entrada (Error cuadrático medio,ECM),

$$ECM = \frac{1}{n}\sum_{i=1}^{n}\left(d_i - \mathrm{w}^t \mathrm{x}_i\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\; d_i^2 + \mathrm{w}^t \mathrm{R} \mathrm{w} - 2\mathrm{p}^t \mathrm{w}\right)$$

donde R =xt x (correlación de entradas) y p = d_i x_i (t indica traspuesta en notación matricial). Si derivamos la ecuación, para obtener el mínimo, respecto de los pesos, tendremos 2Rw* - 2p = 0, y Rw* = p de donde w* = R-1 p
Entonces, vemos que existe un punto donde la pendiente de la función ECM(w) se anula, aunque no tenemos la seguridad de que este punto sea un mínimo (puede ser un máximo). En este proceso se supone la estabilidad estadística de las entradas, lo que básicamente equivale a que los valores esperados cambian muy lentamente en el tiempo.

Si suponemos un combinador lineal con solo dos pesos, el ECM toma la forma de un paraboloide dirigido hacia
arriba (ya que todas las combinaciones de pesos dan un valor no negativo para el ECM). Esto es generalizado para cualquier número n de pesos, donde la figura es un hiperparaboloide (en un espacio n+1-dimensional). Por lo tanto, tendremos un punto de mínimo cuando la pendiente sea 0.

Muchas veces, el cálculo directo de ese punto no es fácil, y se usan métodos de optimización no lineal, por ejemplo el método del gradiente descendente mas pronunciado (Steepest Descent).

Inicialmente se asignan pesos aleatorios w(0) y se determina la dirección de la pendiente más pronunciada en dirección al óptimo (descendiendo hacia el mínimo de la superficie). Normalmente el vector de pesos no se desplaza directamente hacia el mínimo (porque la sección del paraboloide suele ser elíptica). En cada iteración calculamos los nuevos pesos, modificando los de la iteración anterior $w(t+1) = w(t) + \Delta w(t)$. Para calcular la dirección adecuada, necesitamos el gradiente de la superficie, y para controlar que el avance no sea excesivamente grande (nos puede desviar bastante de nuestro propósito) se añade un factor η : $w(t+1) = w(t) - \eta \, ECM(w(t))$

Lo que hace falta es calcular este gradiente en cada iteración. Como el cálculo directo suele ser complejo, se usa una aproximación siguiendo los pasos:

1.- Repetir pasos 2-4 para los ejemplos, hasta que el error sea suficientemente pequeño
2.- Se aplica un vector de entrada x_l.
3.- Se calcula el EC con los pesos actuales y su gradiente (basta con obtener $\delta l(t) = d_l - w^t(t)x_l$)
4.- Se actualiza el vector de pesos con $w(t+1) = w(t) + \eta \, \delta_l(t) \, x_l$ (η pequeña)
5.- Fin.

La expresión del paso 5 resume el algoritmo llamado Aprendizaje por mínimos cuadrados, donde el factor η fija la velocidad de convergencia y la estabilidad del algoritmo, ya que para que se verifiquen las condiciones de convergencia, se debe cumplir que las modificaciones de los pesos sean muy pequeñas en cada paso.

La notación $\delta(t) = d(t) - y(t)$ le da el nombre de regla delta. La expresión se obtiene a partir de la función de error, derivando con la regla de la cadena.

APLICACIONES DE LA RED ADALINE

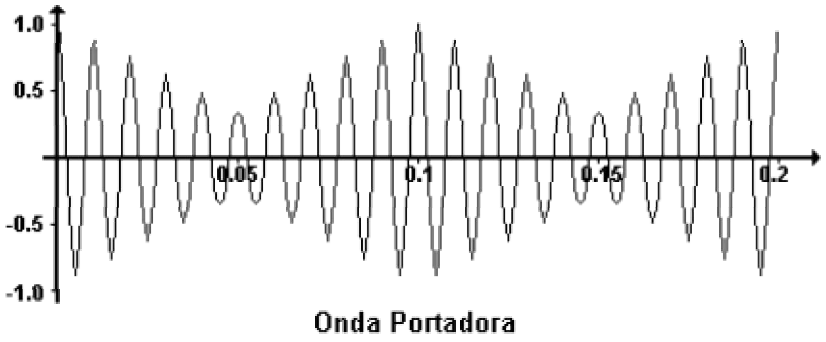
La red Adaline ha sido ampliamente utilizada en el procesamiento de señales; para valorar el real aporte de esta red en ese campo, se detallarán un poco las herramientas hasta ahora empleadas en los procesos de filtrado.

A comienzos del estudio de las comunicaciones electrónicas, se diseñaban filtros analógicos empleando circuitos RLC (Resistencias, Inductores, Condensadores) para eliminar el ruido en las señales empleadas de comunicaciones; este procesamiento se ha transformado en una técnica de múltiples facetas, destacándose en la actualidad el uso de procesadores digitales de señales (DSP), que pueden llevar a cabo los mismos tipos de aplicaciones de filtrado ejecutando filtros de convolución realizados mediante programación convencional, en cualquier lenguaje de programación conocido.

El proceso de filtrado sigue ocupando un lugar muy importante en la industria, pues siempre será necesario eliminar el ruido en señales portadoras de información. Considérese una transmisión de radio en AM, las técnicas electrónicas de comunicación, bien sean para señales de audio o de datos constan de una codificación y una modulación de la señal. La información que hay que transmitir, se puede codificar en forma de una señal analógica que reproduce exactamente las frecuencias y las amplitudes del sonido original. Dado que los sonidos que se están codificando representan un valor continuo que va desde el silencio, pasando por la voz, hasta la música, la frecuencia instantánea de la señal variará con el tiempo, oscilando entre 0 y 10.000 Hz aproximadamente.

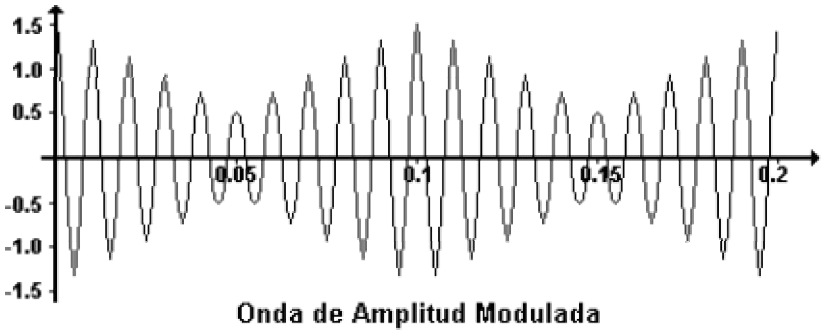
En lugar de intentar transmitir directamente esta señal codificada, se transmite la señal en forma más adecuada para la transmisión por radio; esto se logra modulando la amplitud de una señal portadora de alta frecuencia con la señal de información analógica. Para la radio AM, la frecuencia portadora estará en el intervalo de los 550 a los 1650 kHz , dado que la frecuencia de la portadora es muy superior a la frecuencia máxima de la señal de información, se pierde muy poca información como consecuencia de la modulación; la señal modulada puede ser transmitida después a una estación receptora (o se puede retransmitir a cualquiera que tenga un receptor de radio), en la cual la señal se demodula y se reproduce en forma de sonido.

La razón más evidente para utilizar un filtro en una radio de AM es que cada persona tiene sus preferencias de música y diversión y dado que hay tantas emisoras de radio diferentes es necesario permitir que cada usuario sintonice su receptor a una cierta frecuencia seleccionable. Al sintonizar la radio, lo que se esta haciendo es, modificar las características de respuesta en frecuencia de un filtro *pasa banda* que está dentro de la radio, este filtro solo deja pasar las señales procedentes de la emisora en la que se este interesado y elimina todas las demás señales que estén siendo transmitidas dentro del espectro AM.



Onda Portadora

Onda de Información



Onda de Amplitud Modulada

Técnicas de codificación de información y modulación en amplitud

La herramienta matemática para el diseño de filtros más utilizada es la Serie de Fourier, que describe la naturaleza de las señales periódicas en el dominio frecuencial y viene dada por:

$$x(t) = \sum_{n=0}^{\infty} a_n Cos(2\pi n f_0 t) + \sum_{n=1}^{\infty} b_n Sen(2\pi n f_0 t)$$

En donde:

f_0: Frecuencia fundamental de la señal en el dominio del tiempo
a_n y b_n: Coeficientes necesarios para modular la amplitud de los términos individuales de la serie.

Las primeras realizaciones de los cuatro filtros básicos poseían una gran limitación: Solo eran ajustables en un pequeño intervalo

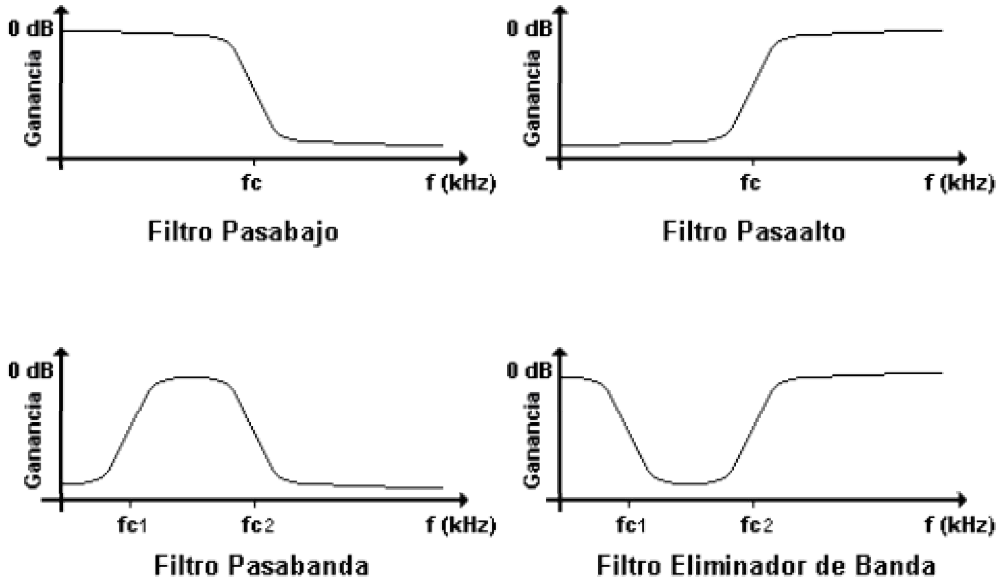


Figura 2.2.6 Características de los cuatro filtros básicos

Todos los filtros se pueden caracterizar a partir de su respuesta $h(n)$ a la función de impulso unitario, que se representa por $\delta(n)$ en la forma:

$$h(n) = R[\delta(n)]$$

La ventaja de esta formulación es que una vez se conoce la respuesta del sistema para el impulso unitario, la salida del sistema para cualquier entrada está dada por

$$y(n) = R[x(n)] = \sum_{i=-\infty}^{\infty} h(i)\, x(n-i)$$

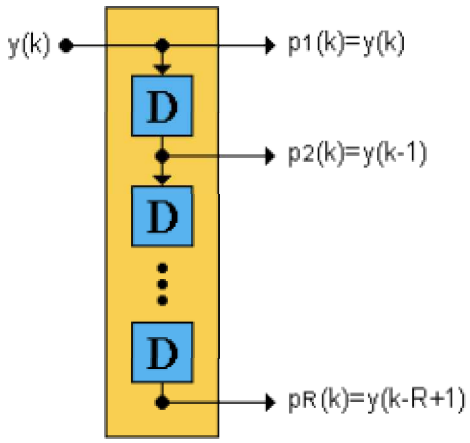
Donde $x(n)$ es la entrada al sistema

Esta ecuación describe una convolución entre la señal de entrada y la respuesta del sistema al impulso unitario. Para este caso, basta tener en cuenta que la convolución es una operación de suma entre productos, similar al tipo de operación que realiza un Perceptrón cuando calcula su señal de activación. La red Adaline emplea este mismo calculo para determinar cuanta estimulación de entrada recibe a partir de una señal instantánea de entrada; esta red tiene diseñado en su interior una forma de adaptar los coeficientes ponderables (pesos de la red) para hacer aumentar o disminuir la estimulación que recibirá la próxima vez que se le presente la misma señal. La utilidad de esta capacidad se pone de manifiesto cuando se diseña un filtro digital por medio de software; con un programa normal, el programador debe saber exactamente como se específica el algoritmo de filtrado y cuales son los detalles de las características de las señales; si se necesitaran modificaciones, o si cambian las características de la señal, es necesario reprogramar; cuando se emplea una red tipo Adaline, el problema se convierte, en

que la red sea capaz de especificar la señal de salida deseada, dada una señal de entrada específica.

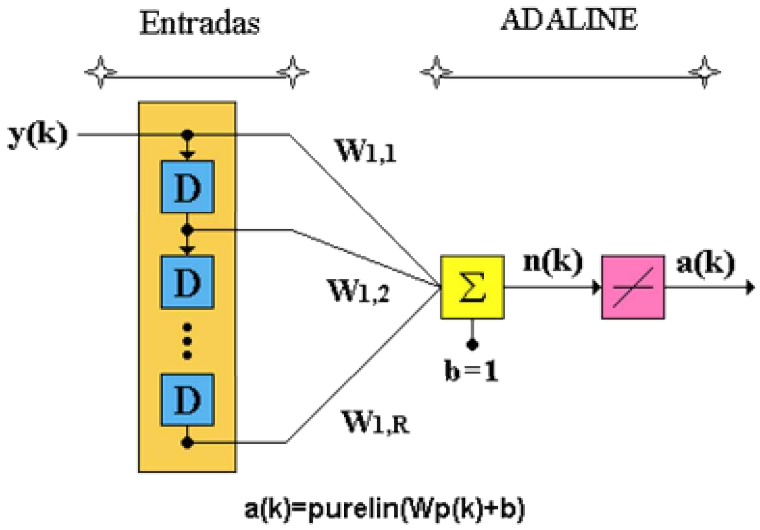
La red Adaline toma la entrada y la salida deseada, y se ajusta a sí misma para ser capaz de llevar a cabo la transformación deseada. Además, si cambian las características de la señal, la red Adaline puede adaptarse automáticamente.

En orden a usar la red tipo Adaline para implementar un filtro adaptivo, se debe incorporar el concepto de retardos en línea.



Retardos en línea

Si se combina la red Adaline con un bloque de retardos en línea, se ha creado un filtro adoptivo.



Filtro adoptivo

Cuya salida está dada por:

$$a(k) = purelin = (\boldsymbol{Wp} + b) = \sum_{i=1}^{R} w_{1,i}\, y(k-i+1) + b$$