2.2 ADALINE

2.2.1 Antecedentes. 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 sólo 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, característica que lo hace bastante práctico en las aplicaciones de procesamiento de señales digitales, por ejemplo las líneas telefónicas de gran distancia utilizan la red Adaline para cancelar el ruido inherente a su recorrido.

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

$$\boldsymbol{a} = \boldsymbol{W}^{T}\boldsymbol{p} \qquad (2.2.1)$$

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 **AD**aptivo 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 **LI**neal porque la salida es una función lineal sencilla de los valores de la entrada. Es una **NE**urona tan solo en el sentido (muy limitado) del PE. También se podría decir que el Adaline es un Elemento Lineal, evitando por completo la definición como **NE**urona

2.2.2 Estructura de la red. La estructura general de la red tipo Adaline puede visualizarse en la figura 2.2.1

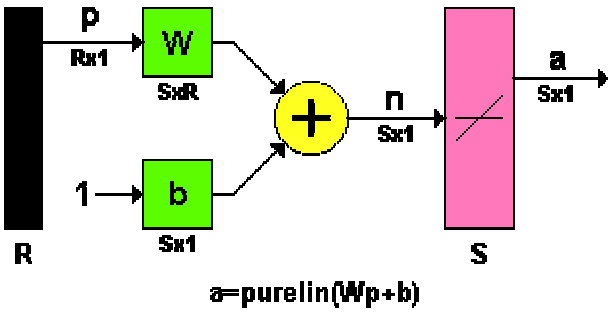

Figura 2.2.1 Estructura de una red Adaline

La salida de la red está dada por:

$$a = purelin(Wp + b) = Wp + b \qquad (2.2.2)$$

Para una red Adaline de una sola neurona con dos entradas el diagrama corresponde a la figura 2.2.2

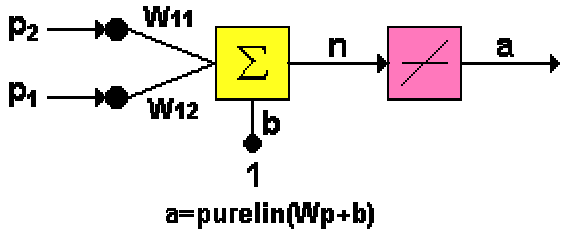

Figura 2.2.2 Adaline de una neurona y dos entradas

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

$$\boldsymbol{w}^T \boldsymbol{p} + b = 0 \qquad (2.2.3)$$

específica la línea que separa en dos regiones el espacio de entrada, como se muestra en la figura 2.2.3

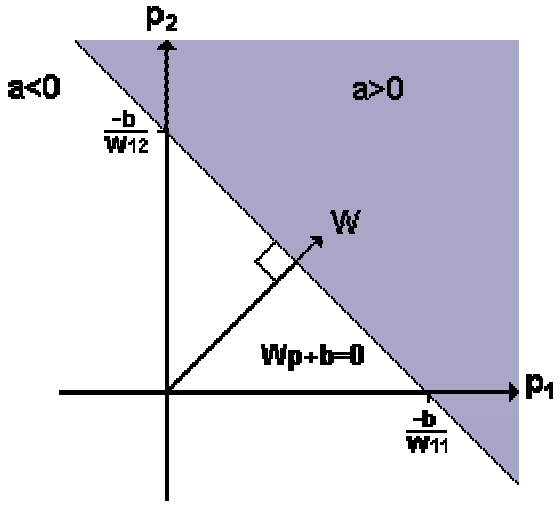

Figura 2.2.3. Característica de decisión de una red tipo Adaline

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.

2.2.3 Regla de aprendizaje. Al igual que el Perceptrón, la red Adaline es una red de aprendizaje supervisado que necesita conocer de antemano los valores asociados a cada entrada. Los pares de entrada/salida tienen la siguiente forma:

$$\{p_1, t_1\}, \{p_2, t_2\}, \ldots, \{p_Q, t_Q\} \qquad (2.2.4)$$

Donde p_Q es la entrada a la red y t_Q es su correspondiente salida deseada, cuando una entrada p es presentada a la red, la salida de la red es comparada con el valor de t que le es asociado.

El algoritmo LMS se deriva de la regla Widrow-Hoff delta, la que en términos generales para un proceso de actualización de los pesos de una red Adaline, se deduce de la siguiente manera:

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + \alpha \frac{e(k)\boldsymbol{p}(k)}{|\boldsymbol{p}(k)|^2} \qquad (2.2.5)$$

En el cual k representa la iteración actual del proceso de actualización, $\boldsymbol{W}(k+1)$ es el siguiente valor que tomará el vector de pesos y $\boldsymbol{W}(k)$ es el valor actual del vector de pesos. El error actual $e(k)$ es definido como la diferencia entre la respuesta deseada $\boldsymbol{t}(k)$ y la salida de la red $\boldsymbol{a}(k) = \boldsymbol{W}^{\boldsymbol{T}}(k)\boldsymbol{p}(k)$ antes de la actualización:

$$e(k) \equiv \boldsymbol{t}(k) - \boldsymbol{W}^{\boldsymbol{T}}(k)\boldsymbol{p}(k) \qquad (2.2.6)$$

La variación del error en cada iteración es representada por

$$\Delta e(k) = \Delta\left(\boldsymbol{t}(k) - \boldsymbol{W}^{T}(k)\boldsymbol{p}(k)\right) = -\boldsymbol{p}^{T}(k) * \boldsymbol{W}(k) \qquad (2.2.7)$$

En concordancia con la ecuación (2.2.5) la actualización de los pesos, teniendo en cuenta el error es:

$$\Delta W(k) = \boldsymbol{W}(k+1) - \boldsymbol{W}(k) = \alpha \frac{e(k)\boldsymbol{p}(k)}{\left|\boldsymbol{p}(k)\right|^{2}} \qquad (2.2.8)$$

Combinando las ecuaciones (2.2.8) y (2.2.7), se obtiene:

$$\Delta e(k) = -\alpha \frac{e(k)\boldsymbol{p}^{T}(k)\boldsymbol{p}(k)}{\left|\boldsymbol{p}(k)\right|^{2}} = -\alpha\ e(k) \qquad (2.2.9)$$

De esta forma, el error es reducido por un factor α mientras los pesos van cambiando a medida que se presenta un valor de entrada. Cada vez que se presenta un nuevo patrón el ciclo de actualización inicia nuevamente; el siguiente error es reducido por un factor α, y el proceso continua. Los valores iniciales del vector de pesos son usualmente escogidos como cero y se actualizan hasta que el algoritmo alcance convergencia.

La elección de α controla la estabilidad y velocidad de la convergencia del proceso de entrenamiento; si se escoge un valor muy pequeño de α ,el algoritmo

pierde velocidad y tarda mucho en alcanzar convergencia, si por el contrario se toma un valor muy grande, el algoritmo pierde estabilidad y se torna oscilante alrededor del valor de convergencia. Para patrones de entrada independientes en el tiempo, la estabilidad es garantizada para valores de α que varíen entre

$$0 < \alpha < 2 \qquad (2.2.10)$$

Si se fija α en un valor mayor a 1 el error es innecesariamente sobre-corregido, por lo tanto un rango de valores prácticos para la rata de aprendizaje es:

$$0.1 < \alpha < 1 \qquad (2.2.11)$$

Este algoritmo es auto-normalizado en el sentido que la elección de α no depende de la magnitud de las señales de entrada; cada peso actualizado es colineal con los parámetros de entrada y su magnitud es inversamente proporcional a $\left|p(k)\right|^2$. Si se emplea como entradas binarias 1 y 0, la actualización no ocurre para pesos cuya entrada sea cero, mientras con entradas binarias ± 1 todos los pesos son actualizados en cada iteración y la convergencia es más rápida. Por esta razón, las entradas simétricas +1 y –1 son generalmente preferidas.

Una descripción geométrica del proceso de actualización de pesos en la regla Widrow-Hoff delta o algoritmo LMS, se describe en la figura 2.2.4

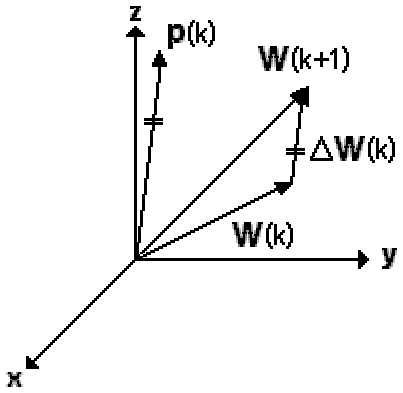

Figura 2.2.4 Actualización de pesos del algoritmo LMS

De acuerdo a la ecuación (2.2.8), $\boldsymbol{W}(k+1)$equivale la suma de $\boldsymbol{W}(k)$ y $\Delta\boldsymbol{W}(k)$, además $\Delta\boldsymbol{W}(k)$ es paralelo con el vector de entrada $\boldsymbol{p}(k)$. De la ecuación (2.2.7), el cambio en el error es igual al producto negativo de $\boldsymbol{p}(k)$ y $\Delta\boldsymbol{W}(k)$, como el algoritmo LMS selecciona a $\Delta\boldsymbol{W}(k)$ de tal forma que sea colineal con $\boldsymbol{p}(k)$, el cambio en el error deseado se calcula con la menor magnitud de $\Delta\boldsymbol{W}(k)$ posible.

Extendiendo el algoritmo a la actualización de las ganancias, se tiene:

$$b(k+1) = b(k) + \alpha\, e(k) \qquad (2.2.12)$$

El algoritmo LMS corrige el error y si todos los patrones de entrada son de igual longitud, la actualización de pesos y ganancias tiende a minimizar el error medio cuadrático, esta es la principal propiedad de este algoritmo.

En el algoritmo LMS, los valores de los incrementos $\Delta \boldsymbol{W}(k)$ y $\Delta \boldsymbol{b}(k)$ se calculan con base en las derivadas parciales de la función del error medio cuadrático con respecto a pesos y ganancias respectivamente.

Para explicar el cálculo del error medio cuadrático se considerará una red Adaline de una sola neurona y se empleará un algoritmo de pasos descendientes aproximado, como el que utilizaron Widrow y Hoff. La función de error es una función matemática definida en el espacio de pesos multidimensional para un conjunto de patrones dados, es una superficie que tendrá muchos mínimos (globales y locales) y la regla de aprendizaje va a buscar el punto en el espacio de pesos donde se encuentra el mínimo global de esa superficie; aunque la superficie de error es desconocida, el método de gradiente descendiente consigue obtener información local de dicha superficie a través del gradiente, con esa información se decide qué dirección tomar para llegar hasta el mínimo global de dicha superficie.

Con este algoritmo calculando el gradiente en cada iteración (gradiente instantáneo) y no el gradiente sobre el error total después de haber presentado todos los patrones, la función para el error medio cuadrático es:

$$e^2(k) = (t(k) - a(k))^2 \qquad (2.2.13)$$

En la ecuación 2.2.13 $t(k)$ representa la salida esperada en la iteración k y $a(k)$representa la salida de la red; el error cuadrático esperado ha sido reemplazado por el error cuadrático en la iteración k, por lo tanto en cada iteración se tiene un gradiente del error de la siguiente forma:

$$[\nabla e^2(k)]_j = \frac{\partial e^2(k)}{\partial w_{i,j}} = 2e(k)\frac{\partial e(k)}{\partial w_{1,j}} \; para \; j = 1,2,...,R \qquad (2.2.14)$$

y

$$\left[\nabla e^2(k)\right]_{R+1} = \frac{\partial e^2(k)}{\partial b} = 2e(k)\frac{\partial e(k)}{\partial b} \qquad (2.2.15)$$

Los primeros R elementos del error son derivadas parciales con respecto a los pesos de la red, mientras que los elementos restantes son derivadas parciales con respecto a las ganancias

Se evaluará primero la derivada parcial de $e(k)$con respecto a $w_{i,j}$:

$$\frac{\partial e(k)}{\partial w_{i,j}} = \frac{\partial\left[t(k) - ({}_1\boldsymbol{w}^{\boldsymbol{T}} * \boldsymbol{p}(k) + b)\right]}{\partial w_{i,j}}$$

$$= \frac{\partial\left[t(k) - \left[\sum_{i=1}^{R} w_{1,i}\, p_i(k) + b\right]\right]}{\partial w_{i,j}} \qquad (2.2.16)$$

Donde $p_i(k)$ es el i-ésimo elemento del vector de entrada en la k-ésima iteración, esto puede simplificarse así:

$$\frac{\partial e(k)}{\partial w_{i,j}} = -p_j(k) \qquad (2.2.17)$$

De manera similar se obtiene el elemento final del gradiente, correspondiente a la derivada parcial del error con respecto a la ganancia:

$$\frac{\partial e(k)}{\partial b} = -1 \qquad (2.2.18)$$

En esta ecuación pueden verse las ventajas de la simplificación del error medio cuadrático al poder ser calculado por medio del error en la iteración k, y así para calcular el error se necesita solo multiplicar el error por el número de entradas.

Esta aproximación de $\nabla e(k)$, puede ser usada en el algoritmo de pasos descendientes tal como aparece en la ecuación (2.2.5) para darle forma final a la actualización de pesos y ganancias del algoritmo LMS de las ecuaciones (2.2.14) y (2.2.15)

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + 2\alpha e(k)\,\boldsymbol{p}(k) \qquad (2.2.19)$$

$$b(k+1) = b(k) + 2\alpha e(k) \qquad (2.2.20)$$

La rata de aprendizaje α se tomó constante durante el proceso de deducción del algoritmo.

En forma matricial el algoritmo de actualización para pesos y ganancias para la red Adaline, se expresa como:

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + 2\alpha \boldsymbol{e}(k)\ \boldsymbol{p}^{T}(k) \qquad (2.2.21)$$

$$\boldsymbol{b}(k+1) = \boldsymbol{b}(\mathrm{k}) + 2\alpha \boldsymbol{e}(k) \qquad (2.2.22)$$

Nótese que ahora el error $\boldsymbol{e}$ y la ganancia $\boldsymbol{b}$ son vectores

2.2.4 Principal aplicación 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 está haciendo es, modificar las características de respuesta en frecuencia de un filtro *pasa banda* que está dentro de la radio, este filtro sólo deja pasar las señales procedentes de la emisora en la que se esté interesado y elimina todas las demás señales que estén siendo transmitidas dentro del espectro AM.

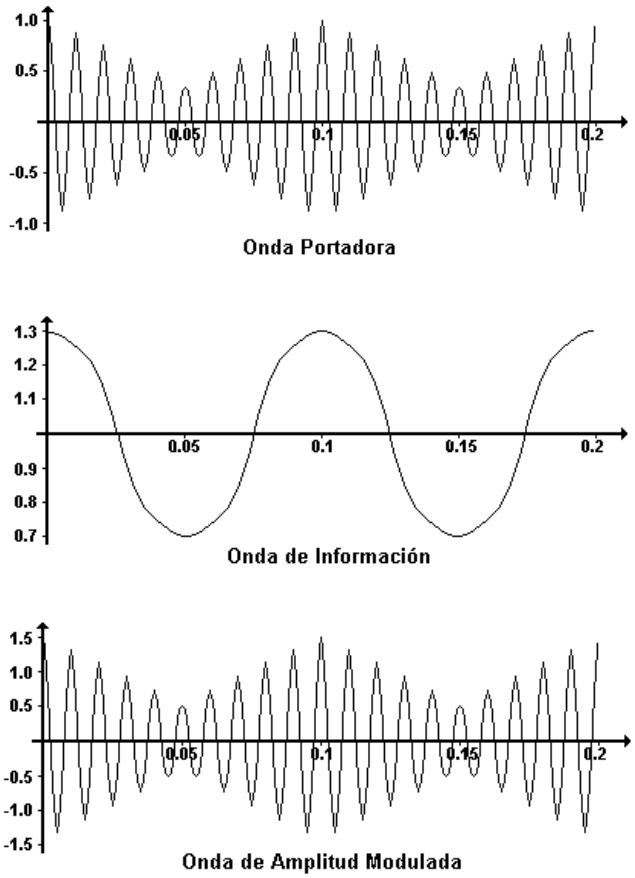

Figura 2.2.5 Técnicas de codificación de información y modulación en amplitud

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:

$$x(t) = \sum_{n=0}^{\infty} a_n Cos(2\pi n f_0 t) + \sum_{n=1}^{\infty} b_n Sen(2\pi n f_0 t) \qquad (2.2.23)$$

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 de la figura 2.2.6 poseían una gran limitación: solo eran ajustables en un pequeño intervalo

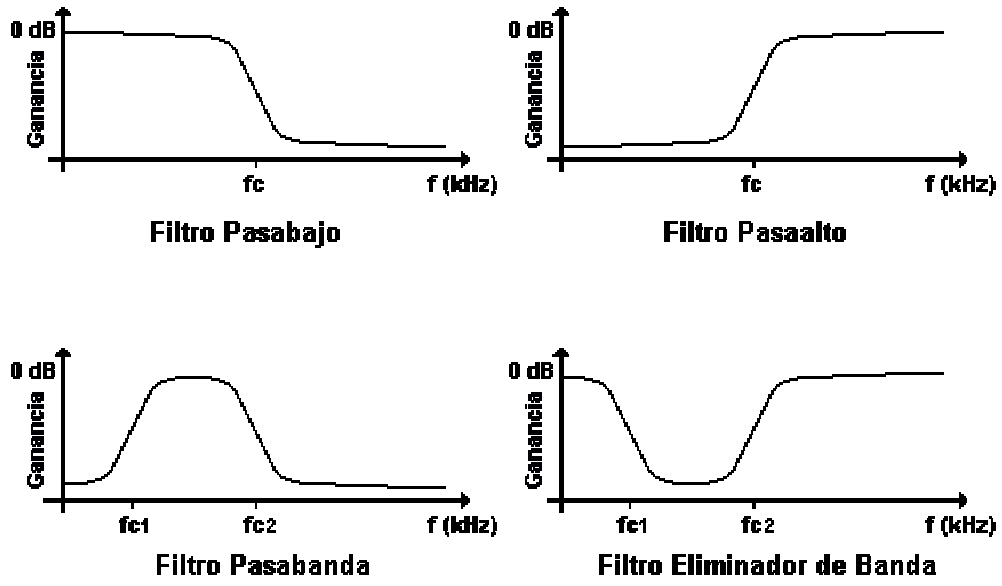

Figura 2.2.6 Características de los cuatro filtros básicos

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:

$$h(n) = R[\delta(n)] \tag{2.2.24}$$

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

$$y(n) = R[x(n)] = \sum_{i=-\infty}^{\infty} h(i)\, x(n-i) \tag{2.2.25}$$

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 cálculo para determinar cuánta 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 cuáles son los detalles de las características de las señales; si se necesitarán 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, el cual se visualiza en la figura 2.2.7

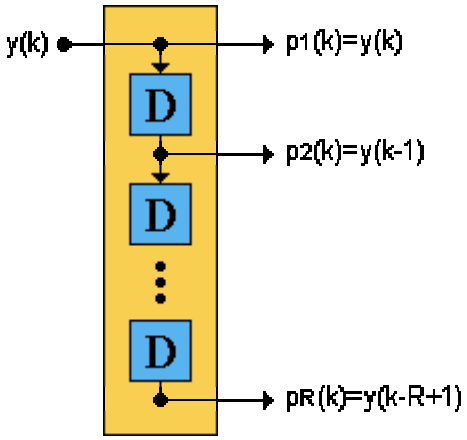

Figura 2.2.7 Retardos en línea

Si se combina la red Adaline con un bloque de retardos en línea, se ha creado un filtro adaptivo como el de la figura 2.2.8

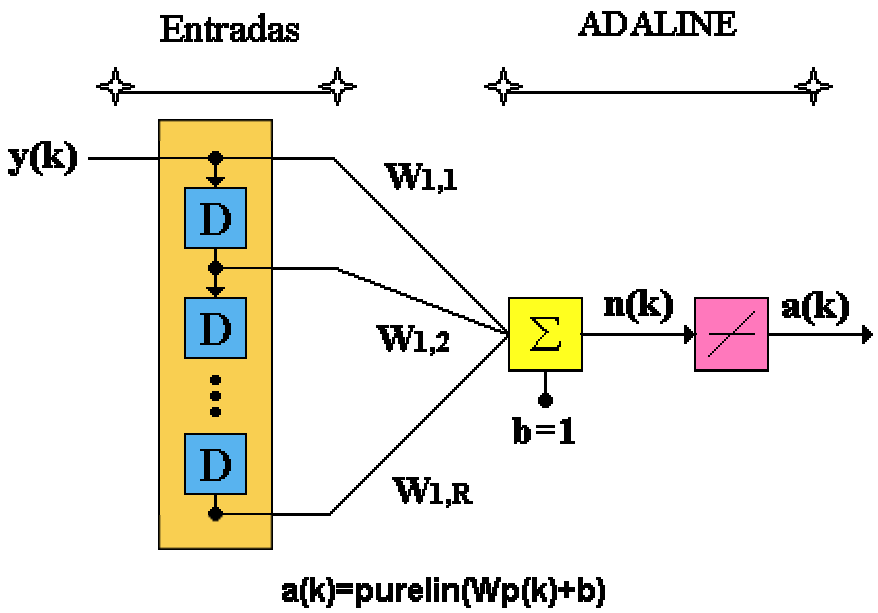

Figura 2.2.8 Filtro adaptivo

Cuya salida está dada por:

$$a(k) = purelin = (\boldsymbol{W}\boldsymbol{p} + b) = \sum_{i=1}^{R} w_{1,i}\, y(k-i+1) + b \qquad (2.2.26)$$